

С. В. РЫБИН

ДИСКРЕТНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

Учебник



ЛАНЬ

САНКТ-ПЕТЕРБУРГ
МОСКВА
КРАСНОДАР
2022

УДК 51
ББК 22.176я73

Р 93 Рыбин С. В. Дискретная математика и информатика : учебник для вузов / С. В. Рыбин. — Санкт-Петербург : Лань, 2022. — 748 с. : ил. — Текст : непосредственный.

ISBN 978-5-8114-8566-6

Материал учебника скомпонован так, чтобы, с одной стороны, дать темы для практических занятий, познакомить студентов с важными идеями на несложных примерах, дать им возможность в совершенстве освоить необходимую технику вычислений, обсуждаемые алгоритмы, а с другой — последовательно и доказательно изложить теоретический материал, который может быть осмыслен на разных уровнях формализма и не обязательно при первом прочтении учебника.

Учебник строится на базе известных из курса средней школы математических идей. Идеи эти достаточно разноплановы, чтобы заинтересовать людей с различными интересами и разной математической подготовкой.

Обсуждаемые в учебнике идеи быстро приводят к интересным приложениям математической теории к практике, появляется возможность экспериментировать с ними уже с первых месяцев обучения в вузе. Эти эксперименты с прикладными алгоритмами могут реализовываться в курсе информатики и программирования, обычно читающемся параллельно с курсом дискретной математики, либо в индивидуальной работе студентов.

Учебник будет полезен студентам технических университетов, специализирующимся в направлении «Информатика».

УДК 51
ББК 22.176я73

Рецензент

Ф. А. НОВИКОВ — доктор технических наук, профессор Высшей школы прикладной математики и вычислительной физики Санкт-Петербургского политехнического университета Петра Великого.

Обложка
П. И. ПОЛЯКОВА

© Издательство «Лань», 2022
© С. В. Рыбин, 2022
© Издательство «Лань»,
художественное оформление, 2022

Оглавление

Предисловие	11
1. Целочисленные алгоритмы	13
1.1. Арифметика целых чисел	13
1.1.1. Деление с остатком	13
1.1.2. Наибольший общий делитель, наименьшее общее кратное и их свойства	14
1.1.3. Простые числа	17
1.1.4. Решето Эратосфена. Разложение числа на простые множители	19
1.1.5. Позиционная запись натуральных чисел	27
1.1.6. Алгоритмы арифметических действий с p -ичными записями натуральных чисел	29
1.1.7. Алгоритмы перевода p -ичной записи натурального числа в q -ичную	31
1.1.8. Алгоритм эффективного возведения числа в натуральную степень	32
Задачи и вопросы	34
1.2. Алгоритм Евклида и цепные дроби	36
1.2.1. Классический алгоритм Евклида	36
1.2.2. Бинарный алгоритм Евклида	37
1.2.3. Линейное представление наибольшего общего делителя	39
1.2.4. Обобщенный алгоритм Евклида	40
1.2.5. Разложение числа в цепную дробь	42
1.2.6. Свойства подходящих дробей и их вычисление	45
1.2.7. Цепные дроби и анализ алгоритма Евклида	46
1.2.8. Бесконечная цепная дробь и ее вычисление	51
1.2.9. Наилучшие приближения	58
1.2.10. Некоторые применения цепных дробей	60
1.2.11. Диофантовы уравнения	61
1.2.12. Числа Фибоначчи	66
Задачи и вопросы	70
1.3. Арифметика остатков	71
1.3.1. Алгебраические структуры	71
1.3.2. Арифметика и свойства сравнений	73
1.3.3. Функция Эйлера и ее свойства	77

1.3.4.	Линейные сравнения	85
1.3.5.	Китайская теорема об остатках	89
1.3.6.	Система остаточных классов	93
1.3.7.	Применение теоремы Эйлера в криптографии. Система шифрования RSA	94
1.3.8.	Простейшие атаки на систему RSA	99
1.3.8.1.	Схема с нотариусом	100
1.3.8.2.	Метод бесключевого чтения RSA, или циклическая атака	100
1.3.9.	Другие примеры использования RSA	101
1.3.9.1.	Электронная подпись	101
1.3.9.2.	Электронные деньги	102
	Задачи и вопросы	104
1.4.	Арифметика многочленов	106
1.4.1.	Основные операции и свойства	106
1.4.2.	Схема Горнера	114
1.4.3.	Алгоритм Евклида для многочленов. Линейное представление НОД	116
1.4.4.	Интерполяционная формула Лагранжа	118
1.4.5.	Разложение многочлена на свободные от квадратов множители	120
1.4.6.	Разложение многочленов на неприводимые множители	122
1.4.6.1.	Приводимость над полем рациональных чисел	123
1.4.6.2.	Связь неприводимости многочленов с целыми коэффициентами над \mathbb{Z}_p и \mathbb{Q}	125
	Задачи и вопросы	126
1.5.	Базисы Гребнера	128
1.5.1.	Задачи на доказательство и многочлены	128
1.5.2.	Случай многочленов одной переменной	129
1.5.3.	Понятие идеала	130
1.5.3.1.	Идеалы в кольце многочленов одной переменной	132
1.5.4.	Линейные многочлены многих переменных. Понятие базиса Гребнера	132
1.5.5.	Алгоритм Бухбергера	135
1.5.5.1.	Главный член многочлена с несколькими переменными	135

1.5.5.2.	Редукция многочлена с несколькими переменными	136
1.5.5.3.	Построение S -многочленов	139
1.5.5.4.	Построение базиса Гребнера	140
	Задачи и вопросы	145
1.6.	Теория множеств и комбинаторика	147
1.6.1.	Вводные задачи	147
1.6.1.1.	Принцип умножения	147
1.6.1.2.	Принцип сложения	148
1.6.1.3.	Использование взаимно однозначного соответствия множеств	150
1.6.1.4.	Различные способы рассуждений	152
1.6.1.5.	Принцип использования разных языков описания	153
1.6.1.6.	Принцип формального оперирования алгебраическими объектами вместо комбинаторных рассуждений	154
1.6.1.7.	Принцип Дирихле	155
1.6.2.	Размещения и сочетания	157
1.6.3.	Биномиальные коэффициенты	160
1.6.4.	Кодирование с исправлением ошибок. Граница Хемминга	162
1.6.5.	Полиномиальное кодирование	165
1.6.6.	Код Шеннона — Фано и алгоритм Хаффмана	171
1.6.7.	Лексикографический порядок. Генерация подмножеств	173
1.6.8.	Перечислительная комбинаторика	175
1.6.8.1.	Перечисление подмножеств	176
1.6.8.2.	Перечисление элементов декартова произведения множеств	180
1.6.8.3.	Перечисление перестановок	182
1.6.9.	Бинарный код Грея	188
1.6.9.1.	Код Грея и задача о Ханойских башнях	189
1.6.10.	Числа Стирлинга первого и второго родов	190
1.6.11.	Числа Каталана	193
1.6.12.	Разбиения чисел	198
1.6.13.	Введение в динамическое программирование. Задача о рюкзаке	202
1.6.14.	Принцип включения-исключения	205
1.6.15.	Обращение Мебиуса	210

1.6.16.	Понятие о группе: теорема Бернсайда	214
1.6.16.1.	Самосовмещения правильного тетраэдра	214
1.6.16.2.	Подгруппы и классы смежности	217
1.6.16.3.	Теорема Бернсайда	222
	Задачи и вопросы	228
1.7.	Производящие функции и рекуррентные уравнения	231
1.7.1.	Производящие функции	231
1.7.2.	Решение однородного линейного рекуррентного уравнения	236
1.7.3.	Решение неоднородного линейного рекуррентного уравнения	243
1.7.4.	Производящая функция для чисел Каталана	245
1.7.5.	Число многочленов заданной степени, неприводимых над полем вычетов	247
	Задачи и вопросы	250
1.8.	Элементы дискретной теории вероятностей	251
1.8.1.	Основные определения	251
1.8.2.	Условные вероятности и формула Байеса	253
1.8.3.	Дискретные случайные величины. Математическое ожидание и дисперсия	258
	Задачи и вопросы	262

2. Графы и бинарные отношения 263

2.1.	Основные понятия теории графов	263
2.1.1.	Простейшие определения и свойства	263
2.1.2.	Машинное представление графов	270
2.1.2.1.	Матрица инциденций	270
2.1.2.2.	Матрица смежности	271
2.1.2.3.	Списки инцидентности	272
2.1.2.4.	Список ребер	273
2.1.3.	Поиск в глубину и ширину в графе	273
2.1.4.	Связность	278
2.1.5.	Эйлеровы графы	289
2.1.6.	Графы де Брюина	298
2.1.7.	Гамильтоновы графы	303
2.1.8.	Турниры	308
2.1.9.	Деревья	310
2.1.10.	Корневые деревья	313
2.1.11.	Код Прюфера	317
2.1.12.	Главные циклы и коциклы	320

2.1.13.	Двудольные графы	329
2.1.14.	Клики и независимые множества	333
2.1.15.	Планарность	337
2.1.16.	Раскраска графов	345
2.1.17.	Хроматические многочлены	354
	Задачи и вопросы	362
2.2.	Алгоритмы на графах	367
2.2.1.	Построение наибольшего паросочетания	367
2.2.2.	Построение минимальных стягивающих деревьев	373
2.2.3.	Задача нахождения кратчайших путей в графе	385
2.2.4.	Алгоритм Форда — Беллмана	386
2.2.5.	Алгоритм Форда — Беллмана для нахождения дерева кратчайших путей	391
2.2.6.	Алгоритм Дейкстры	396
2.2.7.	Сравнительный анализ трудоемкости алгоритмов Форда — Беллмана и Дейкстры	402
2.2.8.	Алгоритм Флойда	403
2.2.9.	Волновой алгоритм	411
2.2.10.	Алгоритм Форда — Фалкерсона	414
2.2.11.	Перебор с возвратом	426
2.2.12.	Метод ветвей и границ	430
	Задачи и вопросы	433
2.3.	Бинарные отношения	436
2.3.1.	Введение. Постановка задачи	436
2.3.2.	Свойства бинарных отношений	437
2.3.3.	Способы задания отношений	440
2.3.4.	Отношения эквивалентности и толерантности	442
2.3.5.	Отношения предпорядка и порядка	444
2.3.6.	Алгоритм топологической сортировки	447
2.3.7.	Частично упорядоченные множества. Диаграмма Хассе	450
2.3.8.	Транзитивное замыкание бинарного отношения. Алгоритм Уоршелла	452
2.3.9.	Примеры на бинарные отношения	458
2.3.10.	Введение в семантические сети	459
	Задачи и вопросы	460

3. Математическая логика и теория алгоритмов 463

3.1.	Логика высказываний	463
------	-------------------------------	-----

3.1.1.	Основные понятия логики высказываний	463
3.1.2.	Равносильность формул логики высказываний	470
3.1.3.	Принцип двойственности	473
3.1.4.	Логическое следствие	475
3.1.5.	Нормальные формы в логике высказываний	479
3.1.6.	Контактные схемы	488
3.1.7.	Метод минимизирующих карт	491
3.1.8.	Метод Куайна — Мак-Класки	498
3.1.9.	Метод резолюций в логике высказываний	503
3.1.10.	Стратегии метода резолюций	508
3.1.11.	Линейная резолюция	512
3.1.12.	Решетки, булевы алгебры, кольца	513
	Задачи и вопросы	519
3.2.	Булевы функции	521
3.2.1.	Основные понятия	521
3.2.2.	Замкнутость и полнота	526
3.2.3.	Полиномы Жегалкина и линейные функции	529
3.2.4.	Быстрое вычисление полинома Жегалкина	536
3.2.5.	Булевы функции в криптографии	538
3.2.6.	Самодвойственные функции	539
3.2.7.	Монотонные функции	544
3.2.8.	Теорема Поста	548
3.2.9.	Разложение Шеннона	555
3.2.10.	Бинарные диаграммы решений	556
	Задачи и вопросы	563
3.3.	Логика предикатов	566
3.3.1.	Основные определения	566
3.3.2.	Семантика логики предикатов	569
3.3.3.	Примеры и задачи на выразительность логики предикатов	571
3.3.4.	Равносильность формул логики предикатов	573
3.3.5.	Логическое следствие	576
3.3.6.	Нормальные формы	578
3.3.7.	Примеры и задачи на приведение к ПНФ и СНФ	582
3.3.8.	Метод резолюций в логике предикатов	583
	3.3.8.1. Подстановка и унификация	584
	3.3.8.2. Метод резолюций	588
3.3.9.	Метод резолюций и ПРОЛОГ	593
	Задачи и вопросы	596

3.4. Нечеткая логика	599
3.4.1. Нечеткие множества	599
3.4.1.1. Введение в нечеткие множества	599
3.4.1.2. Основные понятия	602
3.4.1.3. Операции над нечеткими множествами	606
3.4.1.4. Мера нечеткости множеств	611
3.4.1.5. Функции принадлежности	613
3.4.1.6. Методы построения функций принадлежности	617
3.4.2. Нечеткие бинарные отношения	619
3.4.2.1. Основные понятия	619
3.4.2.2. Свойства нечетких бинарных отношений	622
3.4.3. Нечеткая логика высказываний	624
3.4.3.1. Нечеткие высказывания	624
3.4.3.2. Нечеткие формулы логики высказываний	627
Задачи и вопросы	629
3.5. Теория алгоритмов	631
3.5.1. Машина Тьюринга и функции, вычислимые по Тьюрингу	631
3.5.2. Примеры и задачи на построение машин Тьюринга	637
3.5.3. Операции над машинами Тьюринга	641
3.5.4. Проблема останова машины Тьюринга	643
3.5.5. Частично рекурсивные функции	646
3.5.6. Нормальные алгорифмы Маркова	649
3.5.7. Примеры и задачи на нормальные алгорифмы Маркова	653
3.5.8. Вычислительная сложность алгоритма	656
3.5.9. Сложностные классы задач	658
Задачи и вопросы	660
3.6. Формальные языки и грамматики	662
3.6.1. Основные определения порождающих грамматик	663
3.6.2. Классификация грамматик	665
3.6.3. Контекстно-свободные грамматики	667
3.6.4. Синтаксический анализатор	670
3.6.5. Преобразования контекстно-свободных грамматик	673
3.6.6. Примеры на построение грамматик	676
3.6.7. Префиксная и постфиксная записи формул	677

3.6.8. Алгоритмы обработки скобочных записей	682
Задачи и вопросы	685
3.7. Конечные автоматы-распознаватели	688
3.7.1. Введение	688
3.7.2. Автоматы Мили и Мура	689
3.7.3. Примеры автоматов Мили и Мура	691
3.7.4. Эквивалентность автоматов Мили и Мура	692
3.7.5. Основные понятия автоматов-распознавателей	696
3.7.6. Примеры автоматов-распознавателей	697
3.7.7. Недетерминированные автоматы-распознаватели. Детерминизация	699
3.7.8. Теорема о разрастании для автоматных языков	703
3.7.9. Автоматы и автоматные грамматики	706
3.7.10. Минимизация автоматов-распознавателей	710
3.7.11. Конечные автоматы с эпсилон-переходами	714
3.7.12. Конечные автоматы и регулярные выражения	715
Задачи и вопросы	721
Список литературы	725
Список алгоритмов	731
Предметный указатель	734

ПРЕДИСЛОВИЕ

Данный материал написан на основе учебника «Дискретная математика» [49].

Учебник состоит из трех основных разделов.

1 Целочисленные алгоритмы. Здесь рассматриваются понятия из теории чисел и многочленов, важные для компьютерной математики и криптографии, классические комбинаторные идеи и их обобщения с прикладной проблематикой, в том числе генерированием комбинаторных объектов, кодированием. Подробно обсуждаются производящие функции.

2 Графы и бинарные отношения. В этом разделе обсуждаются понятия классической теории графов, большое внимание уделено алгоритмам на графах, обсуждаются общие принципы доказательства корректности алгоритмов и их эффективности. Подробно рассматриваются бинарные отношения.

3 Математическая логика и теория алгоритмов. В этот раздел включен материал по булевым функциям, обсуждаются основы исчисления предикатов. Много внимания уделяется различным формализациям понятий алгоритма, языка и грамматики. Завершается раздел рассмотрением конечных автоматов.

Материал будет полезен студентам технических университетов, специализирующихся в направлении «Информатика».

От автора

Со всеми замечаниями и предложениями обращаться по адресу:
`rsvvm2leti@yandex.ru`.

ГЛАВА

1

ЦЕЛОЧИСЛЕННЫЕ АЛГОРИТМЫ

1.1. Арифметика целых чисел

1.1.1. Деление с остатком

Теорема 1.1 (о делимости с остатком). Для любых $a, b \in \mathbb{Z}$, где $b \neq 0$, существуют, и притом единственные, $q, r \in \mathbb{Z}$ такие, что имеет место представление

$$a = bq + r, \quad 0 \leq r < |b|. \quad (1.1)$$

Доказательство. Не умаляя общности, предположим, что $a > b > 0$. Все остальные случаи или сводятся к этому, или являются тривиальными. Далее будем рассматривать числа, кратные b , т. е. числа вида $b, 2b, 3b, \dots, kb \dots$ Очевидно, что найдется число $q \geq 1$ такое, что

$$bq \leq a < (q + 1)b. \quad (1.2)$$

Обозначим $r = a - bq$. Из (1.2) следует, что $0 \leq r < b$. Представление (1.1) установлено.

Докажем единственность представления (1.1). Пусть

$$a = bq + r = bq_1 + r_1, \quad 0 \leq r < |b|, \quad 0 \leq r_1 < |b|.$$

Тогда $b(q - q_1) = r_1 - r$. Но $|r_1 - r| < |b|$, а $|b(q - q_1)| \geq |b|$. Противоречие, таким образом, $q = q_1, r = r_1$. ■

Число q в (1.1) называют целой частью дроби $\frac{a}{b}$ и обозначают $\left\lfloor \frac{a}{b} \right\rfloor$ или $a \div b$, а число r — остатком и обозначают $\langle a \rangle_b$ или $a \bmod b$.

Определение 1.1. Будем говорить, что a делится на b (и обозначать как $a : b$), если в представлении (1.1) для a, b остаток r равен нулю. Выражение b делит a часто обозначают как $b \mid a$.

Историческая справка

Знак $a : b$ для обозначения операции деления числа a на число b впервые ввел Г. Лейбниц в 1684 г.

Сформулируем простейшие свойства делимости, вытекающие непосредственно из определения.

Теорема 1.2. Справедливы следующие утверждения:

- 1) $b \mid a, c \mid b$, тогда $c \mid a$ — транзитивность деления;
- 2) $c \mid a_i, i \in 1 : k$, тогда для любого набора $\{\lambda_i \in \mathbb{Z}, i \in 1 : k\}$ справедливо $c \mid \sum_{i=1}^k a_i \lambda_i$;
- 3) $b \mid a$, тогда $\pm b \mid \pm a$.

Доказательство. Докажем, например, первое свойство. Из условия теоремы имеем $a = bq$ и $b = cl$, тогда $a = (cl)q = c(lq)$. ■

Упражнение 1.1. Докажите самостоятельно свойства 2 и 3 теоремы 1.2.

Замечание 1.1

Последнее свойство позволяет в вопросах делимости ограничиться рассмотрением только натуральных чисел.

1.1.2. Наибольший общий делитель, наименьшее общее кратное и их свойства

Определение 1.2. Пусть $a_1, \dots, a_k, c \in \mathbb{N}$. Если $a_1 \mid c, \dots, a_k \mid c$, то говорят, что c есть *общее кратное* a_1, \dots, a_k . Наименьшее среди всех кратных называют *наименьшим общим кратным* (НОК) и обозначают $M(a_1, \dots, a_k)$ или $\{a_1, \dots, a_k\}$. Заметим, что

$$\max\{a_1, a_2, \dots, a_k\} \leq M(a_1, \dots, a_k) \leq \prod_{i=1}^k a_i.$$

Очевидно, что любое кратное делится на наименьшее общее кратное. Действительно, пусть m — общее кратное a_1, a_2, \dots, a_k . Разделим m на M с остатком. Тогда согласно (1.1)

$$m = M(a_1, \dots, a_k)q + r, \quad 0 \leq r < M(a_1, \dots, a_k). \quad (1.3)$$

Из (1.3) получаем, что r — также общее кратное a_1, \dots, a_k , причем меньшее $M(a_1, \dots, a_k)$. Следовательно, $r = 0$.

Определение 1.3. Пусть $a_1, \dots, a_k, d \in \mathbb{N}$. Если $d \mid a_1, \dots, d \mid a_k$, то говорят, что d есть *общий делитель* a_1, \dots, a_k . Наибольший среди всех делителей называют *наибольшим общим делителем* (НОД) и обозначают $D(a_1, \dots, a_k)$ или (a_1, \dots, a_k) . Заметим, что

$$1 \leq D(a_1, \dots, a_k) \leq \min\{a_1, \dots, a_k\}.$$

Покажем, что если d_1, \dots, d_n — общие делители a_1, \dots, a_k , то

$$D(a_1, \dots, a_k) = M(d_1, \dots, d_n). \quad (1.4)$$

Действительно, любое a_i по определению является общим кратным чисел d_1, \dots, d_n . Следовательно, $M(d_1, \dots, d_n) \mid a_i$, тогда $M(d_1, \dots, d_n)$ есть общий делитель всех a_i . Но согласно определению $d_i \mid M(d_1, \dots, d_n)$, $i \in \{1, \dots, n\}$, отсюда получаем (1.4).



Замечание 1.2

Равенство (1.4) можно использовать в качестве другого определения наибольшего общего делителя.

Введем одно из важнейших понятий теории делимости.

Определение 1.4. Числа a_1, \dots, a_k называют *взаимно простыми*, если

$$D(a_1, \dots, a_k) = 1.$$

Установим некоторые свойства наибольшего общего делителя и наименьшего общего кратного.

Теорема 1.3. Справедливы следующие утверждения:

- 1) $d = D(a_1, \dots, a_k) \Leftrightarrow D\left(\frac{a_1}{d}, \dots, \frac{a_k}{d}\right) = 1$;
- 2) $d = D(a_1, \dots, a_k)$, тогда $D(a_1b, \dots, a_kb) = db$;
- 3) если c — общий делитель a_1, \dots, a_k , то $D\left(\frac{a_1}{c}, \dots, \frac{a_k}{c}\right) = \frac{d}{c}$;

$$4) ab = D(a, b)M(a, b).$$

Доказательство. Утверждения 1–3 достаточно просты и предлагаются для самостоятельной работы. Докажем утверждение 4. Имеем ab , кратное a и b , поэтому $M(a, b) \mid ab$. Тогда

$$d := \frac{ab}{M(a, b)}, \quad \frac{a}{d} = \frac{M(a, b)}{b}, \quad \frac{b}{d} = \frac{M(a, b)}{a}. \quad (1.5)$$

Так как $M(a, b)$ делится на a и b , из (1.5) следует, что d — общий делитель a и b . Пусть d_1 — произвольный делитель этих чисел. Тогда

$$\frac{ab}{d_1} = a \left(\frac{b}{d_1} \right) = b \left(\frac{a}{d_1} \right), \quad \text{следовательно, } \frac{ab}{d_1} \text{ — общее кратное } a, b, \text{ и}$$

$$\frac{ab}{d_1} : M(a, b) = \frac{d}{d_1} \in \mathbb{Z},$$

получаем для произвольного делителя $d_1 \mid d$, таким образом $d = D(a, b)$. ■

Упражнение 1.2. Докажите самостоятельно утверждения 1–3 теоремы 1.3.

Вопрос о нахождении $D(a, b)$ будет решен далее (см. раздел 1.2.1). Предположим, что имеется эффективный алгоритм его вычисления. Поставим вопрос о вычислении $D(a_1, \dots, a_k)$. Ответ на этот вопрос дает следующая теорема.

Теорема 1.4. Справедливо равенство

$$D(a_1, a_2, a_3) = D(D(a_1, a_2), a_3).$$

Доказательство. Введем обозначения: $D(a_1, a_2) = e$, $D(e, a_3) = d$. Тогда в силу транзитивности делимости (теорема 1.2) имеем $a_1, a_2 : d$, но и $a_3 : d$, следовательно, d — общий делитель a_1, a_2, a_3 .

Пусть d_1 — произвольный общий делитель a_1, a_2, a_3 . Тогда $e : d_1$, следовательно, d_1 — общий делитель e, a_3 . Тогда $d : d_1$ и, следовательно, $d = D(a_1, a_2, a_3)$. ■

Для наименьшего общего кратного можно получить результат, аналогичный теореме 1.4.

Теорема 1.5. Справедливо равенство

$$M(a_1, a_2, a_3) = M(M(a_1, a_2), a_3).$$

Установим еще несколько свойств взаимно простых чисел.

Теорема 1.6. Если $c \mid ab$ и $D(a, c) = 1$, тогда $c \mid b$.

Доказательство. $c \mid ab$ и $a \mid ab$, следовательно, ab — кратное чисел a и c . Тогда $M(a, c) \mid ab$, но согласно теореме 1.3 $M(a, c) = ac$. Следовательно, $ab \div ac$. Отсюда получаем требуемое. ■

Следствие 1.1. Пусть a имеет набор взаимно простых делителей:

$$\{m_1, \dots, m_k \mid a \div m_i, i \in D(m_i, m_j) = 1, i \neq j\}.$$

Тогда $m_1 m_2 \dots m_k$ тоже делитель числа a .

Доказательство. Докажем утверждение для $k = 2$. Имеем $m_1 \mid a$, следовательно, $a = sm_1$. Тогда $m_2 \mid sm_1$ и $D(m_1, m_2) = 1$, по теореме 1.6 получаем $m_2 \mid s$. Таким образом, $m_1 m_2 \mid a$. Проведенные рассуждения легко продолжить для любого k . ■

Упражнение 1.3. Докажите утверждение: если $D(a, c) = 1$, то

$$D(ab, c) = D(b, c).$$

1.1.3. Простые числа

Определение 1.5. Натуральное число $p > 1$ называют *простым*, если оно делится только на $\pm p$ и на ± 1 .

Теорема 1.7. Пусть p — простое число. Справедливо утверждение: $p \mid ab$ тогда и только тогда, когда $p \mid a$ или $p \mid b$.

Доказательство. Пусть $p \mid ab$. Предположим, что a не делится на p . Тогда $D(a, p) = 1$. По теореме 1.6 получаем требуемое. В обратную сторону утверждение теоремы очевидно. ■

Лемма 1.1. Любое $a \in \mathbb{Z}, a \neq 1$ имеет по крайней мере один простой делитель.

Доказательство. Пусть d_1, \dots, d_n — все положительные делители числа a , кроме 1. Положим $p = \min\{d_1, \dots, d_n\}$. Если бы p было составным, то его делитель (меньший, чем p) был бы делителем a . Противоречие с определением p . ■

Теорема 1.8 (основная теорема теории делимости). Любое число $a \in \mathbb{Z}$ раскладывается и только одним способом на простые сомножители. Соединив одинаковые множители в степени, получаем *каноническое разложение*

$$a = p^\alpha q^\beta r^\gamma \dots, \text{ где } p, q, r \text{ — простые числа; } \alpha, \beta, \gamma \geq 1. \quad (1.6)$$

Доказательство. По лемме 1.1 любое число a имеет простой делитель p . Представим его в виде pa_1 . Если a_1 — составное, то воспользуемся леммой 1.1 для a_1 . Заметим, что $a_1 < a$, поэтому на некотором шаге получим a_k простое. Таким образом, имеем представление

$$a = \prod_{i=1}^k p_i, \quad \text{где } p_i, i \in 1 : k \text{ простые.}$$

Объединив одинаковые множители в степени, приходим к (1.6).

Докажем единственность представления (1.6). Пусть имеются два различных представления

$$a = p_1 p_2 p_3 \cdots p_k = q_1 q_2 q_3 \cdots q_n. \quad (1.7)$$

Тогда $p_1 p_2 p_3 \cdots p_k : q_1$. По теореме 1.7 один из сомножителей слева делится на q_1 . Пусть для определенности, $p_1 : q_1$. Но p_1 и q_1 — простые числа. Поэтому $p_1 = q_1$. Тогда (1.7) можно переписать в виде

$$p_2 p_3 \cdots p_k = q_2 q_3 \cdots q_n.$$

Повторив эту процедуру, приходим к единственности представления (1.6). ■

Следствие 1.2. Пусть a и b представлены в каноническом виде (1.6):

$$a = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_n^{\alpha_n}, \quad b = p_1^{\beta_1} p_2^{\beta_2} \cdots p_n^{\beta_n}.$$

Тогда

$$D(a, b) = p_1^{\gamma_1} p_2^{\gamma_2} \cdots p_n^{\gamma_n}, \quad M(a, b) = p_1^{\delta_1} p_2^{\delta_2} \cdots p_n^{\delta_n},$$

где $\gamma_i = \min\{\alpha_i, \beta_i\}$, а $\delta_i = \max\{\alpha_i, \beta_i\}$.

Историческая справка

В 1876 г. французский математик Ф. Люка доказал, что число $2^{127} - 1$ является простым, и 75 лет оно оставалось наибольшим из известных простых чисел, что не покажется удивительным, если на него взглянуть:

$$2^{127} - 1 = 170141183460469231731687303715884105727.$$

В настоящее время составлены таблицы всех простых чисел, не превосходящих 50 миллионов, далее известны только отдельные их представители.

Наибольшее известное (на декабрь 2020 г.) простое число: $2^{28258993} - 1$ имеет в записи 24862048 десятичных цифр. Оно было найдено в 2018 г. в рамках онлайн-проекта «Great Internet Mersenne Prime Search (GIMPS)».

Определение 1.6. Число $a \in \mathbb{Z}$ называется *свободным от квадратов*, или *бесквадратным*, если не делится ни на один квадрат целого числа, кроме 1.

Например, 14 — свободное от квадратов, а 12 — нет, так как $12 = 2^2 \times 3$.

Очевидно следующее утверждение.

Лемма 1.2. Число $n \in \mathbb{N}$ свободно от квадратов тогда и только тогда, когда для любого его разложения на множители $n = ab$, $D(a, b) = 1$.

1.1.4. Решето Эратосфена. Разложение числа на простые множители

В связи с полученным представлением (1.6) возникает необходимость построить эффективные алгоритмы для решения следующих задач:

- 1) найти все простые числа в данном интервале;
- 2) для любого числа $a \in \mathbb{Z}$ получить его разложение в виде (1.6).

Для решения первой задачи рассмотрим алгоритм, который называют **решетом Эратосфена**. Он позволяет найти все простые числа, не превосходящие N .

Алгоритм 1.1. Решето Эратосфена

```

Выписываем подряд все целые числа от 2 до  $N$ 
 $p \leftarrow 2$ 
while  $p < N$  do
    Вычеркиваем все числа, от  $2p$  до  $N$ , делящиеся на  $p$ 
     $p \leftarrow$  первое невычеркнутое число, большее  $p$ 
end while

```

Замечание 1.3

Алгоритм 1.1 можно улучшить следующим образом. Обращаться можно только к нечетным числам, начиная с $p = 3$. Числа можно вычеркивать, начиная сразу с p^2 , потому что все составные числа меньше его уже будут вычеркнуты к этому времени. И, соответственно, останавливать алгоритм можно, когда $p^2 \geq N$. Это вытекает из теоремы 1.9.

Теорема 1.9. Всякое составное число a имеет делитель $b \leq \sqrt{a}$.

Доказательство. Если $a = bc$, то из пары чисел b, c одно больше \sqrt{a} , а другое меньше, за исключением случая, когда a — точный квадрат, тогда $b = c = \sqrt{a}$. ■

Историческая справка

Эратосфен Киренский (276–194 гг. до н. э.) — греческий математик, астроном, географ и поэт, заведовал великой Александрийской библиотекой.

Один из интересных фактов жизни Эратосфена — вычисление радиуса Земли. Результат оказался всего лишь на 80 км меньше, чем фактический радиус. Эратосфена можно считать также основателем научной хронологии.

Пример 1.1 (решето Эратосфена). Пусть $N = 50$. Тогда после окончания работы алгоритма (при $p = 7$) получаем следующую картину (на месте вычеркнутых чисел стоит знак *):

3 5 7 * 11 13 * 17 19 * 23 * * 29 31 * * 37 * 41 43 * 47 * .

Возникает вопрос: сколько же простых чисел? Ответ был получен еще Евклидом:

Теорема 1.10 (Евклид). Множество простых чисел бесконечно.

Доказательство. Допустим конечность множества простых чисел: $\{p_1, \dots, p_k\}$. Положим $p = p_1 \cdots p_k + 1$. Очевидно, что число p не делится ни на одно из p_i . Таким образом, либо p — простое, либо имеет простой делитель, больший любого из p_i . Полученное противоречие и доказывает теорему. ■

Мы показали, что число простых чисел бесконечно. Пусть $\pi(n)$ — функция, возвращающая число простых чисел, меньших n . Точно задать эту функцию не удастся, но для нее есть хорошая оценка. Справедлива следующая теорема:

Теорема 1.11 (Чебышев).

$$\lim_{x \rightarrow \infty} \frac{\pi(x)}{x} = \frac{1}{\ln(x)} \quad (1.8)$$

Предельное соотношение (1.8) означает, что у случайно выбранного числа от 1 до n шанс оказаться простым примерно равен $1/\ln(n)$. Таким образом, если взять порядка $\ln(n)$ случайных чисел от 1 до n , то при больших n высока вероятность, что хотя бы одно из этих чисел будет простым. Значения $1/\ln(n)$ для различных n приведены в таблице 1.1.

Функция $\pi(n)$ асимптотически сверху приближается к своему пределу, так что оценка (1.8) дает слегка заниженные значения.

Перейдем к решению второй поставленной задачи — разложению числа a на простые множители в виде (1.6). Отыскание простых множителей натурального числа называют для краткости «*факторизацией*» («фактор» значит — множитель, составная часть; в этом последнем смысле слово обычно и употребляется).

Таблица 1.1.

n	10^5	10^6	10^9	10^{12}	10^{15}
$\ln(n)$	11.51	13.81	20.72	27.63	34.53
$1/\ln(n)$	0.08686	0.07238	0.048254	0.036191	0.02895

Начнем с простейшего алгоритма, который называют **методом пробных делителей**.

Введем обозначения:

$\{2 = p_0 < \dots < p_n \leq \sqrt{a}\}$ — последовательность *пробных делителей* — простых чисел;

$\{d_0, \dots, d_s \leq \sqrt{a}\}$ — последовательность найденных делителей числа a .

Алгоритм 1.2. Метод пробных делителей

$k \leftarrow 0, i \leftarrow 0$

while $a > 1$ **do**

$q \leftarrow a \div p_k$

 // Деление с остатком: $a = qp_k + r$

$r \leftarrow a \bmod p_k$

if $r = 0$ **then**

 // p_k — делитель a , повторяем деление на p_k

$d_i \leftarrow p_k$

$i \leftarrow i + 1$

$a \leftarrow q$

else if $q > p_k$ **then**

$k \leftarrow k + 1$

 // Переходим к следующему пробному делителю

else

$d_i \leftarrow a$

 // a — простое число, завершаем работу

return

end if

end while

Пример 1.2 (метод пробных делителей). Рассмотрим работу алгоритма 1.2 на примере.

Пусть $a = 6930$. Возьмем последовательность пробных делителей

$$\{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, \dots, \}:$$

1) $a = 2 \cdot 3465, d_0 = 2, i = 1, a = 3465;$

2) $a = 2 \cdot 1732 + 1, k = 1;$

3) $a = 3 \cdot 1155, d_1 = 3, i = 2, a = 1155;$

4) $a = 3 \cdot 385, d_2 = 3, i = 3, a = 385;$

- 5) $a = 3 \cdot 128 + 1, k = 2;$
 6) $a = 5 \cdot 77, d_3 = 5, i = 4, a = 77;$
 7) $a = 5 \cdot 15 + 2, k = 3;$
 8) $a = 7 \cdot 11, d_4 = 7, i = 5, a = 11;$
 9) $a = 7 \cdot 1 + 4, d_5 = 11.$

Имеем: $a = 6930 = 2 \cdot 3 \cdot 3 \cdot 5 \cdot 7 \cdot 11 = 2 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11.$

Легко видеть, что описанный алгоритм 1.2 эффективно работает на небольших числах. При их увеличении быстро растет число «холостых» делений.

Рассмотрим алгоритм, предложенный французским математиком Пьером Ферма, который, *используя только операции умножения и сложения (без делений)*, позволяет представить любое число в виде произведения двух (не обязательно простых) сомножителей.

Историческая справка

Пьер Ферма (1601–1665), крупный юрист, видный общественный деятель своей родины — города Тулузы, занимался математикой в часы досуга. О жизни его известно мало, книг он не печатал. Оставшиеся после него рукописи были изданы его сыном уже после смерти отца. Ферма состоял в переписке почти со всеми выдающимися математиками той эпохи; такой крупный ученый, как Паскаль, считал его лучшим математиком своего времени. Одновременно с Декартом Ферма заложил основы аналитической геометрии, вместе с Паскалем — основы теории вероятностей. Но лучшие его открытия принадлежат теории чисел.

Не умаляя общности, можно считать, что исходное число a является нечетным. Выделить степени двойки достаточно легко (сдвигами вправо двоичного представления числа). Будем искать представление a в виде

$$a = x^2 - y^2 = (x - y)(x + y).$$

Введем обозначения: $R(x, y) = (x - y)(x + y) - a$. Требуется, поочередно увеличивая x и y на 1, добиться равенства $R(x, y) = 0$. Заметим, что

$$\begin{aligned} R(x + 1, y) &= (x - y + 1)(x + y + 1) - a = R(x, y) + 2x + 1, \\ R(x, y + 1) &= (x - y - 1)(x + y + 1) - a = R(x, y) - (2y + 1). \end{aligned}$$

Чтобы лишний раз не умножать на 2, введем обозначения: $R_x = 2x + 1$, $R_y = 2y + 1$. Тогда последние равенства примут следующий вид:

$$\begin{cases} R(x + 1, y) = R(x, y) + R_x, & (1.9a) \\ R(x, y + 1) = R(x, y) - R_y. & (1.9b) \end{cases}$$

Заметим, что при увеличении x или y на единицу R_x или R_y увеличиваются соответственно на двойку.

Будем считать, что известна приближительная оценка целой части \sqrt{a} , обозначим ее через h :

$$h^2 \leq a < (h + 1)^2 \quad (1.10)$$

Далее будет рассмотрен метод построения числа h , удовлетворяющего (1.10), см. алгоритм 1.4.

Алгоритм 1.3. Метод Ферма

```

 $R_x \leftarrow 2h + 1$  //  $x = h$ 
 $R_y \leftarrow 1$  //  $y = 0$ 
 $R(x, y) \leftarrow h^2 - a$ 
while  $R(x, y) \neq 0$  do
  if  $R(x, y) > 0$  then
     $R(x, y) \leftarrow R(x, y) - R_y$  // Переходим к  $R(x, y + 1)$  согласно 1.9b
     $R_y \leftarrow R_y + 2$ 
  else
     $R(x, y) \leftarrow R(x, y) + R_x$  // Переходим к  $R(x + 1, y)$  согласно 1.9a
     $R_x \leftarrow R_x + 2$ 
  end if
end while
 $a = \frac{R_x - R_y}{2} \left( \frac{R_x + R_y}{2} - 1 \right)$ 

```

Корректность работы описанного алгоритма устанавливает следующая теорема.

Теорема 1.12. Алгоритм 1.3 всегда завершает работу.

Доказательство. При доказательстве рассмотрим отдельно два случая составного и простого числа a на входе. Не умаляя общности, считаем, что a — нечетное число.

Пусть a — составное число, т. е. $a = bc$, $b \leq c$. При этом b и c нечетные числа. Тогда

$$a = bc = (x - y)(x + y) = x^2 - y^2.$$

Поскольку $x - y \leq x + y$, положим $b = x - y$ и $c = x + y$. Получим:

$$x = \frac{c + b}{2}, \quad y = \frac{c - b}{2}. \quad (1.11)$$

Так как b и c — нечетные числа, x и y — целые. При $b = c$ число a есть полный квадрат и алгоритм Ферма находит делитель на шаге 1. Будем предполагать, что a не является полным квадратом. Тогда

$$1 < b < c < a.$$

Покажем, что

$$\lfloor \sqrt{a} \rfloor < \frac{c+b}{2} < \frac{a+1}{2}. \quad (1.12)$$

Рассмотрим правое неравенство (1.12). Заменяя a на bc и вычитая из обеих частей $c+1$, мы приходим к неравенству

$$b-1 < bc-c.$$

Однако $b > 1$, поэтому обе части последнего неравенства можно разделить на $b-1$. В результате получаем равносильное неравенство $c > 1$. Таким образом, правая часть неравенства (1.12) доказана.

Установим теперь левое неравенство (1.12). Отметим, что, поскольку

$$\lfloor \sqrt{a} \rfloor \leq \sqrt{a},$$

достаточно доказать, что

$$\sqrt{a} \leq \frac{b+c}{2}.$$

Очевидно, что последнее неравенство выполняется тогда и только тогда, когда

$$a \leq \frac{(b+c)^2}{4}.$$

Запишем очевидное тождество

$$\left(\frac{c+b}{2}\right)^2 - \left(\frac{c-b}{2}\right)^2 = bc = a. \quad (1.13)$$

Отсюда

$$\left(\frac{c+b}{2}\right)^2 - a = \left(\frac{c-b}{2}\right)^2 \geq 0.$$

Обе части неравенства (1.12) установлены.

Вернемся к алгоритму. Напомним, что стартовые значения переменных x и y равны соответственно $\lfloor \sqrt{a} \rfloor$ и 1, одна из них увеличивается на 1 на каждом шаге алгоритма. Из неравенства (1.12) следует, что если число a составное, то x дойдет до значения $(c+b)/2$ раньше, чем до $(a+1)/2$. При

этом значения x и y алгоритма связаны равенством (1.11). Поэтому если a составное, то алгоритм всегда останавливается при некотором $x < (a+1)/2$, вычислив два делителя этого числа. ■

Замечание 1.4

Заметим, что данное составное число a можно представить в виде $a = bc$, где $1 < b < c < a$, возможно, несколькими различными способами. Какое из разложений находит алгоритм Ферма? Поиск начинается при $x = \lfloor \sqrt{a} \rfloor$, и x увеличивается по ходу алгоритма. Значит, найденные алгоритмом делители b и c таковы, что разность

$$\frac{b+c}{2} - \lfloor \sqrt{a} \rfloor$$

наименьшая из возможных.

Пример 1.3. Пусть $a = 2520$. Согласно алгоритму 1.4 находим $h = \lfloor \sqrt{a} \rfloor = 50$. Протокол работы алгоритма представлен в таблице 1.2. Поля, изменяющиеся на данной итерации, выделены серым цветом.

Таблица 1.2.

Итерация	$R(x, y)$	R_x	R_y	Итерация	$R(x, y)$	R_x	R_y
0	-20	101	1	6	56	103	11
1	81	103	1	7	45	103	13
2	80	103	3	8	32	103	15
3	77	103	5	9	17	103	17
4	72	103	7	10	0	103	19
5	65	103	9				

Имеем: $a = 2520 = 42 \times 60$.

Пример 1.4. Пусть $a = 221$, тогда $h = \lfloor \sqrt{a} \rfloor = 14$. Протокол работы алгоритма представлен в таблице 1.3.

Таблица 1.3.

Итерация	$R(x, y)$	R_x	R_y	Итерация	$R(x, y)$	R_x	R_y
0	-25	29	1	2	3	31	3
1	4	31	3	3	0	31	5

Имеем: $a = 221 = 17 \times 13$.

Комбинируя методы Ферма и пробных делителей, можно построить эффективный алгоритм для разложения достаточно большого числа a на простые множители в виде (1.6).

Вернемся к вопросу эффективного построения целочисленного приближения квадратного корня, удовлетворяющего неравенству (1.10). Для ее решения можно воспользоваться целочисленной версией алгоритма на основе *итерационной формулы Герона*. Все операции в алгоритме проводятся только в целочисленном формате.

Историческая справка

Герон Александрийский — греческий математик и механик. Предположительно, жил во второй половине I в. н. э. Подробности жизни Герона неизвестны. Его не без оснований считают одним из величайших инженеров за всю историю человечества. Например, он первым изобрел паровую турбину, первым начал создавать программируемые устройства.

Алгоритм 1.4. Вычисление квадратного корня в \mathbb{Z}

```

h ← a
while (1) do
    β ← (h + a/h) / 2
    if β < h then
        h ← β
    else
        return h
    end if
end while
// h = ⌊√a⌋

```

Замечание 1.5

При программной реализации алгоритма 1.4 операция деления на двойку реализуется как сдвиг числа на один бит.

Проведем обоснование корректности работы алгоритма 1.4. Справедлива следующая теорема.

Теорема 1.13. Число α , построенное по алгоритму 1.4, удовлетворяет неравенству (1.10).

Доказательство. Заметим, что для любого $h > 0$ справедливо:

$$h + \frac{a}{h} \geq 2\sqrt{a}. \quad (1.14)$$

Действительно, из очевидного неравенства $(h^2 - a)^2 \geq 0$ следует, что

$$(h^2 + a)^2 \geq 4h^2a, \quad \text{или} \quad h^2 + a \geq 2h\sqrt{a}.$$

Последнее неравенство равносильно (1.14).

Таким образом, из (1.14) следует, что последовательность вычисляемых в алгоритме 1.4 значений h убывает и при этом ограничена снизу: $h \geq \sqrt{a}$.

Покажем, что алгоритм 1.4 остановится только при выполнении условия $h = \lfloor \sqrt{a} \rfloor$.

Предположим противное, $h > \lfloor \sqrt{a} \rfloor$, тогда $h^2 > a$. Согласно условию шага 3 алгоритма 1.4 выполнено неравенство $\beta \geq h$. Далее имеем

$$\beta - h = \left(h + \frac{a}{h} \right) / 2 - h = \left(\frac{a}{h} - h \right) / 2 = \frac{a - h^2}{2h} < 0.$$

Полученное противоречие завершает доказательство. ■

Пример 1.5. Рассмотрим работу алгоритма 1.4 для нахождения начального значения $\lfloor \sqrt{a} \rfloor$ из примера 1.3. Протокол работы алгоритма представлен в таблице 1.4.

Таблица 1.4.

	h	β		h	β
1	2520	$(2520 + 1)/2 = 1260$	5	162	$(162 + 15)/2 = 88$
2	1260	$(1260 + 2)/2 = 631$	6	88	$(88 + 28)/2 = 58$
3	631	$(631 + 3)/2 = 317$	7	58	$(58 + 43)/2 = 50$
4	317	$(317 + 7)/2 = 162$	8	50	$(50 + 50)/2 = 50$

Протокол работы алгоритма 1.4 для примера 1.4 показан в таблице 1.5.

Таблица 1.5.

	h	β		h	β
1	221	$(221 + 1)/2 = 111$	5	18	$(18 + 12)/2 = 15$
2	111	$(111 + 1)/2 = 56$	6	15	$(15 + 14)/2 = 14$
3	56	$(56 + 3)/2 = 29$	7	14	$(14 + 15)/2 = 14$
4	29	$(29 + 7)/2 = 18$			

1.1.5. Позиционная запись натуральных чисел

Определение 1.7. Упорядоченный набор неотрицательных целых чисел $(a_n \dots a_0)_p$ называют p -ичной записью натурального числа s (представлением числа s в p -ичной системе счисления или просто p -ичным числом),

если

$$c = p^n a_n + p^{n-1} a_{n-1} + \dots + p a_1 + a_0, \quad (1.15)$$

где p — натуральное число, большее 1, $0 \leq a_k < p$ и $a_n \neq 0$.

Числа a_k в p -ичной записи называют цифрами и обычно обозначают отдельными символами, например $10 = A, 11 = B$ и т. д.

Историческая справка

Изобретение позиционной нумерации, приписывается шумерам и вавилонянам (шестидесятеричная система счисления), затем такая нумерация продолжила свое развитие в Индии. Здесь приблизительно в V в. н. э. возникла десятичная система счисления. Затем началось ее быстрое распространение из Индии на Запад и Восток. В IX в. появляются рукописи на арабском языке, в которых излагается эта система счисления. Однако только к XVI в. новая нумерация получила широкое распространение в науке. В России она начинает распространяться при Петре I и в начале XVIII в. вытесняет славянскую алфавитную систему счисления, которая сохранилась до наших дней в богослужебных книгах.

В алфавитных системах для записи чисел использовался буквенный алфавит. В славянской системе роль цифр играли не все буквы, а только те, которые имеются в греческом алфавите. Над буквой, обозначающей цифру, ставился специальный знак — титло.

Теорема 1.14. Каждое натуральное число имеет единственную p -ичную запись.

Доказательство. Пусть c имеет две различные p -ичные записи: $a_n a_{n-1} \dots a_1 a_0$ и $b_m b_{m-1} \dots b_1 b_0$. Тогда

$$\begin{aligned} c &= p^n a_n + \dots + p a_1 + a_0 = p(p^{n-1} a_n + \dots + a_1) + a_0 = p c_1 + a_0 = \\ &= p^m b_m + \dots + p b_1 + b_0 = p(p^{m-1} b_m + \dots + b_1) + b_0 = p c_2 + b_0. \end{aligned}$$

Так как частное и остаток при делении на p определяются однозначно,

$$a_0 = b_0 \text{ и } c_1 = p^{n-1} a_n + \dots + a_1 = p^{m-1} b_m + \dots + b_1 = c_2.$$

Применяя аналогичные рассуждения к c_1 и c_2 , получим $a_1 = b_1$ и т. д. ■

Замечание 1.6

Существуют и другие способы позиционной записи натуральных чисел, т. е. представление их упорядоченными наборами цифр.

Пример 1.6 (факториальная запись).

$$c = (a_n a_{n-1} \cdots a_1)! \Leftrightarrow c = a_n n! + a_{n-1} (n-1)! + \cdots + a_1 \cdot 1!,$$

где $0 \leq a_k \leq k$, $a_n \neq 0$.

Упражнение 1.4. Докажите единственность факториальной записи натуральных чисел.

Замечание 1.7

Алгоритм построения факториальной записи числа будет получен далее (см. раздел 1.6.8, алгоритм 1.35).

1.1.6. Алгоритмы арифметических действий с p -ичными записями натуральных чисел

Алгоритмы сложения, вычитания, умножения «столбиком» и деления «уголком» для p -ичных записей чисел совпадают с известными алгоритмами для десятичных записей, если заменить таблицы сложения (вычитания) и умножения (деления). В этом случае говорят о выполнении операций в p -ичной арифметике.

Для рассматриваемых далее алгоритмов введем следующие обозначения:

$$a = (a_n \cdots a_0)_p, b = (b_m \cdots b_0)_p \text{ — записи исходных чисел,}$$

$$c = (c_l \cdots c_0)_p \text{ — результат работы алгоритма.}$$

Алгоритм 1.5. Сложение p -ичных чисел

```

if  $n < m$  then
     $a \leftrightarrow b; \quad n \leftrightarrow m$  // длина числа  $a \geq$  длины числа  $b$ 
end if
 $s \leftarrow 0$  //  $s$  — величина переноса в старший разряд
for  $i \leftarrow 0, m$  do
     $c_i \leftarrow (a_i + b_i + s) \bmod p$  // вычисление очередной с конца цифры результата
     $s \leftarrow (a_i + b_i + s) \div p$  // вычисление величины межразрядного переноса
end for

// продолжение сложения числа после «прохождения» старшего разряда числа
меньшей длины
for  $i \leftarrow m + 1, n$  do
     $c_i \leftarrow (a_i + s) \bmod p$ 

```

```

     $s \leftarrow (a_i + s) \div p$ 
end for

```

// формирование старшего разряда результата, если длина результата больше длины слагаемых

```

if  $s > 0$  then
     $c_{n+1} \leftarrow s$ 
end if

```

Рассмотрим алгоритм умножения p -ичного числа на p -ичную цифру α , где $1 \leq \alpha < p$.

Алгоритм 1.6. Умножение p -ичного числа на цифру

```

 $s \leftarrow 0$ 
for  $i \leftarrow 0, n$  do
     $c_i \leftarrow (a_i \alpha + s) \bmod p$ 
     $s \leftarrow (a_i \alpha + s) \div p$ 
end for
if  $s > 0$  then
     $m \leftarrow n + 1$ 
     $c_m \leftarrow s$ 
else
     $m \leftarrow n$ 
end if

```

Рассмотрим алгоритм умножения p -ичного числа на число p^k (сдвиг числа на k разрядов влево). Заметим, что число разрядов сдвинутого числа увеличится на k .

Алгоритм 1.7. Сдвиг p -ичного числа на k разрядов влево

```

for  $i \leftarrow n, 0$  do
     $c_{i+k} \leftarrow a_i$ 
end for
for  $i \leftarrow (k-1), 0$  do
     $c_i \leftarrow 0$ 
end for

```

Перейдем к перемножению p -ичных чисел. Алгоритм сводится к поразрядным умножениям и сдвигам.

Алгоритм 1.8. Перемножение p -ичных чисел

```

 $c \leftarrow 0$ 
for  $i \leftarrow 0, m$  do
    // поразрядное умножение со сдвигом выполняется по алгоритмам 1.6 и 1.7

```

```

    c ← c + (abi)pi
end for

```

Задание 1.1. Упростите алгоритмы арифметических операций для двоичных чисел.

Задание 1.2. Сформулируйте алгоритмы сложения и умножения натуральных чисел в факториальной записи.

Задание 1.3. Напишите алгоритмы вычитания и деления нацело, основываясь на таблицах вычитания и деления p -ичных чисел.

Пример 1.7. Используя рассмотренные алгоритмы 1.5–1.8, построим таблицы сложения (таблица 1.6a) и умножения (таблица 1.6b) для 5-ичных чисел.

Таблица 1.6.

$(+)_5$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	10
2	2	3	4	10	11
3	3	4	10	11	12
4	4	10	11	12	13

(a)

$(\times)_5$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	11	13
3	0	3	11	14	22
4	0	4	13	22	31

(b)

1.1.7. Алгоритмы перевода p -ичной записи натурального числа в q -ичную

Первый из алгоритмов перевода использует p -ичную арифметику, второй — q -ичную. В основе первого алгоритма лежит та же идея, что и в доказательстве единственности p -ичной записи натурального числа.

Алгоритм 1.9. Перевод из p -ичной записи в q -ичную в p -ичной арифметике

```

i ← 0
while a ≠ 0 do
    bi ← a mod q           // деление a на q выполняется в p-ичной арифметике
    a ← a ÷ q
    i ← i + 1
end while
m ← i - 1

```

Второй алгоритм основан на так называемой схеме Горнера [см. (1.76)]:

$$\begin{aligned} a &= (a_n a_{n-1} \dots a_0)_p = a_n p^n + a_{n-1} p^{n-1} + \dots + a_1 p + a_0 = \\ &= ((\dots ((a_n p + a_{n-1}) p + a_{n-2}) p + \dots + a_1) p + a_0). \end{aligned} \quad (1.16)$$

Алгоритм 1.10. Перевод из p -ичной записи в q -ичную в q -ичной арифметике

```

b ← 0
for  $i \leftarrow n, 0$  do
     $b \leftarrow bp + a_i$            // действия выполняются в  $q$ -ичной арифметике
end for

```

Замечание 1.8

При $p > q$ применение последнего алгоритма подразумевает предварительный перевод всех цифр в q -ичную запись. Аналогично при $p < q$ алгоритм 1.9 подразумевает дополнительные действия по переводу цифр порождаемой записи из p -ичной записи в q -ичную.

1.1.8. Алгоритм эффективного возведения числа в натуральную степень

Схема Горнера (1.16) может быть применена для построения эффективного алгоритма возведения числа a в натуральную степень m . Рассмотрим двоичную запись числа m :

$$\begin{aligned} m &= (b_n \dots b_0)_2 = 2^n b_n + \dots + 2b_1 + b_0 = \\ &= ((\dots (b_n 2 + b_{n-1}) 2 + b_{n-2}) 2 + \dots + b_1) 2 + b_0, \end{aligned} \quad (1.17)$$

где $b_i \in \{0, 1\}$, $b_n \neq 0$.

Тогда

$$a^m = a^{2m_1 + b_0} = (a^{m_1})^2 a^{b_0} = \begin{cases} (a^{m_1})^2 & \text{при } b_0 = 0, \\ (a^{m_1})^2 a & \text{при } b_0 = 1, \end{cases} \text{ где}$$

$$m_1 = (b_n \dots b_1)_2 = 2^{n-1} b_n + \dots + 2b_2 + b_1.$$

Алгоритм 1.11. Возведение числа в натуральную степень

```

c ← 1                               // Результат  $c = a^m$ 
for  $i \leftarrow n, 0$  do
     $c \leftarrow c \cdot c$ 

```



```

    if  $b_i > 0$  then
         $c \leftarrow c \cdot a$ 
    end if
end for

```

Замечание 1.9

Алгоритм 1.11 легко модифицируется для возведения a в степень m в кольце \mathbb{Z}_k . Для этого достаточно операции умножения осуществлять в кольце вычетов по модулю k (см. теорему 1.28).

Если двоичное представление степени (1.17) заранее неизвестно, алгоритм 1.11 можно модифицировать, встроив в него побитовую обработку числа m .

Алгоритм 1.12. Модификация алгоритма 1.11

```

 $c \leftarrow 1$  // Результат  $c = a^m$ 
 $x \leftarrow a$ 
 $y \leftarrow m$ 
while  $y \neq 0$  do
    while  $y \div 2 = 0$  do //  $y$  — четное число
         $x \leftarrow x \cdot x$ 
         $y \leftarrow y/2$ 
    end while
     $c \leftarrow c \cdot x$ 
     $y \leftarrow y - 1$ 
end while

```

Пример 1.8. Вычислим $27^{125}(391)$. Согласно алгоритму 1.11, имеем (см. таблицу 1.7).

$$m = 125 = (1111101)_2, \quad a = 27, \quad c = 1, \quad i = 6, 5, \dots, 0.$$

Таким образом, $27^{125}(391) = 164$.

Таблица 1.7.

Значение i	Значение c
6	$c = c^2 \cdot 27(391) = 27$
5	$c = c^2 \cdot 27 = 27^3(391) = 19\,683(391) = 133$
4	$c = c^2 \cdot 27 = 133^2 \cdot 27(391) = 47\,7603(391) = 192$
3	$c = c^2 \cdot 27 = 192^2 \cdot 27(391) = 995\,328(391) = 233$

Продолжение таблицы 1.7

Значение i	Значение c
2	$c = c^2 \cdot 27 = 233^2 \cdot 27(391) = 1\,465\,803(391) = 335$
1	$c = c^2 = 335^2(391) = 112\,225(391) = 8$
0	$c = c^2 \cdot 27 = 8^2 \cdot 27(391) = 1\,728(391) = 164$

Задачи и вопросы

1. Как провести факторизацию чисел при наличии «бесконечно быстрого компьютера»?
2. Как провести факторизацию «без делений»?
3. Натуральное число представлено в каноническом представлении в виде произведения степеней своих простых делителей

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$$

Сколько всего делителей у числа n ?

4. Разложите на простые множители 100!
5. Последовательность натуральных чисел a_0, a_1, \dots строится следующим образом: $a_0 \neq a_1$, $D(a_0, a_1) > 1$, а каждое следующее число a_k есть остаток от деления a_{k-2} на a_{k-1} . Существует ли $a_i = 1$?
6. Пусть $c = D(a, b)$, $d = D(a, c^2)$, где a и b — различные не взаимно простые натуральные числа. Справедливо ли неравенство $d \geq c$?
7. Переведите число 2021 из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную системы счисления.
8. Переведите двоичное число 1001101 в десятичную систему счисления.
9. Вычислите значение суммы в десятичной системе счисления:

$$32_4 + 111_2 + 75_8.$$

10. Сколько операций умножения выполнит алгоритм 1.11 при вычислении:

а) a^{2^k} ;

б) a^{2^k-1} .

11. Докажите, что количество умножений в алгоритме 1.11 возведения числа в степень m не превышает $2 \lceil \log_2 m \rceil$.

12. Найдите формулу, выражающую число умножений в алгоритме 1.11 возведения a в степень m через l — число цифр в двоичной записи m и r — число нулей в двоичной записи m .

end while

Пример 1.9 (классический алгоритм Евклида). Найдем наибольший общий делитель 4704 и 2808. Применим алгоритм 1.13. Протокол работы алгоритма запишем в виде таблицы. 1.8.

Таблица 1.8.

Итерация	a	b	q	r
1	4704	2808	1	1896
2	2808	1896	1	912
3	1896	912	2	72
4	912	72	12	48
5	72	48	1	24
6	48	24	2	0
7	24	0		

Таким образом, $D(4704, 2808) = 24$.




Историческая справка

Древнегреческие математики называли этот алгоритм взаимным вычитанием. Этот алгоритм не был открыт Евклидом, так как упоминание о нем имеется уже у Аристотеля. Самим Евклидом алгоритм был описан в геометрическом виде и реализуется с помощью циркуля и линейки.

Пусть даны два отрезка длиной a и b . Вычтем из большего отрезка меньший и заменим больший отрезок полученной разностью. Повторяем эту операцию, пока отрезки не станут равны. Если это произойдет, то исходные отрезки соизмеримы, а последний полученный отрезок есть их наибольшая общая мера. Если общей меры нет, то процесс бесконечен.

1.2.2. Бинарный алгоритм Евклида


Классический алгоритм Евклида не всегда является наилучшим способом для нахождения наибольшего общего делителя. Сравнительно недавно для решения этой задачи был предложен совсем иной алгоритм, не требующий операции деления. Основанный исключительно на операциях вычитания, он проверяет, является ли число четным или нет, и сдвигает вправо двоичное представление четного числа (деление пополам). Бинарный алгоритм нахождения $D(a, b)$ основан на нескольких простых свойствах натуральных чисел.

 **Замечание 1.10**

Обсуждаемый алгоритм был известен еще в I в. в Китае, но опубликован лишь в 1967 г. израильским физиком и программистом Джозефом Стейном. Ввиду этого встречается альтернативное название метода — алгоритм Стейна.

Лемма 1.3. Пусть $a, b \in N$. Тогда справедливы следующие соотношения:

- 1) a и b четны, тогда $D(a, b) = 2D(a/2, b/2)$;
- 2) a четно и b нечетно, тогда $D(a, b) = D(a/2, b)$;
- 3) $D(a, b) = D(a - b, b)$;
- 4) Если a и b нечетны, то $a - b$ четно;
- 5) $|a - b| < \max\{a, b\}$.

 **Замечание 1.11**

Свойство 3 леммы 1.3 использовалось в схеме (1.18)–(1.19).

Алгоритм 1.14. Бинарный алгоритм Евклида

```

k ← 0           // счетчик для степени двоек, выделяемых по свойству 1 леммы 1.3
while a и b — четны do           // «выдавливает» из a, b общую степень двойки
    k ← k + 1; a ← a/2; b ← b/2
end while
while a ≠ b do                   // если a = b, то D(a, b) = a · 2k
    while a — четно do
        a ← a/2                   // свойство 2 леммы 1.3
    end while
    while b — четно do
        b ← b/2                   // свойство 2 леммы 1.3
    end while
    max(a, b) ← |a - b|/2         // свойства 3, 5 леммы 1.3
end while
D(a, b) = a · 2k

```

Пример 1.10 (бинарный алгоритм Евклида). Рассмотрим работу алгоритма с данными из примера 1.9. Протокол его работы представлен в таблице 1.9.

Имеем: $D(4704, 2808) = 2^3 \times 3 = 24$.

Таблица 1.9.

k	a	b
$0 \rightarrow 1 \rightarrow 2 \rightarrow 3$	$4704 \rightarrow 2352 \rightarrow 1176 \rightarrow 588$	$2808 \rightarrow 1404 \rightarrow 702 \rightarrow 351$
3	$588 \rightarrow 294 \rightarrow 147$	351
3	147	$204 \rightarrow 102 \rightarrow 51$
3	$96 \rightarrow 48 \rightarrow 24 \rightarrow 12 \rightarrow 6 \rightarrow 3$	51
3	3	$48 \rightarrow 24 \rightarrow 12 \rightarrow 6 \rightarrow 3$

Замечание 1.12

Для решения задачи бинарным алгоритмом потребовалось 3 вычитания и 16 делений на 2 (сдвигов мантиссы на 1 разряд вправо). Для решения этой же задачи классическим алгоритмом Евклида требуется 6 делений с остатком.

При использовании классического алгоритма Евклида получается дополнительная информация (последовательность неполных частных q_0, q_1, \dots), которая будет необходима для решения других задач.

1.2.3. Линейное представление наибольшего общего делителя

Действуя в схеме (1.19) обратным ходом снизу вверх, получаем представление $D(a, b)$ через a, b :

$$\begin{aligned}
 r_k &= r_{k-2} - r_{k-1}q_k = r_{k-2} - (r_{k-3} - r_{k-2}q_{k-1})q_k = \\
 &= -r_{k-3}q_k + r_{k-2}(1 + q_kq_{k-1}) = \\
 &= -r_{k-3}q_k + (r_{k-4} - r_{k-3}q_{k-2})(1 + q_kq_{k-1}) = \\
 &= r_{k-4}(1 + q_kq_{k-1}) + r_{k-3}(-q_k - q_{k-2}(1 + q_kq_{k-1})) = \dots = ax + by.
 \end{aligned}$$

Таким образом, справедливо следующее утверждение.

Утверждение 1.1. Для любых $a, b \in \mathbb{Z}$ уравнение $ax + by = D(a, b)$ всегда имеет решение.

Нетрудно получить рекуррентные формулы для вычисления x, y . Введем последовательность $\{y_i\}$ следующим образом:

$$y_0 = 0, \quad y_1 = 1, \quad y_{i+1} = y_{i-1} - q_{k+1-i}y_i, \quad i = 1, 2, \dots, k+1. \quad (1.20)$$

Тогда $D(a, b) = ay_{k+1} + by_{k+2}$.

Пример 1.11. Найдем линейное представление для наибольшего общего делителя из примера 1.9 (таблица 1.10).

Таким образом, $4704 \times 40 - 2808 \times 67 = 24$.

Таблица 1.10.

i	0	1	2	3	4	5	6
y_i	0	1	-1	13	-27	40	-67
q_{6-i}			1	12	2	1	1

1.2.4. Обобщенный алгоритм Евклида

Из результатов предыдущего подраздела следует, что справедливо представление

$$d = D(a, b) = ax + by. \quad (1.21)$$

Представление наибольшего общего делителя в виде (1.21) называется соотношением Безу, а числа x и y — коэффициентами Безу.

Историческая справка

Впервые данный факт опубликовал в 1624 г. французский математик Клод Гаспар Баше де Мезириак для случая взаимно простых чисел a и b . Этьен Безу в конце XVIII в. обобщил результат, распространив его и на кольцо многочленов.

Равенство (1.21) можно записать в векторной форме, т. е. наибольший общий делитель можно представить вектором (упорядоченным) набором коэффициентов разложения по исходным числам $d = (x, y)$. Легко видеть, что сами числа представляются такими векторами очень просто:

$$a = (1, 0), \text{ так как } a = 1 \times a + 0 \times b,$$

$$b = (0, 1), \text{ так как } b = 0 \times a + 1 \times b.$$

Для нахождения вектора (x, y) применим схему (1.19) классического алгоритма Евклида, но параллельно будем выполнять соответствующие действия с векторами разложений. Напомним, что арифметические действия с упорядоченными наборами чисел, так же как с векторами в координатах, выполняются покомпонентно.

Таким образом, одновременно решаются две задачи: вычисление наибольшего общего делителя и нахождение его линейного представления.

Обозначим векторы разложения для чисел a и b как (x_a, y_a) и (x_b, y_b) . Покажем работу алгоритма на примере.

Пример 1.12. Найдем $D(64, 81)$ и его линейное представление.

Имеем, $a = 64$, $b = 81$. Тогда $(x_a, y_a) = (1, 0)$, а $(x_b, y_b) = (0, 1)$. Работа алгоритма показана в таблице 1.11.

Так как в алгоритме Евклида получена единица (следующий шаг последний — второе число будет равно 0), приходим к искомому разложению,

где $x = 19$, $y = -15$:

$$64 \times 19 + 81 \times (-15) = 1 = D(64, 81).$$

Таблица 1.11.

a	(x_a, y_a)	b	(x_b, y_b)	Комментарий
64	(1, 0)	81	(0, 1)	
64	(1, 0)	17	(-1, 1)	$b = b - a$, $(x_b, y_b) = (x_b, y_b) - (x_a, y_a)$
13	(4, -3)	17	(-1, 1)	$a = a - 3b$, $(x_a, y_a) = (x_a, y_a) - 3(x_b, y_b)$
13	(4, -3)	4	(-5, 4)	$b = b - a$, $(x_b, y_b) = (x_b, y_b) - (x_a, y_a)$
1	(19, -15)	4	(-5, 4)	$a = a - 3b$, $(x_a, y_a) = (x_a, y_a) - 3(x_b, y_b)$

Сформулируем алгоритм.

Алгоритм 1.15. Обобщенный алгоритм Евклида

Инициализация:

$$\begin{aligned} (x_a, y_a) &\leftarrow (1, 0) \\ (x_b, y_b) &\leftarrow (0, 1) \end{aligned}$$

while $b \neq 0$ **do**

$$\begin{aligned} t &\leftarrow a, T \leftarrow (x_a, y_a) \\ a &\leftarrow b, (x_a, y_a) \leftarrow (x_b, y_b) \\ q &\leftarrow t \div b \\ b &\leftarrow t - qb, (x_b, y_b) \leftarrow T - q(x_b, y_b) \end{aligned}$$

end while

Наибольший общий делитель будет находиться в a , а коэффициенты разложения в (x_a, y_a) .

 **Замечание 1.13**

Если $a < b$, то после первого шага алгоритма a и b поменяются местами. Если оба числа равны нулю, алгоритм неприменим. Переменные t и T используются для временного хранения a и вектора разложения (x_a, y_a) соответственно.

Пример 1.13. Вернемся к примеру 1.12. Протокол работы расширенного алгоритма Евклида удобно записать в виде, показанном в таблице 1.12.

Пример получения новых значений первых компонент наборов (x_a, y_a) и (x_b, y_b) показан в третьей строке таблицы (клетки выделены серым цветом): из числа, расположенного в первой клетке, вычитают число во второй клетке, помноженное на число, стоящее справа от него во второй строке (число в рамке), результат записывают в третью клетку.

Таблица 1.12.

Остатки	64	81	64	17	13	4	1	0
Частные q			0	1	3	1	3	4
x	1	0	1	-1	4	-5	19	-81
y	0	1	0	1	-3	4	-15	64

Аналогично выполняют операции для нахождения вторых компонент (четвертая строка, клетки выделены серым цветом). Легко проследить аналогию с рекуррентными формулами (1.20) предыдущего подраздела.

Заметим, что для каждого столбика выполняется равенство

$$ax + by = \text{остаток.}$$

Пример 1.14. Запишем протокол работы расширенного алгоритма Евклида для примера 1.9 (см. таблицу 1.13).

Таблица 1.13.

	4704	2808	1896	912	72	48	24	0
q			1	1	2	12	1	2
x	1	0	1	-1	3	-37	40	117
y	0	1	-1	2	-5	62	-67	196

Таким образом,

$$D(4704, 2808) = 24 = 4704 \times 40 + 2808 \times (-67).$$

1.2.5. Разложение числа в цепную дробь

Историческая справка

Цепные дроби были введены в 1572 г. итальянским математиком Рафаэлем Бомбелли. Современное обозначение непрерывных дробей впервые встречается у итальянского математика Пьетро Антонио Каталди в 1613 г. Систематически первым изложил теорию цепных дробей Леонард Эйлер (раздел 1.3.3). Он же дал несколько важных примеров использования и обобщения цепных дробей.

Запишем схему (1.19) в виде

$$\left\{ \begin{array}{l} \frac{a}{b} = q_0 + \frac{r_0}{b}, \\ \frac{b}{r_0} = q_1 + \frac{r_1}{r_0}, \\ \vdots \\ \frac{r_{k-2}}{r_{k-1}} = q_k + \frac{r_k}{r_{k-1}}, \\ \frac{r_{k-1}}{r_k} = q_{k+1}. \end{array} \right. \quad (1.22)$$

Из (1.22) получаем

$$\frac{a}{b} = q_0 + \frac{1}{\frac{b}{r_0}} = q_0 + \frac{1}{q_1 + \frac{1}{\frac{r_0}{r_1}}} = \dots = q_0 + \frac{1}{q_1 + \frac{1}{q_2 + \frac{1}{\dots + \frac{1}{q_{k+1}}}}}. \quad (1.23)$$

Определение 1.8. Выражение (1.23) называют *цепной, или непрерывной, дробью*. Дроби, возникающие при разложении вида (1.23), называют *подходящими дробями*. Последняя подходящая дробь равна a/b . Числа $q_0, q_1, q_2, \dots, q_{k+1}$ называют *звеньями дроби*. Более коротко (1.23) записывают в виде

$$\frac{a}{b} = [q_0; q_1, q_2, \dots, q_{k+1}]. \quad (1.24)$$

В представлении (1.24) q_0 есть целая часть дроби a/b . Легко показать следующее:

- 1) если $0 < a < b$, т. е. a/b — правильная положительная дробь, то представление (1.24) для нее имеет вид $[0; q_1, q_2, \dots, q_{k+1}]$;
- 2) если дробь $a/b < 0$, то $a/b = [q_0; q_1, q_2, \dots, q_{k+1}]$, где $q_0 \leq -1$ есть целая часть дроби a/b ;
- 3) любое $a \in \mathbb{Z}$ является цепной дробью с одним звеном $[a]$.

Пример 1.15. Пусть $a = -48$, $b = 109$. Тогда

$$\frac{a}{b} = -\frac{48}{109} = -1 + \frac{61}{109} = -1 + \frac{1}{\frac{109}{61}}$$

Таблица 1.14.

Итерация	a	b	q	r	Итерация	a	b	q	r
1	109	61	1	48	5	9	4	2	1
2	61	48	1	13	6	4	1	4	0
3	48	13	3	9	7	1	0		
4	13	9	1	4					

Для $a = 109 > b = 61 > 0$ применим алгоритм 1.13 (см. таблицу 1.14).

Тогда окончательно

$$-\frac{48}{109} = [-1; 1, 1, 3, 1, 2, 4].$$

Таким образом, справедлива следующая теорема.

Теорема 1.15. Для любых $a, b \in \mathbb{Z}, b \neq 0$ дробь a/b можно одним и только одним способом представить в виде (1.24), где все звенья q_i , начиная с q_1 , положительны и последнее звено $q_{k+1} > 1$.

Условие $q_{k+1} > 1$ в теореме обеспечивает единственность разложения (1.24), так как

$$[q_0; q_1, \dots, q_{k+1}] = [q_0; q_1, \dots, q_{k+1} - 1, 1].$$

Действительно, последнее уравнение в (1.19) $r_{k-1} = r_k q_{k+1}$ можно записать в виде

$$\begin{cases} r_{k-1} &= r_k(q_{k+1} - 1) + r_k, \\ r_k &= r_k \cdot 1. \end{cases}$$

Пример 1.16. Рассмотрим дробь из примера 1.9. Используя результаты примера 1.9, получаем представление (1.24):

$$\frac{4704}{2808} = [1; 1, 2, 12, 1, 2].$$

Заметим, что если полученную цепную дробь снова привести к обычной,

то получится *несократимая* дробь: $\frac{4704}{2808} = \frac{196}{117}$.

1.2.6. Свойства подходящих дробей и их вычисление

Введем обозначения для подходящих дробей, определенных в предыдущем подразделе при разложении (1.23):

$$\delta_0 = q_0, \quad \delta_1 = q_0 + \frac{1}{q_1}, \quad \delta_2 = q_0 + \frac{1}{q_1 + \frac{1}{q_2}} \quad \dots \quad (1.25)$$

Обозначим числитель и знаменатель δ_k через P_k и Q_k соответственно. Получим рекуррентные формулы для определения последовательностей числителей $\{P_k\}$ и знаменателей $\{Q_k\}$.

Положив $P_{-1} = 1, Q_{-1} = 0$, имеем

$$\begin{aligned} \delta_0 &= \frac{q_0}{1} = \frac{P_0}{Q_0} \text{ (полагаем } P_0 = q_0, Q_0 = 1), \\ \delta_1 &= \frac{q_0 + \frac{1}{q_1}}{1} = \frac{q_0 q_1 + 1}{q_1 + 0} = \frac{q_1 P_0 + P_{-1}}{q_1 Q_0 + Q_{-1}} = \frac{P_1}{Q_1}, \quad \delta_2 = \frac{q_2 P_1 + P_0}{q_2 Q_1 + Q_0} = \frac{P_2}{Q_2}, \dots \end{aligned}$$

При анализе числителей и знаменателей последовательности $\delta_0, \delta_1, \dots$ становятся очевидными рекуррентные соотношения для P_k и Q_k :

$$\begin{cases} P_s = q_s P_{s-1} + P_{s-2}, \\ Q_s = q_s Q_{s-1} + Q_{s-2}. \end{cases} \quad (1.26)$$

Пример 1.17. Используя результаты примера 1.9, получим последовательность подходящих дробей (таблица 1.15).

Таблица 1.15.

s	-1	0	1	2	3	4	5
q_s		1	1	2	12	1	2
P_s	1	1	2	5	62	67	196
Q_s	0	1	1	3	37	40	117

Таким образом,

$$\delta_0 = 1, \quad \delta_1 = 2, \quad \delta_2 = \frac{5}{3}, \quad \delta_3 = \frac{62}{37}, \quad \delta_4 = \frac{67}{40}, \quad \delta_5 = \frac{196}{11} = \frac{4704}{2808}.$$

Установим важное свойство подходящих дробей. Рассмотрим разность двух соседних подходящих дробей. Имеем

$$\begin{aligned}\delta_s - \delta_{s-1} &= \frac{P_s}{Q_s} - \frac{P_{s-1}}{Q_{s-1}} = \frac{P_s Q_{s-1} - P_{s-1} Q_s}{Q_s Q_{s-1}} = \frac{h_s}{Q_s Q_{s-1}}, \text{ где} \\ h_s &= P_s Q_{s-1} - P_{s-1} Q_s = (q_s P_{s-1} + P_{s-2}) Q_{s-1} - (q_s Q_{s-1} + Q_{s-2}) P_{s-1} = \\ &= -(P_{s-1} Q_{s-2} - P_{s-2} Q_{s-1}) = -h_{s-1}.\end{aligned}$$

Таким образом, $h_s = -h_{s-1}$, но $h_0 = P_0 Q_{-1} - P_{-1} Q_0 = -1$, следовательно, $h_s = (-1)^{s+1}$ и доказана следующая лемма.

Лемма 1.4. Справедливо равенство

$$\frac{P_s}{Q_s} - \frac{P_{s-1}}{Q_{s-1}} = \frac{(-1)^{s+1}}{Q_s Q_{s-1}}. \quad (1.27)$$

Следствие 1.3. Подходящие дроби несократимы.

Доказательство. Так как $P_s Q_{s-1} - P_{s-1} Q_s = (-1)^{s+1}$, то любой общий делитель P_s, Q_s равен 1. Таким образом, $D(P_s, Q_s) = 1$. ■

Следствие 1.4.

$$\lim_{s \rightarrow \infty} \left| \frac{P_s}{Q_s} - \frac{P_{s-1}}{Q_{s-1}} \right| = 0. \quad (1.28)$$

Доказательство. Согласно теореме 1.15 все звенья цепной дроби q_i положительны (кроме, быть может, q_0). Тогда из (1.26) следует, что все Q_s (при $s \geq 0$) также положительны и последовательность натуральных чисел Q_s возрастает. Из (1.27) получаем требуемое. ■

1.2.7. Цепные дроби и анализ алгоритма Евклида

Введенные понятия цепной дроби и подходящих дробей оказываются очень полезными для анализа работы алгоритма Евклида. Дадим необходимые обозначения. Рассмотрим трехдиагональный определитель:

$$\begin{vmatrix} q_0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & q_1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & q_2 & 1 & \cdots & 0 & 0 \\ & & \dots & \dots & \dots & & \\ 0 & 0 & 0 & 0 & \cdots & q_{n-2} & 1 \\ 0 & 0 & 0 & 0 & \cdots & -1 & q_{n-1} \end{vmatrix} = K_n(q_0, q_1, \dots, q_{n-1}). \quad (1.29)$$

Определение 1.9. Определитель (1.29) называют *континуантой* n -го порядка или индекса.

Разложим континуанту n -го порядка по последнему столбцу:

$$K_n(q_0, q_1, \dots, q_{n-1}) = q_{n-1}K_{n-1}(q_0, q_1, \dots, q_{n-2}) + K_{n-2}(q_0, q_1, \dots, q_{n-3}). \quad (1.30)$$

Соотношение (1.30) очень напоминает рекуррентные соотношения (1.26) для числителей и знаменателей подходящих дробей. Это не случайно, и две следующие леммы подтверждают предположение о связи континуант и цепных дробей.

Лемма 1.5. Континуанта $K_n(q_0, q_1, \dots, q_{n-1})$ равна сумме всевозможных произведений элементов q_0, q_1, \dots, q_{n-1} , одно из которых содержит все эти элементы, а другие получаются из него выбрасыванием одной или нескольких пар сомножителей с соседними номерами (если выброшены все сомножители, то считаем, что осталась 1).

Доказательство. Индукция по n . База индукции:

$$K_1(q_0) = q_0, \quad K_2(q_0, q_1) = \begin{vmatrix} q_0 & 1 \\ -1 & q_1 \end{vmatrix} = q_0q_1 + 1,$$

и утверждение леммы справедливо для континуант первого и второго порядков.

Шаг индукции. Пусть утверждение леммы справедливо для континуант $(n-2)$ -го и $(n-1)$ -го порядков. Применяв разложение (1.30), получим требуемое. ■

Пример 1.18.

$$\begin{aligned} K_6(q_0, q_1, q_2, q_3, q_4, q_5) &= q_0q_1q_2q_3q_4q_5 + q_2q_3q_4q_5 + q_0q_3q_4q_5 + q_0q_1q_4q_5 + \\ &+ q_0q_1q_2q_5 + q_0q_1q_2q_3 + q_4q_5 + q_2q_5 + q_0q_5 + \\ &+ q_2q_3 + q_0q_3 + q_0q_1 + 1. \end{aligned}$$

 **Замечание 1.14**

Количество слагаемых в континуанте n -го порядка есть сумма числа слагаемых в континуантах $(n-1)$ -го и $(n-2)$ -го порядков, т. е. континуанта $(q_0 q_1 \dots q_{n-1})$ содержит ϕ_{n+1} слагаемых, где ϕ_{n+1} — $(n+1)$ -е число Фибоначчи (см. раздел 1.2.12). Считаем, что континуанта нулевого порядка K_0 содержит 0 слагаемых.

Явная связь континуант и цепных дробей впервые была установлена Эйлером.

Лемма 1.6 (Эйлер). Справедливо тождество

$$[q_0; q_1, \dots, q_{n-1}] = \frac{K_n(q_0, q_1, \dots, q_{n-1})}{K_{n-1}(q_1, q_2, \dots, q_{n-1})}. \quad (1.31)$$

Доказательство. Индукция по n . База индукции:

$$[q_0; q_1] = q_0 + \frac{1}{q_1} = \frac{q_0 q_1 + 1}{q_1} = \frac{K_2(q_0, q_1)}{K_1(q_1)}.$$

Шаг индукции. Пусть тождество (1.31) верно для дробей с $n-1$ звеном включительно. Представим n -звенную дробь $(q_0, q_1, \dots, q_{n-1})$ дробью с $n-1$ звеном, где последнее звено имеет вид $q_{n-2} + \frac{1}{q_{n-1}}$. Применив к этой дроби индукционное предположение, с учетом разложения (1.30) имеем

$$\begin{aligned} [q_0; q_1, \dots, q_{n-1}] &= \left[q_0; q_1, \dots, q_{n-2} + \frac{1}{q_{n-1}} \right] = \\ &= \frac{K_{n-1} \left(q_0, q_1, \dots, \left(q_{n-2} + \frac{1}{q_{n-1}} \right) \right)}{K_{n-2} \left(q_1, q_2, \dots, \left(q_{n-2} + \frac{1}{q_{n-1}} \right) \right)} = \\ &= \frac{\left(q_{n-2} + \frac{1}{q_{n-1}} \right) K_{n-2}(q_0, q_1, \dots, q_{n-3}) + K_{n-3}(q_0, q_1, \dots, q_{n-4})}{\left(q_{n-2} + \frac{1}{q_{n-1}} \right) K_{n-3}(q_1, q_2, \dots, q_{n-3}) + K_{n-4}(q_1, q_2, \dots, q_{n-4})} = \\ &= \frac{K_{n-1}(q_0, q_1, \dots, q_{n-2}) + \frac{K_{n-2}(q_0, q_1, \dots, q_{n-3})}{q_{n-1}}}{K_{n-2}(q_1, q_2, \dots, q_{n-2}) + \frac{K_{n-3}(q_1, q_2, \dots, q_{n-3})}{q_{n-1}}} = \\ &= \frac{K_n(q_0, q_1, \dots, q_{n-1})}{K_{n-1}(q_1, q_2, \dots, q_{n-1})}. \end{aligned}$$

■

Замечание 1.15

Из (1.31) становится очевидно, что

$$\frac{K_n(q_0, q_1, \dots, q_{n-1})}{K_{n-1}(q_1, q_2, \dots, q_{n-1})} = \frac{P_{n-1}}{Q_{n-1}} = \delta_{n-1}.$$

Таким образом, введенное понятие континуанты — это другой взгляд на подходящие дроби, дающий возможность связать их с последовательностью Фибоначчи (см. лемму 1.5).

Перейдем к анализу алгоритма Евклида. Нас будет интересовать наихудший случай — когда алгоритм Евклида работает особенно долго. Сформулируем вопрос точнее: для каких двух наименьших чисел надо применить алгоритм Евклида, чтобы он работал в точности заданное число шагов? Ответ на этот вопрос дает следующая теорема.

Теорема 1.16 (Ламе). Пусть n — произвольное натуральное число, и $a > b > 0$ такие, что алгоритму Евклида для обработки a и b необходимо выполнить точно n шагов (делений с остатком), причем a — наименьшее натуральное число с таким свойством. Тогда

$$a = \phi_{n+2}, \quad b = \phi_{n+1},$$

где ϕ_k — k -е число Фибоначчи.

Доказательство. Разложим a/b в цепную дробь. Согласно лемме 1.6 получаем

$$\frac{a}{b} = [q_0; q_1, \dots, q_{n-1}] = \frac{K_n(q_0, q_1, \dots, q_{n-1})}{K_{n-1}(q_1, q_2, \dots, q_{n-1})},$$

где q_0, q_1, \dots, q_{n-1} — неполные частные из алгоритма Евклида. По условию теоремы их ровно n .

Принимая во внимание несократимость подходящих дробей (см. следствие 1.3 к лемме 1.4 и замечание 1.15), становится очевидным, что континуанты $(q_0 q_1 \dots q_{n-1})$ и $(q_1 q_2 \dots q_{n-1})$ взаимно просты.

Пусть $D(a, b) = d$. Тогда

$$\begin{cases} a = K_n(q_0, q_1, \dots, q_{n-1}) d, \\ b = K_{n-1}(q_1, q_2, \dots, q_{n-1}) d. \end{cases} \quad (1.32)$$

В силу единственности разложения в цепную дробь (см. теорему 1.15) в случае $a > b > 0$ справедливы неравенства $q_0, q_1, \dots, q_{n-2} \geq 1$, $q_{n-1} \geq 2$.

Очевидно, что $d \geq 1$. По лемме 1.5 континуанта есть многочлен с неотрицательными коэффициентами от переменных q_0, q_1, \dots

Его минимальное значение, очевидно, достигается при

$$q_0 = q_1 = \dots = q_{n-2} = 1, \quad q_{n-1} = 2.$$

Положив $d = 1$ и подставив эти значения q_i в (1.32), получим требуемое. ■

Историческая справка

Габриэль Ламе (1795–1870) — французский математик, механик, физик и инженер. В 1820–1831 гг. работал в России в Институте корпуса инженеров путей сообщения в Петербурге, где возглавлял первую в России кафедру прикладной механики. Он один из 72 ученых, чьи имена увековечены на Эйфелевой башне. Доказательство теоремы 1.16 было опубликовано Ламе в 1845 г.

Следствие 1.5. Если натуральные числа a и b не превосходят N , то число шагов (операций деления с остатком), необходимых алгоритму Евклида для обработки a и b , не превышает

$$\left[\log_{\alpha} (\sqrt{5} N) \right] - 2, \text{ где } \alpha = \frac{1 + \sqrt{5}}{2}.$$

Доказательство. Доказательство очевидно из примера 1.133 (см. раздел 1.7.1). ■

Коэффициент $\alpha = 1.6180337 \dots$ известен в геометрии как «золотое сечение».

Замечание 1.16

Легко подсчитать, что

$$\log_{\alpha} (\sqrt{5} N) \approx 4.785 \cdot \lg N + 1.672,$$

поэтому, например, для вычисления НОД пары чисел, меньших миллиона, алгоритму Евклида потребуется не более чем

$$4.785 \cdot 6 + 1.672 - 2 = 31 - 2 = 29$$

шагов.

Из теоремы Ламе следует, что наилучший случай для алгоритма Евклида — два последовательных числа Фибоначчи. Рекуррентное соотношение для чисел Фибоначчи (1.46), по сути, является строкой алгоритма Евклида при делении ϕ_{k+2} на ϕ_{k+1} с остатком.

В заключение подраздела, используя введенную последовательность Фибоначчи (1.46), получим еще одно свойство подходящих дробей.

Теорема 1.17. Знаменатели подходящих дробей растут не медленнее последовательности Фибоначчи:

$$Q_s \geq \phi_s \text{ при } s \geq 1,$$

где ϕ_s — s -е число Фибоначчи.

Доказательство. Наиболее медленный рост знаменателей подходящих дробей будет наблюдаться при

$$Q_s = Q_{s-1} + Q_{s-2}, \text{ т. е. при } q_1 = q_2 = \dots = q_s = 1.$$

$$\begin{aligned}
\alpha &= 5 + \frac{1}{\alpha_1}, \\
\alpha_1 &= \frac{1}{\sqrt{28} - 5} = \frac{\sqrt{28} + 5}{3} = 3 + \frac{1}{\alpha_2}, \\
\alpha_2 &= \frac{3}{\sqrt{28} - 4} = \frac{\sqrt{28} + 4}{4} = 2 + \frac{1}{\alpha_3}, \\
\alpha_3 &= \frac{4}{\sqrt{28} - 4} = \frac{\sqrt{28} + 4}{3} = 3 + \frac{1}{\alpha_4}, \\
\alpha_4 &= \frac{3}{\sqrt{28} - 5} = \sqrt{28} + 5 = 10 + \frac{1}{\alpha_5}, \\
\alpha_5 &= \frac{1}{\sqrt{28} - 5} = \frac{\sqrt{28} + 5}{3} = 3 + \frac{1}{\alpha_6}.
\end{aligned}$$

Заметим, что $\alpha_5 = \alpha_1$ и происходит возврат к первому уравнению. Процесс «зацикливается». Таким образом, получаем представление $\sqrt{28}$ в виде бесконечной периодической последовательности. Строгое обоснование этому представлению будет дано далее.

Установим некоторые свойства разложения (1.33).

Теорема 1.18. Точное значение числа $\alpha \in \mathbb{R}$ всегда находится между соседними подходящими дробями, причем оно ближе к последующей, чем к предыдущей дроби. При этом подходящие дроби с четными номерами находятся слева, а с нечетными — справа от α .

Доказательство. Считая в разложении (1.34) $\alpha_s = \frac{1}{\eta_s}$ последним знаменателем (аналогом q_s), запишем это разложение в виде

$$\alpha = [q_0; q_1, \dots, q_{s-1}, \alpha_s]. \quad (1.35)$$

Тогда α можно считать s -й подходящей дробью. Применив (1.26), получим

$$\alpha = \frac{P_{s-1}\alpha_s + P_{s-2}}{Q_{s-1}\alpha_s + Q_{s-2}}. \quad (1.36)$$

Из (1.36) имеем:

$$\begin{aligned}
\alpha Q_{s-1}\alpha_s + \alpha Q_{s-2} &= P_{s-1}\alpha_s + P_{s-2}, \\
\alpha_s(\alpha Q_{s-1} - P_{s-1}) &= P_{s-2} - \alpha Q_{s-2}, \\
\alpha_s Q_{s-1} \left(\alpha - \frac{P_{s-1}}{Q_{s-1}} \right) &= Q_{s-2} \left(\frac{P_{s-2}}{Q_{s-2}} - \alpha \right), \quad (1.37)
\end{aligned}$$

но $\alpha_s > 1$, $Q_{s-1} > Q_{s-2} > 0$, т. е. $\alpha_s Q_{s-1} > Q_{s-2}$, тогда из (1.37):

$$\left| \alpha - \frac{P_{s-1}}{Q_{s-1}} \right| < \left| \frac{P_{s-2}}{Q_{s-2}} - \alpha \right|.$$

При этом знаки у $\alpha - \delta_{s-1}$ и $\delta_{s-2} - \alpha$ одинаковы. Замечание, что

$$\delta_0 = q_0 < \alpha,$$

т. е. лежит слева от α , завершает доказательство. ■

Учитывая теорему 1.18 и свойство (1.27), получаем следующее следствие (см. рисунок 1.1).

Следствие 1.6.

$$|\alpha - \delta_s| \leq |\delta_{s+1} - \delta_s| = \frac{1}{Q_{s+1}Q_s}. \quad (1.38)$$

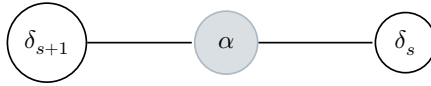


Рис. 1.1.

Замечание 1.17

Равенство в (1.38) достигается только тогда, когда α — рациональная дробь, а δ_{s+1} — последняя подходящая дробь.

Перейдем к формальному определению бесконечной цепной дроби. Разобьем последовательность подходящих дробей на две подпоследовательности с четными и нечетными номерами. Как уже отмечалось, последовательность знаменателей $\{Q_k\}$ возрастает. Применив (1.27) и теорему 1.18, имеем:

$$\begin{aligned} \frac{P_{2m-1}}{Q_{2m-1}} - \frac{P_{2m}}{Q_{2m}} &= \frac{1}{Q_{2m-1}Q_{2m}} > \frac{P_{2m+1}}{Q_{2m+1}} - \frac{P_{2m}}{Q_{2m}} = \frac{1}{Q_{2m+1}Q_{2m}} > \\ &> \frac{P_{2m+1}}{Q_{2m+1}} - \frac{P_{2m+2}}{Q_{2m+2}} = \frac{1}{Q_{2m+1}Q_{2m+2}} > 0. \end{aligned}$$

Из последней цепочки равенств получаем:

$$\frac{P_{2m-1}}{Q_{2m-1}} > \frac{P_{2m+1}}{Q_{2m+1}}, \quad \frac{P_{2m}}{Q_{2m}} < \frac{P_{2m+2}}{Q_{2m+2}}.$$

Таким образом, подпоследовательность $\left\{ \frac{P_{2m+1}}{Q_{2m+1}} \right\}$ убывает и по (1.27) ограничена снизу (например, δ_2), следовательно, имеет предел. Аналогично подпоследовательность $\left\{ \frac{P_{2m}}{Q_{2m}} \right\}$ возрастает и ограничена сверху (например, δ_1), следовательно, также имеет предел. В силу (1.28) эти пределы совпадают.

Определение 1.10. *Бесконечной цепной дробью* будем называть предел последовательности подходящих дробей.

Перейдем к вопросу вычисления бесконечной цепной дроби.

Утверждение 1.2. Бесконечную цепную дробь можно вычислить с любой степенью точности ε .

Доказательство. Используя неравенство (1.38) и возрастание последовательности $\{Q_k\}$, получим оценку

$$|\alpha - \delta_m| \leq \frac{1}{Q_{m+1}Q_m} < \frac{1}{Q_m^2} < \varepsilon. \quad (1.39)$$

Таким образом для достижения нужной точности вычисляем подходящие дроби $\{\delta_m\}$ до тех пор, пока $Q_m^2 \leq \frac{1}{\varepsilon}$. ■

 **Замечание 1.18**

Оценка $1/Q_m^2$ в (1.39) является менее точной, чем оценка $1/(Q_{m+1}Q_m)$. Но для последней нужно проводить вычисления знаменателей подходящих дробей на один шаг вперед.

Пример 1.20. вычислению $\sqrt{28}$. В примере 1.19 была получена бесконечная цепная дробь с периодом (3, 2, 3, 10).

Определение 1.11. Бесконечную цепную дробь, в которой определенная последовательность неполных частных, начиная с некоторого места, периодически повторяется, называют *периодической цепной дробью*. Период бесконечной цепной дроби будем записывать в круглых скобках следующим образом:

$$\sqrt{28} = [5; 3, 2, 3, 10, 3, 2, 3, 10, \dots, 3, 2, 3, 10, \dots] = [5; (3, 2, 3, 10)].$$

Вычислим $\sqrt{28}$ с точностью до $\varepsilon = 10^{-4}$. Используя (1.26), получаем последовательность подходящих дробей (таблица 1.16).

Во время вычисления производим контроль точности по знаменателю.

Таблица 1.16.

s	-1	0	1	2	3	4	5	6
q_s		5	3	2	3	10	3	2
P_s	1	5	16	37	127	1307	4048	9403
Q_s	0	1	3	7	24	247	765	1777

1) $\frac{P_1}{Q_1} = \frac{16}{3} = 5,3(3)$, точность $\frac{1}{3^2} = 0,1(1)$ — гарантирована одна значащая цифра.

2) $\frac{P_2}{Q_2} = \frac{37}{7} = 5,2857\dots$, точность $\frac{1}{7^2} = 0,02048\dots$ — две значащие цифры гарантированы.

3) $\frac{P_3}{Q_3} = \frac{127}{24} = 5,2916(6)$, точность $\frac{1}{24^2} = 0,0017\dots$ — три значащие цифры гарантированы. (В данном случае реально получаем четыре, так как оценка (1.39) достаточно грубая.)

4) $\frac{P_4}{Q_4} = \frac{1307}{247} = 5,291500\dots$, точность $\frac{1}{247^2} = 0,000016\dots$ — получаем пять, а реально шесть значащих цифр.

Достигнута нужная оценка. Останавливаем процесс вычислений.

$$\sqrt{28} \approx 5,291500.$$

Установим важное свойство бесконечной цепной дроби. Дадим необходимые определения.

Определение 1.12. Иррациональное число, являющееся корнем некоторого квадратного уравнения с целыми коэффициентами, называют *квадратичной иррациональностью*.

Пример 1.21. Квадратичными иррациональностями являются:

$$\frac{5 + \sqrt{17}}{4}, \frac{7 + \sqrt{29}}{10}, 3 + \sqrt{7}.$$

Выражения

$$\sqrt[3]{7}, \sqrt[5]{3} + 13, \frac{\sqrt[7]{7} + 13}{8}, \pi$$

квадратичными иррациональностями не являются.

Теорема 1.19 (Лагранж). Квадратичные иррациональности и только они представимы в виде бесконечной периодической цепной дроби.

Доказательство. Пусть $\alpha = (q_1, q_2, \dots, q_n, \dots)$ — бесконечная периодическая цепная дробь.

Воспользуемся доказательством теоремы 1.18. Запишем для α представление (1.35).

Очевидно, что в силу периодичности цепной дроби величины α_s удовлетворяют соотношению $\alpha_{s+h} = \alpha_s$, где $s > 1$, а h — период дроби. Используя (1.36), имеем

$$\alpha = \frac{P_{s-1}\alpha_s + P_{s-2}}{Q_{s-1}\alpha_s + Q_{s-2}} = \frac{P_{s+h-1}\alpha_{s+h} + P_{s+h-2}}{Q_{s+h-1}\alpha_{s+h} + Q_{s+h-2}} = \frac{P_{s+h-1}\alpha_s + P_{s+h-2}}{Q_{s+h-1}\alpha_s + Q_{s+h-2}}. \quad (1.40)$$

Из (1.40) получаем

$$\frac{P_{s-1}\alpha_s + P_{s-2}}{Q_{s-1}\alpha_s + Q_{s-2}} = \frac{P_{s+h-1}\alpha_s + P_{s+h-2}}{Q_{s+h-1}\alpha_s + Q_{s+h-2}}$$

— квадратное уравнение с целыми коэффициентами для нахождения α_s . Значит, α_s — квадратичная иррациональность, тогда в силу (1.36) α тоже квадратичная иррациональность.

Докажем достаточность. Пусть α удовлетворяет квадратному уравнению с целыми коэффициентами:

$$A\alpha^2 + B\alpha + C = 0. \quad (1.41)$$

Разложим α в цепную дробь и подставим в уравнение (1.41) вместо α его выражение в (1.36). После преобразований получается квадратное уравнение

$$A_s\alpha_s^2 + B_s\alpha_s + C_s = 0, \quad (1.42)$$

где

$$\begin{cases} A_s = AP_{s-1}^2 + BP_{s-1}Q_{s-1} + CQ_{s-1}^2, \\ B_s = 2AP_{s-1}P_{s-2} + B(P_{s-1}Q_{s-2} + P_{s-2}Q_{s-1}) + 2CQ_{s-1}Q_{s-2}, \\ C_s = AP_{s-2}^2 + BP_{s-2}Q_{s-2} + CQ_{s-2}^2. \end{cases}$$

Очевидно, что $A_s, B_s, C_s \in \mathbb{Z}$ и $C_s = A_{s-1}$. Покажем, что дискриминанты квадратных уравнений (1.41) и (1.42) совпадают при всех s .

При доказательстве леммы 1.4 было установлено равенство

$$P_sQ_{s-1} - P_{s-1}Q_s = (-1)^{s+1}.$$

Тогда

$$\begin{aligned} B_s^2 - 4A_s C_s &= (B^2 - 4AC)(P_{s-1}Q_{s-2} + P_{s-2}Q_{s-1})^2 = (B^2 - 4AC)(-1)^{2s} = \\ &= B^2 - 4AC. \end{aligned}$$

Согласно (1.39):

$$\left| \alpha - \frac{P_{s-1}}{Q_{s-1}} \right| < \frac{1}{Q_{s-1}^2}.$$

Тогда найдется число γ_{s-1} такое, что

$$P_{s-1} = \alpha Q_{s-1} + \frac{\gamma_{s-1}}{Q_{s-1}}, \quad \text{где } |\gamma_{s-1}| < 1.$$

Вычислим коэффициент A_s в квадратном уравнении (1.42):

$$\begin{aligned} A_s &= AP_{s-1}^2 + BP_{s-1}Q_{s-1} + CQ_{s-1}^2 = \\ &= A \left(\alpha Q_{s-1} + \frac{\gamma_{s-1}}{Q_{s-1}} \right)^2 + B \left(\alpha Q_{s-1} + \frac{\gamma_{s-1}}{Q_{s-1}} \right) Q_{s-1} + CQ_{s-1}^2 = \\ &= (A\alpha^2 + B\alpha + C) Q_{s-1}^2 + 2A\alpha\gamma_{s-1} + A \frac{\gamma_{s-1}^2}{Q_{s-1}^2} + B\gamma_{s-1} = \\ &= 2A\alpha\gamma_{s-1} + A \frac{\gamma_{s-1}^2}{Q_{s-1}^2} + B\gamma_{s-1}, \end{aligned}$$

так как α — корень уравнения (1.41). Тогда для любого натурального s

$$|A_s| = \left| 2A\alpha\gamma_{s-1} + A \frac{\gamma_{s-1}^2}{Q_{s-1}^2} + B\gamma_{s-1} \right| < 2|A\alpha| + |A| + |B|.$$

Как было замечено ранее, $C_s = A_{s-1}$, поэтому для C_s справедлива та же оценка.

Таким образом, целые коэффициенты A_s и C_s уравнения (1.42) ограничены по абсолютному значению и, следовательно, при изменении s могут принимать лишь конечное число различных значений.

Так как дискриминанты уравнений (1.41) и (1.42) совпадают, то и коэффициент B_s может принимать лишь конечное число различных значений. Поэтому при изменении s от 1 до ∞ может встретиться только конечное число различных уравнений вида (1.42), т. е. лишь конечное число различных чисел α_s . Это значит, что некоторые два числа α_s и α_{s+h} с разными номерами обязательно совпадают, что и указывает на периодичность цепной дроби. ■

Замечание 1.19

Отметим без доказательства следующие свойства разложений квадратичных иррациональностей:

- 1) при разложении корня квадратного из целого положительного числа, не являющегося полным квадратом, период всегда начинается со второго звена;
- 2) периодическая цепная дробь, у которой период начинается с первого звена, получается тогда и только тогда, когда квадратичная иррациональность больше 1, а сопряженная иррациональность лежит в интервале $(-1, 0)$ (это свойство было доказано Э. Галуа в 1828 г. Он доказал также, что в этом случае сопряженная квадратичная иррациональность имеет те же элементы, но расположенные в обратном порядке).

Пример 1.22. Как известно, число e — трансцендентное и не является квадратичной иррациональностью. Тем не менее интересно разложение числа e в бесконечную (непериодическую) цепную дробь.

$$e = [2; 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1, 1, 10, 1, \dots].$$

1.2.9. Наилучшие приближения

Определение 1.13. Дробь $\frac{a}{b}$ называют наилучшим приближением к числу α , если не существует дроби $\frac{x}{y}$, у которой

$$0 < y \leq b \text{ и } \left| \alpha - \frac{x}{y} \right| < \left| \alpha - \frac{a}{b} \right|.$$

Замечание 1.20

Знаменатели дробей, не умаляя общности, считаем положительными. Наилучшее приближение не обязательно является единственным. У иррационального числа наилучших приближений бесконечно много.

Пример 1.23. Пусть $\alpha = \frac{6}{17}$, тогда $\frac{1}{2}$, $\frac{1}{3}$, $\frac{4}{11}$, $\frac{5}{14}$ являются наилучшими приближениями, причем других нет.

Теорема 1.20. Рассмотрим промежуток $\left(\frac{a}{b}; \frac{c}{d} \right)$, обладающий свойством

$$bc - ad = 1, \quad \frac{c}{d} - \frac{a}{b} = \frac{1}{bd}.$$

Тогда знаменатели всех рациональных чисел, лежащих внутри промежутка, больше знаменателей его концов, т. е. если $\frac{x}{y} \in \left(\frac{a}{b}; \frac{c}{d}\right)$, то $y > b$ и $y > d$.

Доказательство. Рассмотрим разность

$$\frac{x}{y} - \frac{a}{b} > 0 : \frac{x}{y} - \frac{a}{b} = \frac{xb - ay}{by} \geq \frac{1}{by},$$

так как $xb - ay$ — большее нуля целое. С другой стороны,

$$\frac{x}{y} - \frac{a}{b} < \frac{c}{d} - \frac{a}{b} = \frac{bc - ad}{bd} = \frac{1}{bd}.$$

Откуда $\frac{1}{bd} > \frac{1}{by}$, следовательно, $y > d$. Аналогично доказывается, что $y > b$. ■

Следствие 1.7. Пусть в условиях теоремы 1.20 $\alpha \in \left(\frac{a}{b}; \frac{c}{d}\right)$, тогда та

из двух границ $\frac{a}{b}$ и $\frac{c}{d}$, которая лежит ближе к числу α , является наилучшим приближением к нему.

Следствие 1.8. Все подходящие дроби для разложения числа α в цепную (непрерывную) дробь, начиная со второй, являются наилучшими приближениями к α .

Доказательство. Рассмотрим последовательность подходящих дробей к α :

$$\frac{P_0}{Q_0}, \frac{P_1}{Q_1}, \dots, \frac{P_n}{Q_n}, \frac{P_{n+1}}{Q_{n+1}}, \dots$$

Так как $P_{n+1}Q_n - P_nQ_{n+1} = (-1)^n$, то любые две подходящие дроби, взятые в нужном порядке, обладают свойствами концов промежутка, описанного в теореме. Согласно свойствам подходящих дробей α всегда находится между последовательными подходящими дробями. Поэтому ближайшая к α граница является наилучшим приближением. Согласно свойствам подходящих дробей это всегда дробь с большим номером. ■

Упражнение 1.5. Найдите три наилучших приближения к π .

1.2.10. Некоторые применения цепных дробей

✓ **Зубчатые колеса.** Рассмотрим задачу, аналогичную той, с которой голландский математик Х. Гюйгенс (1629–1695) столкнулся при построении модели солнечной системы с помощью набора зубчатых колес и которая привела его к открытию ряда важных свойств цепных дробей.

Пусть требуется соединить два вала зубчатыми колесами так, чтобы отношение их угловых скоростей равнялось данному числу α . Известно, что угловые скорости колес обратно пропорциональны количеству зубцов, значит, обратное отношение чисел зубцов равно α .

Но количество зубцов — это целые числа, хотя и не очень большие, тогда как α может быть и иррациональным. Следовательно, задачу можно решить только приближенно, взяв для α приближенное значение в форме простой дроби с не очень большим знаменателем.

Из предыдущих рассуждений вытекает, что наиболее выгодно взять одну из подходящих дробей цепной дроби, в которую раскладывается число α .

✓ **Календарный стиль.** Известно, что в году 365.24220... так называемых «средних» суток или 365 суток 5 часов 48 минут и 46 секунд. Конечно, такое сложное отношение года к суткам в практической жизни совершенно неудобно. Нужно заменить его более простым, хотя и менее точным.

Разложив 365.24220... в цепную дробь, получим

$$365.24220 \dots = 365 + \frac{1}{4 + \frac{1}{7 + \frac{1}{1 + \frac{1}{3 + \dots}}}}$$

Первыми подходящими дробями здесь будут:

$$365, \quad 365\frac{1}{4}, \quad 365\frac{7}{29}, \quad 365\frac{8}{33}, \quad 365\frac{31}{128}.$$

Историческая справка

Приближение $365\frac{1}{4}$ было известно еще древним египтянам, ассирийцам, китайцам, хотя они не имели регулярных високосных годов. 7 марта 238 г. до н. э. вышел Канопский декрет Птолемея Эвергета, которым предписывалось, чтобы каждый четвертый год имел не 365, а 366 сут. Но через 40 лет этот декрет был забыт, и только в 47 году до н. э. Юлий Цезарь при участии Сосигена Александрийского возобновил его, вставив в каждом четвертом году лишний день в феврале. Этот

день называли *bissextilis*, откуда происходит и русское название «високосный». Это — старый, или юлианский, календарь.

Новый, или григорианский, стиль дает приближение $365\frac{97}{400}$, или 365.2425 суток, т. е. 365 суток 5 часов 49 минут и 12 секунд. Это значение больше истинного лишь на 26 с. Такая дробь дает приближение немного хуже, чем вторая подходящая дробь $365\frac{7}{29}$. Этот стиль отличается от юлианского тем, что в нем каждый сотый год не високосный, кроме тех, число сотен которых делится на 4. Так, 1700-й, 1800-й, 1900-й годы не високосные, тогда как 1600-й, 2000-й годы високосные, т. е. 400 лет имеют 97 лишних суток, а не 100 суток, как в юлианском стиле.

Уже в XV в. было замечено отставание юлианского стиля (тогда на 10 суток) и поступили предложения реформировать календарь, что осуществилось только в конце XVI в. в католических странах буллой папы Григория XIII от 1 марта 1582 г. 10 суток (5–14 октября) были вычеркнуты, т. е. вместо пятого сразу наступило 15 октября 1582 г.

Интересная система календаря была предложена в Персии в 1079 г. персидским астрономом, математиком и поэтом Омаром Альхайями. Он ввел цикл из 33 лет, в котором семь раз високосным годом считался четвертый, а восьмой раз — пятый, таким образом, продолжительность года $365\frac{8}{33}$ суток — это четвертая подходящая дробь. Погрешность календаря составляет всего 19 с в год.

В 1864 г. немецкий астроном Иоганн Медлер предложил с XX в. ввести следующую поправку к юлианскому календарю: через каждые 128 лет пропускать один високосный год из 32, которые выпадают на этот период. Этот календарь является самым точным из всех перечисленных. Средняя продолжительность года по этому

календарю составляет $365\frac{31}{128}$ суток, или 365 суток 5 часов 48 минут и 45 секунд.

При этом погрешность сокращается всего до 1 секунды. Однако календарь не был принят, видимо, из-за того, что период 128 лет не является круглым числом.

1.2.11. Диофантовы уравнения

Определение 1.14. Пусть $a, b, c, x, y \in \mathbb{Z}$. Уравнение вида

$$ax + by = c, \quad (1.43)$$

где x, y — неизвестные, называют *диофантовым*.

Историческая справка

О подробностях жизни Диофанта Александрийского практически ничего не известно. Все, что знают историки, почерпнуто из надписи на его гробнице. Основной труд Диофанта — «Арифметика». Уцелели только шесть книг оригинала, общее их число точно неизвестно (предположительно — тринадцать).

Его труд — один из наиболее увлекательных трактатов, сохранившихся от греко-римской древности. В нем впервые встречается систематическое использование алгебраических символов, есть особые знаки для обозначения неизвестного, минуса, обратной величины, возведения в степень. Среди уравнений, решаемых Диофантом,

присутствуют такие, как

$$x^2 - 26y^2 = 1 \text{ и } x^2 - 30y^2 = 1,$$

теперь известные как частные случаи «уравнения Пелля», причем Диофант интересуется их решениями именно в целых числах.

«Арифметика» Диофанта неожиданно оказала еще и огромное косвенное влияние на развитие математической науки последних трех столетий. Дело в том, что французский математик П. Ферма (1601–1665), изучая «Арифметику», сделал на полях этой книги знаменитую пометку: «Я нашел воистину удивительное доказательство того, что уравнение

$$x^n + y^n = z^n \text{ при } n > 2$$

не имеет решений в целых числах, однако поля этой книги слишком малы, чтобы здесь его уместить». Это утверждение получило название «Великой теоремы Ферма».

Установим критерий разрешимости диофантового уравнения.

Теорема 1.21. Пусть $d = D(a, b)$. Уравнение (1.43) разрешимо тогда и только тогда, когда $c \vdots d$. При этом все множество решений описывается уравнениями:

$$x = x_0 - \frac{b}{d}t, \quad y = y_0 + \frac{a}{d}t, \quad (1.44)$$

где x_0, y_0 — любое частное решение (1.43); t — произвольное целое число.

Доказательство. Так как $d = D(a, b)$, то левая часть уравнения (1.43) делится на d . Следовательно, для разрешимости необходимо и достаточно, чтобы и правая часть делилась на d .

Пусть $c = de$. Согласно утверждению 1.1 уравнение $ax + by = d$ всегда имеет решение. Обозначим его через x_1, y_1 . Тогда $x_0 = x_1e, y_0 = y_1e$ — решение исходного уравнения.

Найдем общий вид решения. Пусть x, y — произвольное решение (1.43). Покажем, что оно представимо в виде (1.44). Имеем

$$a(x - x_0) + b(y - y_0) = 0, \quad \text{тогда} \quad \frac{a}{d}(x - x_0) = -\frac{b}{d}(y - y_0).$$

Согласно теореме 1.3 $D\left(\frac{a}{d}, \frac{b}{d}\right) = 1$.

Тогда согласно теореме 1.6 $(y - y_0) \vdots \frac{a}{d}$, следовательно, $y = y_0 + \frac{a}{d}t$.

Подставив найденное значение y в уравнение, получаем $x = x_0 - \frac{b}{d}t$. Таким образом, если x, y — решение (1.43), то оно имеет вид (1.44). Обратное утверждение проверяется непосредственной подстановкой. ■

Следствие 1.9. Справедливы следующие утверждения:

- 1) если $D(a, b) = 1$, то уравнение (1.43) всегда разрешимо;
- 2) решения уравнений $ax + by = c$ и $|a|x + |b|y = c$ могут отличаться только знаками.

Доказательство. Первое утверждение очевидно. Пусть x, y — решения уравнения $ax + by = c$, а x_1, y_1 — соответственно уравнения $|a|x + |b|y = c$. Тогда $x = x_1 \text{sign}(a)$, $y = y_1 \text{sign}(b)$. ■

Опишем алгоритм решения уравнения (1.43).

Алгоритм 1.16. Решение диофантового уравнения

Находим линейное представление $D(|a|, |b|)$, используя обобщенный алгоритм Евклида.

$$d = D(|a|, |b|) = |a|x_1 + |b|y_1.$$

if $c = d\alpha$ **then** *// c делится на d*

$x_0 \leftarrow \alpha x_1 \text{sign}(a)$ *// (x₀, y₀) — частное решение (1.43)*

$y_0 \leftarrow \alpha y_1 \text{sign}(b)$

Подставив (x_0, y_0) в (1.44), получаем общее решение исходного уравнения.

else

уравнение (1.43) не имеет решений

end if

 **Замечание 1.21**

Заметим, что коэффициенты последнего столбца таблицы протокола расширенного алгоритма Евклида равны соответственно $-b/d$ и a/d . Это следует из представления

$$0 = a \left(-\frac{b}{d} \right) + b \left(\frac{a}{d} \right).$$

Пример 1.24. Решим уравнение $126x - 102y = 18$.

Применив расширенный алгоритм Евклида для чисел 126 и 102, запишем протокол его работы (таблица 1.17).

Таблица 1.17.

	126	102	24	6	0
q			1	4	4
x	1	0	1	-4	17
y	0	1	-1	5	-21

Таким образом,

$$D(126, 102) = 6 = 126(-4) + 102(5), \text{ тогда } 18 = 126(-12) - 102(-15).$$

Уравнение разрешимо, так как $18 : 6$. Тогда $x_0 = -12$, $y_0 = -15$. Согласно замечанию 1.21 к алгоритму 1.16 с учетом знака b имеем $\frac{b}{d} = -17$, $\frac{a}{d} = 21$. Запишем общее решение:

$$x = -12 + 17t, \quad y = -15 + 21t, \quad t \in \mathbb{Z}.$$

Пример 1.25. Решим уравнение $253x - 449y = 1$.

Применив расширенный алгоритм Евклида для чисел 253 и 449, запишем протокол его работы (таблица 1.18).

Таблица 1.18.

	253	449	253	196	57	25	7	4	3	1	0
q			0	1	1	3	2	3	1	1	3
x	1	0	1	-1	2	-7	16	-55	71	-126	449
y	0	1	0	1	-1	4	-9	31	-40	71	-253

Таким образом,

$$D(253, 449) = 1 = 253(-126) + 449(71) = 253(-126) - 449(-71).$$

Уравнение разрешимо, так как $1 : 1$. Тогда $x_0 = -126$, $y_0 = -71$. Запишем общее решение:

$$x = -126 + 449t, \quad y = -71 + 253t, \quad t \in \mathbb{Z}.$$

Пример 1.26. Решим уравнение $2071x - 2318y = -95$.

Применив расширенный алгоритм Евклида для чисел 2071 и 2318, запишем протокол его работы (таблица 1.19).

Таблица 1.19.

	2071	2318	2071	247	95	57	38	19	0
q			0	1	8	2	1	1	2
x	1	0	1	-1	9	-19	28	-47	122
y	0	1	0	1	-8	17	-25	42	-109

Таким образом, представление для наибольшего общего делителя:

$$D(2071, 2318) = 19 = 2071(-47) + 2318(42).$$

Для исходного уравнения получаем

$$2071(235) - 2318(210) = -95.$$

Уравнение разрешимо, так как $95 : 19$. Тогда $x_0 = 235$, $y_0 = 210$. Запишем общее решение:

$$x = 235 + 122t, \quad y = 210 + 109t, \quad t \in \mathbb{Z}.$$

Теорему 1.21 можно обобщить на случай нескольких переменных.

Теорема 1.22. Уравнение $a_1x_1 + \dots + a_nx_n = c$ разрешимо тогда и только тогда, когда $D(a_1, \dots, a_n) \mid c$.

Доказательство. Пусть $d = D(a_1, \dots, a_n)$.

Так как $a_1, \dots, a_n : d$, то и $a_1x_1 + \dots + a_nx_n : d$. Следовательно, если c не делится на d , то решений нет. Пусть $c = de$.

Докажем по индукции, что уравнение

$$a_1x_1 + a_2x_2 + \dots + a_nx_n = D(a_1, a_2, \dots, a_n) \quad (1.45)$$

всегда разрешимо.

База индукции (при $n = 2$) вытекает из теоремы 1.21. Предположим разрешимость уравнения для произвольного n и докажем этот факт для $n + 1$.

Рассмотрим уравнение

$$a_1u_1 + a_2u_2 + \dots + a_nu_n + a_{n+1}u_{n+1} = D(a_1, a_2, \dots, a_n, a_{n+1}),$$

при этом согласно теореме 1.4:

$$D(a_1, a_2, \dots, a_n, a_{n+1}) = D(D(a_1, a_2, \dots, a_n), a_{n+1}) = D(d, a_{n+1}).$$

Обозначим $\delta = D(d, a_{n+1})$. Тогда по теореме 1.21 уравнение

$$dy + a_{n+1}u_{n+1} = \delta$$

разрешимо. Подставив вместо $d \rightarrow a_1x_1 + a_2x_2 + \dots + a_nx_n$, получаем решение уравнения

$$a_1(x_1y) + a_2(x_2y) + \dots + a_n(x_ny) + a_{n+1}u_{n+1} = \delta.$$

Индукционный переход доказан.

Таким образом, уравнение (1.45) разрешимо для всех n . Учитывая, что $c = de$, получаем решение исходного уравнения:

$$a_1(x_1e) + a_2(x_2e) + \dots + a_n(x_ne) = c.$$

■

1.2.12. Числа Фибоначчи

Историческая справка

Фибоначчи — «Сын Боначчо», он же Леонардо Пизанский (1180–1240) — известный средневековый математик, философ, купец и т. д. Путешествовал и торговал в странах Востока. По возвращении в Европу он записал собранные сведения, добавил много собственных исследований и издал книги «Практика геометрии» и «Книга абака».

Последовательность Фибоначчи возникает у самого Леонардо при решении следующей задачи: сколько пар кроликов может произойти от одной пары в течение года, если: а) каждая пара ежемесячно порождает новую пару, которая со второго месяца способна к размножению, и б) кролики не погибают.

Последовательность Фибоначчи появляется не только при изучении цепных дробей, но и во многих других разделах математики, физики, биологии, искусствоведения. Добавим, что «Книга абака» была одним из решающих источников проникновения в Западную Европу десятичной системы счисления и арабских цифр.

Название «числа Фибоначчи» стало общеупотребительным только в XIX в., благодаря математику Люка. Однако индийские математики упоминали числа этой последовательности еще в XII в. Для них это было связано с задачей о количестве ритмических рисунков, образующихся в результате чередования долгих и кратких слогов в стихах или сильных и слабых долей в музыке. Число таких рисунков, имеющих в целом n долей, равно F_n .

Числа Фибоначчи появляются и в работе Кеплера 1611 г., который размышлял о числах, встречающихся в природе («О шестиугольных снежинках»).

Интересен пример растения тысячелистника, у которого число стеблей (а значит и цветков) всегда есть число Фибоначчи. Причина этого проста: будучи изначально с единственным стеблем, этот стебель затем делится на два, затем от главного стебля ответвляется еще один, затем первые два стебля снова разветвляются, затем все стебли, кроме двух последних, разветвляются, и т. д. Таким образом, каждый стебель после своего появления «пропускает» одно разветвление, а затем начинает делиться на каждом уровне разветвлений, что и дает в результате числа Фибоначчи. Вообще го-

вора, у многих цветов (например, лилий) число лепестков является тем или иным числом Фибоначчи.

Перейдем к точным определениям.

Определение 1.15. Рекуррентное соотношение для чисел Фибоначчи имеет вид

$$\phi_{k+2} = \phi_{k+1} + \phi_k, \quad k \geq 1, \quad \phi_1 = \phi_2 = 1. \quad (1.46)$$

Фибоначчи в 1228 г. привел первые 14 чисел этой последовательности:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, \dots, 39\,088\,169, \dots$$

Замечание 1.22

Максимальным числом Фибоначчи, которое помещается в тип `int`, является $\phi_{46} = 1836311903$. В 64-битовый целочисленный тип `long long` помещается максимум — ϕ_{92} .

Если в задаче требуется находить значения ϕ_n для $n > 92$, то следует воспользоваться длинной арифметикой.

Рассмотрим некоторые свойства чисел Фибоначчи.

Лемма 1.7 (соотношение Кассини). Для последовательности чисел Фибоначчи справедливо тождество

$$\phi_{n-1}\phi_{n+1} - \phi_n^2 = (-1)^n$$

Доказательство. Введем обозначение: $\Delta_n = \phi_{n-1}\phi_{n+1} - \phi_n^2$. Далее, согласно (1.46) имеем:

$$\begin{aligned} \Delta_{n+1} &= \phi_n\phi_{n+2} - \phi_{n+1}^2 = (\phi_{n+1} - \phi_n)\phi_n - (\phi_n + \phi_{n-1})^2 = \\ &= \phi_{n+1}\phi_n + \phi_n^2 - \phi_n^2 - 2\phi_n\phi_{n-1} - \phi_{n-1}^2 = \\ &= \phi_{n+1}\phi_n - 2\phi_n\phi_{n-1} - \phi_{n-1}^2 = (\phi_n + \phi_{n-1})\phi_n - 2\phi_n\phi_{n-1} - \phi_{n-1}^2 = \\ &= \phi_n^2 + \phi_{n-1}\phi_n - 2\phi_n\phi_{n-1} - \phi_{n-1}^2 = \phi_n^2 - \phi_n\phi_{n-1} - \phi_{n-1}^2 = \\ &= \phi_n^2 - (\phi_n + \phi_{n-1})\phi_{n-1} = \phi_n^2 - \phi_{n+1}\phi_{n-1} = -\Delta_n. \end{aligned}$$

Таким образом, $\Delta_{n+1} = -\Delta_n$. Учитывая, что $\Delta_2 = 1$, получаем требуемое. ■

Историческая справка

Кассини — это знаменитая династия французских астрономов. Наиболее известным из них считается основатель этой династии Джовани Доменико Кассини (1625–1712). Именем Джовани Кассини названы многие астрономические объекты: Кратер Кассини на Луне, Кратер Кассини на Марсе, Щель Кассини — промежуток в кольцах Сатурна, Законы Кассини — три открытые Кассини закона движения Луны.

Теорема 1.23 (Цекендорф). Любое натуральное число N можно представить единственным образом в виде суммы чисел Фибоначчи:

$$N = \phi_{k_1} + \phi_{k_2} + \dots + \phi_{k_r}, \quad k_1 > k_2 + 1, k_2 > k_3 + 1, \dots, k_r \geq 1, \quad (1.47)$$

т. е. в записи (1.47) нельзя использовать два соседних числа Фибоначчи.

Доказательство. Воспользуемся индукцией по номерам чисел Фибоначчи n .

Из равенства (1.46) легко заметить, что любое натуральное число N попадает в промежуток между двумя соседними числами Фибоначчи, т. е. для некоторого $n \geq 2$ верно неравенство

$$\phi_n \leq N < \phi_{n+1}.$$

Таким образом, $N = \phi_n + N_1$, где $N_1 = N - \phi_n < \phi_{n-1}$, так что разложение числа N_1 уже не будет содержать слагаемого ϕ_{n-1} . ■

Из равенства (1.47) очевидно следует, что любое число можно однозначно записать в *фибоначчиевой системе счисления*, например:

$$\begin{aligned} 9 &= 8 + 1 = \phi_6 + \phi_1 = (100001)_\phi, \\ 6 &= 5 + 1 = \phi_5 + \phi_1 = (10001)_\phi, \\ 19 &= 13 + 5 + 1 = \phi_7 + \phi_5 + \phi_1 = (1010001)_\phi, \end{aligned}$$

причём ни в каком числе не могут идти две единицы подряд.

Перевод числа N в фибоначчиеву систему счисления осуществляется простым алгоритмом: перебираем числа Фибоначчи от больших к меньшим и, если некоторое $\phi_k \leq N$, включаем ϕ_k в запись числа N , полагаем $N = N - \phi_k$ и продолжаем поиск.

Формула для общего члена последовательности чисел Фибоначчи (1.121) будет получена далее в разделе 1.7.1 с помощью аппарата производящих функций.

Числа Фибоначчи тесно связаны с *числом Фидия*¹ Φ , характеризующим так называемое золотое сечение или «божественную пропорцию».

Определение 1.16. Рассмотрим деление единичного отрезка на две части. Большую часть отрезка обозначим через α . Если большая часть так относится к меньшей, как весь отрезок к большей его части, то говорят, что имеет место «золотое сечение» отрезка (рисунок 1.2).

В этом случае данное отношение называют числом Фидия.

¹ Названо так в честь древнегреческого скульптора и архитектора, для работ которого характерно применение золотого сечения.

$$\Phi = \frac{\alpha}{1 - \alpha} = \frac{1}{\alpha}. \quad (1.48)$$

Замечание 1.23

Легко показать, что в определении 1.16 можно рассматривать отрезок любой длины.

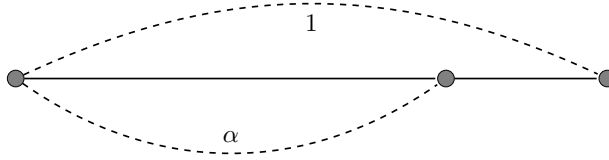


Рис. 1.2.

Из (1.48) просто вычислить число Фидия. Имеем:

$$\alpha^2 = 1 - \alpha, \quad \Phi = \frac{1 + \sqrt{5}}{2}.$$

Установим связь чисел Фибоначчи с числом Фидия.

Теорема 1.24. Справедливо следующее соотношение:

$$\phi_n = \frac{\Phi^n - (1 - \Phi)^n}{\sqrt{5}} \quad (1.49)$$

Доказательство. Применим метод математической индукции.

Нетрудно проверить, что соотношение (1.49) выполняется для ϕ_1 и ϕ_2 . База индукции установлена.

Предположим, что равенство (1.49) выполняется для всех $\phi_k, k < n$, и покажем его справедливость для ϕ_n . Заметим, что из (1.48) следуют очевидные равенства:

$$1 + \Phi = \Phi^2, \quad 2 - \Phi = (1 - \Phi)^2. \quad (1.50)$$

Далее, с учетом (1.50) получаем:

$$\begin{aligned} \phi_n &= \phi_{n-1} + \phi_{n-2} = \frac{\Phi^{n-1} - (1 - \Phi)^{n-1}}{\sqrt{5}} + \frac{\Phi^{n-2} - (1 - \Phi)^{n-2}}{\sqrt{5}} = \\ &= \frac{\Phi^{n-2}(1 + \Phi) - (1 - \Phi)^{n-2}(2 - \Phi)}{\sqrt{5}} = \frac{\Phi^n - (1 - \Phi)^n}{\sqrt{5}} \end{aligned}$$



В следующем разделе будет получено интересное свойство периодичности последовательности чисел Фибоначчи, связанное с модульной арифметикой (теорема 1.33).

Задачи и вопросы

1. Сколько шагов сделает бинарный алгоритм Евклида при нахождении $D(4^{10}, 6)$.

2. Для какой пары чисел алгоритм Евклида работает медленнее всего?

3. Число α раскладывается в бесконечную цепную периодическую дробь

$$\alpha = (a_0; (a_1, a_2, \dots, a_k)).$$

Какое из чисел больше — α или $a_0 + \frac{1}{a_1}$?

4. Найдите обыкновенную дробь с наименьшим знаменателем, для которой $\frac{34}{23}$ является предпоследней подходящей дробью.

5. Представьте $\sqrt{\alpha}$ в виде периодической цепной дроби и вычислите с точностью до $\varepsilon = 10^{-5}$:

а) $\alpha = 252$;

б) $\alpha = 155$;

в) $\alpha = 119$.

6. Сформулируйте критерий разрешимости диофантова уравнения.

7. Докажите, что диофантово уравнение $ax + (a + 1)y = a + 2$ всегда разрешимо. Пусть (x_0, y_0) — частное решение этого уравнения. Опишите все множество решений этого уравнения.

8. Решите диофантово уравнение:

а) $4636x + 4389y = -171$;

б) $1534x + 1547y = 26$;

в) $2461x - 2438y = -115$.

9. Возможно ли такое представление числа в фибоначчиевой системе счисления: $(1, 0, 0, 1, 1, 0, 1, 1, 0, 1)$?

1.3. Арифметика остатков

1.3.1. Алгебраические структуры

Для последующих подразделов нам понадобятся некоторые понятия алгебры. Дадим необходимые определения. В дальнейшем запись (Ω, \circ) означает, что на непустом множестве Ω задана операция \circ .

Определение 1.17. Множество (Ω, \circ) называют *группой*, если выполнены следующие условия:

- 1) операция \circ ассоциативна, т. е. для любых $a, b, c \in \Omega$ выполнено

$$a \circ (b \circ c) = (a \circ b) \circ c;$$

- 2) множество Ω обладает нейтральным элементом относительно операции \circ , т. е. существует элемент $e \in \Omega$ такой, что для любого элемента $a \in \Omega$ выполнено

$$a \circ e = e \circ a = a;$$

- 3) для любого элемента $a \in \Omega$ существует противоположный элемент $\tilde{a} \in \Omega$ такой, что

$$a \circ \tilde{a} = \tilde{a} \circ a = e.$$

Если для любых $a, b \in \Omega$ справедливо

$$a \circ b = b \circ a,$$

то такую группу называют *коммутативной* или *абелевой*. В коммутативных группах операцию \circ обычно называют операцией сложения и обозначают $+$, нейтральный элемент e называют нулем группы и обозначают 0 , хотя этот элемент, вообще говоря, числом не является. Противоположный элемент \tilde{a} обозначают $-a$.

Пример 1.27. Примеры абелевых групп:

а) аддитивные группы (группы по сложению) целых чисел \mathbb{Z} , рациональных чисел \mathbb{Q} и вещественных чисел \mathbb{R} ;

б) мультипликативные группы (группы по умножению) положительных рациональных чисел \mathbb{Q}_+ и положительных вещественных чисел \mathbb{R}_+ .

Определение 1.18. Непустое множество Ω , на котором заданы операции сложения и умножения, называют *кольцом*, если выполнены следующие два условия:

- а) $(\Omega, +)$ — абелева группа;

б) умножение дистрибутивно относительно сложения, т. е. для любых элементов $x, y, z \in \Omega$ выполнены равенства:

$$(x + y)z = xz + yz; \quad x(y + z) = xy + xz.$$

Кольцо называют *коммутативным*, если операция умножения в нем коммутативна; кольцо называют *ассоциативным*, если операция умножения в нем ассоциативна. Кольцо называют *кольцом с единицей*, если оно обладает нейтральным элементом относительно умножения.

Пусть Ω — ассоциативное кольцо с единицей (обозначим ее как e). Элемент $a \in \Omega$ называют *обратимым*, если существует элемент $b \in \Omega$ такой, что

$$ab = ba = e.$$

Легко проверить, что элемент b , о котором идет речь, находится однозначно, поэтому его обозначают a^{-1} и называют элементом, обратным к a .

Если для ненулевых элементов $a, b \in \Omega$ справедливо $ab = 0$, то такие элементы называют *делителями нуля*. Очевидно, что делители нуля необратимы.

Пример 1.28. Примеры колец:

- а) кольцо целых чисел \mathbb{Z} , кольцо рациональных чисел \mathbb{Q} , кольцо вещественных чисел \mathbb{R} ;
- б) кольцо $\mathbb{Z}[i]$ целых гауссовых чисел вида $a + bi$, где $a, b \in \mathbb{Z}$;
- в) кольцо $\mathbb{Z}[\sqrt{2}]$ вещественных чисел вида $a + b\sqrt{2}$ с целыми a, b .

Важнейшим типом колец являются поля.

Определение 1.19. Ассоциативное коммутативное кольцо с единицей называют *полем*, если в нем всякий ненулевой элемент обратим.

Пример 1.29. Примеры полей:

- а) числовые поля \mathbb{Q}, \mathbb{R} ;
- б) поле $\mathbb{Q}[i]$ рациональных чисел вида $a + bi$, где $a, b \in \mathbb{Q}$;
- в) поле $\mathbb{Q}[\sqrt{2}]$ вещественных чисел вида $a + b\sqrt{2}$ с рациональными a, b .

Другие примеры колец и полей будут приведены в следующем подразделе.

1.3.2. Арифметика и свойства сравнений

Определение 1.20. Пусть $a, b, m \in \mathbb{Z}, m > 0$. Говорят, что a сравнимо с b по модулю m ($a \equiv b \pmod{m}$), или сокращенно $a \equiv b(m)$, если $(a-b) : m$.

Историческая справка

| Знак \equiv для обозначения операции сравнения впервые ввел К. Гаусс в 1801 г.

Теорема 1.25. Для отношения сравнения справедливо:

- 1) $a \equiv b(m)$, тогда $b \equiv a(m)$ (симметричность);
- 2) $a \equiv b(m), b \equiv c(m)$, тогда $a \equiv c(m)$ (транзитивность);
- 3) $a \equiv a(m)$ (рефлексивность).

Замечание 1.24

| Известные бинарные отношения ($>, <, \leq, \geq, :$) такими свойствами не обладают.

Определение 1.21. Классом вычетов по модулю m называют множество чисел с одинаковым остатком при делении на m .

Теорема 1.26. Все числа из \mathbb{Z} распределяются относительно данного модуля на m непересекающихся классов вычетов со следующими свойствами:

- 1) все числа одного класса сравнимы друг с другом по модулю m ;
- 2) числа из разных классов друг с другом не сравнимы.

Доказательство. Пусть $a \in \mathbb{Z}$. Тогда по теореме 1.1 $a = mq + r, 0 \leq r < m$. Очевидно, что классов вычетов ровно m и все числа из одного класса сравнимы друг с другом.

Возьмем числа из разных классов: $a \equiv r(m), b \equiv r_1(m), r \neq r_1$. Тогда $a \not\equiv b(m)$.

Действительно, предположим противное. Пусть $(a-b) : m$, но при этом $a = qt + r, b = q_1t + r_1$, тогда $(r - r_1) : m$, следовательно, $r = r_1$. ■

Определение 1.22. Если взять по одному представителю из каждого класса вычетов, то эти m чисел образуют полную систему вычетов по модулю m .

Пример 1.30. Простейшие полные системы вычетов:

- 1) $\{0, 1, 2, \dots, m-1\}$ — наименьшие положительные вычеты;
- 2) $\{0, -1, -2, \dots, -(m-1)\}$ — наименьшие отрицательные вычеты;

- 3) для произвольного $a \in \mathbb{Z} \setminus \{a, a+1, a+2, \dots, a+(m-1)\}$;
 4) если $D(a, m) = 1$, то $\{0, a, 2a, \dots, (m-1)a\}$.

Теорема 1.27. Справедливы следующие свойства сравнений:

- 1) $a \equiv b(m)$, $m : k$, тогда $a \equiv b(k)$;
 2) $a \equiv b(k_1), \dots, a \equiv b(k_n)$, тогда $a \equiv b(M(k_1, \dots, k_n))$;
 3) $a \equiv b(m)$, тогда $ac \equiv bc(mc)$.
 4) $a \equiv b(m_1), \dots, a \equiv b(m_k)$, $D(m_i, m_j) = 1$, $i \neq j$, тогда

$$a \equiv b(m_1 m_2 \dots m_k).$$

Доказательство. Свойство 1 очевидно из определения.

Свойство 2 следует из того, что $(a-b)$ — общее кратное чисел k_1, \dots, k_n , а любое кратное делится на наименьшее (см. раздел 1.1.2).

Для доказательства свойства 3 заметим, что

$$\frac{ac - bc}{mc} = \frac{a - b}{m}.$$

Свойство 4 следует из определения операции сравнения по модулю и следствия к теореме 1.6. ■

Следствие 1.10. $a \equiv b(m)$, тогда $ac \equiv bc(m)$ для любого $c \in \mathbb{Z}$.

Теорема 1.28 (арифметика сравнений). Сравнения по одному модулю можно почленно складывать, вычитать и перемножать.

Доказательство. Пусть $a \equiv b(m)$, $a_1 \equiv b_1(m)$. Тогда

$$((a - b) + (a_1 - b_1)) = ((a + a_1) - (b + b_1)) : m,$$

следовательно, $a + a_1 \equiv b + b_1(m)$. С другой стороны, так как $a \equiv b(m)$, по следствию 1.10 $-a \equiv -b(m)$. Повторив предыдущие рассуждения, получим $a - a_1 \equiv b - b_1(m)$.

Используя следствие 1.10 для сравнения $a \equiv b(m)$, получаем $aa_1 \equiv ba_1(m)$, применяя это же следствие для второго сравнения $a_1 \equiv b_1(m)$, имеем $ba_1 \equiv bb_1(m)$. Тогда по транзитивности сравнений (см. теорему 1.25) $aa_1 \equiv bb_1(m)$. ■

Следствие 1.11. Утверждение теоремы 1.28 можно обобщить на любое конечное число сравнений. В частности, если $a \equiv b(m)$, то

$$a^n \equiv b^n(m) \quad \text{для любого } n \in \mathbb{N}.$$

Теорема 1.28 определяет действия над классами вычетов по данному модулю m . Если $a \in A, b \in B$, тогда $A + B$ — класс вычетов, к которому принадлежит $a + b$, а AB — это класс, к которому принадлежит ab .

Таким образом, справедлива следующая теорема.

Теорема 1.29. Множество классов вычетов (обозначают \mathbb{Z}_m , также используют обозначения $\mathbb{Z}/(m)$ и $\mathbb{Z}/m\mathbb{Z}$) является ассоциативным коммутативным кольцом с единицей.

Доказательство предлагается провести самостоятельно с использованием определения 1.18.

Пример 1.31. Построим несколько простейших признаков делимости, как иллюстрацию арифметики сравнений. Пусть число m представлено в десятичной системе счисления. Согласно (1.15) для $p = 10$ имеем

$$m = 10^n a_n + 10^{n-1} a_{n-1} + \dots + 10a_1 + a_0. \quad (1.51)$$

1. Заметим, что $10 \equiv 1 \pmod{9}$. Тогда $10^k \equiv 1 \pmod{9}, k \in \mathbb{N}$. Из (1.51) получаем признак делимости на 9:

$$m \equiv a_n + a_{n-1} + \dots + a_1 + a_0 \pmod{9}.$$

2. Очевидно, что $10 \equiv -1 \pmod{11}$. Тогда $10^k \equiv (-1)^k \pmod{11}$ для $k \in \mathbb{N}$. Получаем признак делимости на 11:

$$m \equiv a_0 - a_1 + a_2 + \dots + (-1)^n a_n \pmod{11}.$$

Теорема 1.30. Если модуль m — составное число, то \mathbb{Z}_m не является полем.

Доказательство. Пусть

$$m = p_1 p_2, \quad 1 < p_1, p_2 < m. \quad (1.52)$$

Будем считать, что $P_1, P_2 \in \mathbb{Z}_m$ — классы вычетов, которым принадлежат p_1, p_2 . Тогда, в силу (1.52):

$$P_1 P_2 = 0, \quad P_1, P_2 \neq 0,$$

и элементы P_1, P_2 необратимы. Следовательно, \mathbb{Z}_m не поле. ■

Далее будет показано (см. следствие 1.17 к теореме 1.39), что в случае простого модуля \mathbb{Z}_m есть поле.

Пример 1.32. Рассмотрим таблицы арифметических действий для \mathbb{Z}_6 (таблица 1.20а и 1.20б).

В \mathbb{Z}_6 имеются делители нуля. Действительно: $2 \cdot 3 = 0$, $3 \cdot 4 = 0$. Для элементов $\{2, 3, 4\}$ нет обратных.

Таблица 1.20.

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

(a)

×	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

(b)

Замечание 1.25

Как легко заметить из последнего примера, таблица по сложению (таблица 1.20a) в \mathbb{Z}_6 является левосторонней матрицей. Этот факт нетрудно доказать для произвольного модуля.

Перейдем к делению сравнений.

Теорема 1.31. Пусть $ac \equiv bc(m)$, $D(c, m) = d$, тогда $a \equiv b \left(\frac{m}{d}\right)$.

Доказательство. Пусть $c = c_1d$, $m = m_1d$, тогда $ac - bc = lm$, таким образом,

$$l = \frac{ac - bc}{m} = \frac{(a - b)c_1}{m_1},$$

следовательно, $(a - b)c_1 \div m_1$, но $(c_1, m_1) = 1$, тем самым $(a - b) \div m_1$ или, что то же самое, $a \equiv b \left(\frac{m}{d}\right)$. ■

Следствие 1.12. Рассмотрим два важных предельных случая теоремы 1.31:

1) если $m \div c$, тогда из $ac \equiv bc(m)$ следует $a \equiv b \left(\frac{m}{c}\right)$;

2) если $D(m, c) = 1$, тогда из $ac \equiv bc(m)$ следует $a \equiv b(m)$.

Результаты, полученные в этом подразделе, обобщает следующая теорема.

Теорема 1.32. Пусть $F(a, b, \dots)$ — произвольная целая рациональная функция от $a, b, \dots \in \mathbb{Z}$, т. е.

$$F(a, b, \dots) = \sum_k C_k a^{\alpha_k} b^{\beta_k} \dots, \text{ где } C_k, \alpha_k, \beta_k, \dots \in \mathbb{Z}, \alpha_k, \beta_k, \dots > 0.$$

Если $a \equiv a_1(m)$, $b \equiv b_1(m)$, ..., то $F(a, b, \dots) \equiv F(a_1, b_1, \dots)(m)$.

В качестве простейшего приложения теоремы докажем следующее утверждение.

Утверждение 1.3. Квадрат всякого нечетного числа сравним с единицей по модулю 8.

Доказательство. Действительно, любое нечетное число можно представить в виде $4k \pm 1$. Тогда $(4k \pm 1)^2 = 16k^2 \pm 8k + 1 \equiv 1 \pmod{8}$. ■

С модульной арифметикой связано интересное свойство последовательности чисел Фибоначчи (раздел 1.2.12).

Теорема 1.33. Последовательность чисел Фибоначчи $\{\phi_i\}$, взятых по модулю m , является периодичной, причем период начинается с $\phi_1 = 1$.

Доказательство. Рассмотрим $m^2 + 1$ пар чисел Фибоначчи, взятых по модулю m :

$$(\phi_1, \phi_2), (\phi_2, \phi_3), \dots, (\phi_{m^2+1}, \phi_{m^2+2}).$$

По модулю m может быть только m^2 различных пар, поэтому среди этой последовательности найдутся как минимум две одинаковые пары. Это означает, что последовательность периодична.

Возьмем среди всех таких одинаковых пар две пары с наименьшими номерами. Обозначим их как (ϕ_n, ϕ_{n+1}) и (ϕ_k, ϕ_{k+1}) . Если $n > 1$, тогда предыдущие пары (ϕ_{n-1}, ϕ_n) и (ϕ_{k-1}, ϕ_k) по свойству чисел Фибоначчи тоже будут равны друг другу. Однако это противоречит выбору совпадающих пар с наименьшими номерами, таким образом, $n = 1$. ■

1.3.3. Функция Эйлера и ее свойства

Историческая справка

Леонард Эйлер (1707–1783) — самый плодовитый математик XVIII в., если только не всех времен. Опубликовано более двухсот томов его научных трудов, но это еще далеко не полное собрание его сочинений.

Л. Эйлер окончил университет в швейцарском городе Базеле, где изучал математику под руководством Иоганна Бернулли, а когда в 1725 г. сыновья Иоганна, Николай и Даниил, уехали в Петербург, Леонардо последовал за ними в недавно учрежденную по замыслу Петра I Российскую (Петербургскую) академию наук. Эйлер жил в России до 1741 г., затем работал в Берлинской академии под особым покрови-

тельством Фридриха II, а с 1766 г. и до конца жизни снова в России. Очевидно, что Эйлер по праву можно считать российским ученым, ибо основные годы его творчества прошли в Петербурге и он являлся академиком именно Петербургской академии наук под особым покровительством Екатерины II.

В 1766 г. ученый ослеп, но продолжал работать. Пользуясь своей феноменальной памятью, он диктовал статьи и книги, общее число которых достигло 886. Его работы посвящены анализу, алгебре, дискретной математике (теории графов), вариационному исчислению, функциям комплексного переменного, астрономии, гидравлике, теоретической механике, кораблестроению, артиллерии, теории музыки и т. д. Колоссальная продуктивность Эйлера в разных областях математики и в других областях науки была и остается поводом для изумления. Обозначения Эйлера почти современны, точнее, современная математическая символика почти эйлерова. Можно составить огромный список известных и важных математических открытий, приоритет в которых принадлежит великому ученому. Многие его идеи ждут разработки до настоящего времени.

Определение 1.23. Функция Эйлера $\varphi(m)$ ставит в соответствие каждому натуральному m количество чисел, меньших m и взаимно простых с m . Будем полагать $\varphi(1) = 1$.

Вычислять функцию Эйлера согласно определению затруднительно. Выведем общую формулу для произвольного аргумента.

Лемма 1.8. Пусть $m = p^n$, где p — простое число. Тогда

$$\varphi(m) = p^n \left(1 - \frac{1}{p} \right).$$

Доказательство. Из интервала $(0, p^n)$ на p делятся только числа, кратные p . Это $p, 2p, 3p, \dots, p^n - p$. Их всего $p^{n-1} - 1$. Остальные числа из данного интервала взаимно просты с p . Поэтому

$$\varphi(m) = \varphi(p^n) = p^n - 1 - (p^{n-1} - 1) = p^n \left(1 - \frac{1}{p} \right). \quad \blacksquare$$

Докажем вспомогательную теорему.

Теорема 1.34. Пусть x пробегает полную систему вычетов по модулю a , а y — соответственно по модулю b . При этом $D(a, b) = 1$. Тогда:

- 1) $z = ay + bx$ пробегает полную систему вычетов по модулю ab ;
- 2) $D(z, ab) = 1$ тогда и только тогда, когда $D(x, a) = 1$, $D(y, b) = 1$.

Доказательство. x принимает a значений, y принимает b значений, следовательно, z принимает ab значений. Покажем, что никакие два значения z несравнимы друг с другом по модулю ab . Пусть

$$z_1 = ay_1 + bx_1, \quad z_2 = ay_2 + bx_2, \quad ay_1 + bx_1 \equiv ay_2 + bx_2 \pmod{ab},$$

тогда согласно теореме 1.27:

$$ay_1 + bx_1 \equiv ay_2 + bx_2 \pmod{a}, \quad ay_1 + bx_1 \equiv ay_2 + bx_2 \pmod{b}.$$

По следствию 1.12 (п. 2) (так как $D(a, b) = 1$) имеем:

$$x_1 \equiv x_2 \pmod{a}, \quad y_1 \equiv y_2 \pmod{b}.$$

Но x_1, x_2 — из полной системы вычетов по модулю a , а y_1, y_2 — из полной системы вычетов по модулю b . Тогда $x_1 = x_2, y_1 = y_2, z_1 = z_2$. Таким образом, первая часть теоремы доказана.

Пусть $D(z, ab) = 1$, тогда очевидно, что

$$D(ay + bx, a) = 1, \quad D(ay + bx, b) = 1.$$

Следовательно, $z - ay = bx$ взаимно просто с a , а $z - bx = ay$ взаимно просто с b . Так как $D(a, b) = 1$, то $D(a, x) = 1, D(b, y) = 1$. Повторив последние рассуждения в обратном порядке, получаем достаточность п. 2. ■

Следствие 1.13.

$$D(a, b) = 1, \text{ тогда } \varphi(ab) = \varphi(a)\varphi(b). \quad (1.53)$$

Доказательство. В полной системе вычетов по модулю a существует $\varphi(a)$ значений x таких, что $D(a, x) = 1$. Аналогично для модуля b получаем $\varphi(b)$ значений y таких, что $D(b, y) = 1$.

Следовательно, всего имеется $\varphi(a)\varphi(b)$ значений z , взаимно простых с ab . Но значения z образуют полную систему вычетов по модулю ab и чисел, взаимно простых с ab , в ней $\varphi(ab)$. ■

Пример 1.33. Для иллюстрации доказательства следствия 1.13 составим таблицу 1.21 величин (x, y) при $a = 5, b = 8$.

Возможные значения для x — числа $0, 1, 2, 3, 4$, возможные значения для y — числа $0, 1, 2, 3, 4, 5, 6, 7$. Из них для x имеется четыре значения взаимно простых с a (так как $\varphi(5) = 4$).

Соответственно для y также есть четыре значения взаимно простых с b ($\varphi(8) = 4$). Эти значения выделены в таблице серым цветом, как и соответствующие им значения $z = ay + bx$.

Выделенные значения z дают 16 чисел, меньших 40 и взаимно простых с ним. Таким образом:

$$\varphi(40) = \varphi(5)\varphi(8) = 4 \cdot 4 = 16.$$

Таблица 1.21.

x	y							
	0	1	2	3	4	5	6	7
0	0	5	10	15	20	25	30	35
1	8	13	18	23	28	33	38	3
2	16	21	26	31	36	1	6	11
3	24	29	34	39	4	9	14	19
4	32	37	2	7	12	17	22	27

Замечание 1.26

Равенство (1.53) легко обобщить на случай нескольких попарно взаимно простых сомножителей.

Следствие 1.14. Пусть $m \in \mathbb{Z}$. Согласно (1.6) m представимо в виде

$m = p^\alpha q^\beta \dots$, где p, q, \dots — простые числа, $\alpha, \beta, \dots \geq 1$. Тогда

$$\varphi(m) = p^{\alpha-1}(p-1)q^{\beta-1}(q-1) \dots = m \left(1 - \frac{1}{p}\right) \left(1 - \frac{1}{q}\right) \dots \quad (1.54)$$

Доказательство. Так как p^α, q^β, \dots попарно взаимно просты, то, применяя последовательно следствие 1.13 и лемму 1.8, получаем требуемое. ■

Пример 1.34.

$$\varphi(49) = \varphi(7^2) = 7^2 - 7 = 42,$$

$$\varphi(30) = \varphi(2 \cdot 3 \cdot 5) = \varphi(2)\varphi(3)\varphi(5) = (2-1)(3-1)(5-1) = 8,$$

$$\varphi(60) = 60 \left(1 - \frac{1}{2}\right) \left(1 - \frac{1}{3}\right) \left(1 - \frac{1}{5}\right) = 16.$$

Рассмотрим важное применение формулы (1.54).

Теорема 1.35 (формула Гаусса). Пусть d пробегает все делители числа m . Тогда $m = \sum_{d|m} \varphi(d)^1$.

Доказательство. Пусть m представимо в виде (1.6). Тогда для любого делителя d справедливо представление

$$d = p^\chi q^\lambda \dots, \text{ где } 0 \leq \chi \leq \alpha, 0 \leq \lambda \leq \beta, \dots$$

¹ Индекс $d|m$ у знака \sum означает суммирование по всем делителям числа m .

Рассмотрим произведение

$$[1 + \varphi(p) + \varphi(p^2) + \dots + \varphi(p^\alpha)][1 + \varphi(q) + \varphi(q^2) + \dots + \varphi(q^\beta)] \dots$$

Перемножив все скобки, имеем

$$\sum_{x, \lambda, \dots} \varphi(p^x) \varphi(q^\lambda) \dots = \sum_{x, \lambda, \dots} \varphi(p^x q^\lambda \dots) = \sum_d \varphi(d).$$

С другой стороны, согласно лемме 1.8 каждую скобку можно представить в виде

$$1 + \varphi(p) + \varphi(p^2) + \dots + \varphi(p^\alpha) = 1 + (p-1) + (p^2-p) + \dots + (p^\alpha - p^{\alpha-1}) = p^\alpha.$$

Перемножив, получим $p^\alpha q^\beta \dots = m$. ■

Пример 1.35.

$$\begin{aligned} \sum_{d|30} \varphi(d) &= \varphi(1) + \varphi(2) + \varphi(3) + \varphi(5) + \varphi(6) + \varphi(10) + \varphi(15) + \varphi(30) = \\ &= 1 + 1 + 2 + 4 + 2 + 4 + 8 + 8 = 30. \end{aligned}$$

Функция Эйлера обладает еще одним интересным свойством.

Теорема 1.36. Если $n \in \mathbb{N}$ и $n > 2$, то $\varphi(n)$ — четное число.

Доказательство. Так как $n > 2$, то возможны два случая.

$$n = \begin{cases} 2^k n_1, & k > 1, \\ p^s n_2, & s \geq 1, p - \text{ простое число и } p \geq 3. \end{cases}$$

В обоих случаях четность $\varphi(n)$ очевидным образом следует из (1.54). ■

Для функции Эйлера существуют задачи, не разрешенные и по сей день.

Гипотеза (Кармайкл, 1907). Если посмотреть на последовательность значений функции Эйлера: $\{1, 1, 2, 2, 4, 2, 6, 4, 6, 4, \dots\}$, можно заметить, что она содержит много повторяющихся чисел.

Гипотеза Кармайкла состоит в том, что не существует такого значения n , которое функция Эйлера принимала бы только один раз.

Введем важное определение.

Определение 1.24. Функцию $f: \mathbb{R} \rightarrow \mathbb{R}$ называют *мультипликативной* если:

1) функция f определена всюду на \mathbb{N} и существует $a \in \mathbb{N}$ такой, что $f(a) \neq 0$;

2) для любых взаимно простых натуральных чисел a_1 и a_2 выполняется $f(a_1 a_2) = f(a_1) f(a_2)$.

Очевидно, что в силу следствия 1.13 к теореме 1.34 функция Эйлера является мультипликативной функцией, но далеко не единственной.

Рассмотрим еще один важный пример мультипликативной функции.

Определение 1.25. *Функция Мебиуса $\mu(m)$ определяется следующим образом.*

1. Полагаем $\mu(1) = 1$.
2. Пусть число $m \in \mathbb{N}$ представлено в каноническом виде (1.6):

$$m = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}, \quad \alpha_i \geq 1, \quad i \in 1 : k. \quad (1.55)$$

Тогда

$$\mu(m) = \begin{cases} 0, & \text{если } \max_{i \in 1:k} \{\alpha_i\} > 1, \\ (-1)^k & \text{в противном случае.} \end{cases}$$

Таким образом, если число m делится на квадрат натурального числа, отличный от единицы, то $\mu(m) = 0$.

Пример 1.36.

$$\mu(30) = \mu(2 \cdot 3 \cdot 5) = (-1)^3 = -1, \quad \mu(60) = \mu(2^2 \cdot 3 \cdot 5) = 0.$$

Упражнение 1.6. Покажите, что функция Мебиуса мультипликативна.

Для функции Мебиуса можно получить результат, аналогичный теореме 1.35.

Теорема 1.37. Справедливо тождество

$$\sum_{d|m} \mu(d) = \begin{cases} 0, & \text{если } m \geq 2, \\ 1 & \text{в противном случае.} \end{cases}$$

Доказательство. Если $m = 1$, то $\mu(m) = 1$ по определению. Пусть $m \geq 2$ представлено в виде (1.55). Тогда

$$\begin{aligned} \sum_{d|m} \mu(d) &= \mu(1) + [\mu(p_1) + \cdots + \mu(p_k)] + [\mu(p_1 p_2) + \cdots + \mu(p_{k-1} p_k)] + \\ &\quad + [\mu(p_1 p_2 p_3) + \cdots + \mu(p_{k-2} p_{k-1} p_k)] + \cdots + \mu(p_1 p_2 \cdots p_k) = \\ &= 1 - k + C_k^2 + \cdots + (-1)^s C_k^s + \cdots + (-1)^k = \sum_{s=0}^k (-1)^s C_k^s = 0. \end{aligned} \quad (1.56)$$

■

Замечание 1.27

Подробно биномиальные коэффициенты C_k^s будут рассмотрены в разделе 1.6.3. Последнее равенство в (1.56) будет доказано в следствии 1.22 и теореме 1.62.

Установим еще одно свойство функции Эйлера.

Теорема 1.38 (Эйлер — Ферма). Пусть $D(a, m) = 1$, тогда

$$a^{\varphi(m)} \equiv 1 \pmod{m}.$$

Доказательство. Пусть $D(a, m) = 1$, $\varphi(m) = k$. Тогда классов вычетов, взаимно простых с m , будет k . Пусть $\{a_1, a_2, \dots, a_k\}$ — представители этих классов. Тогда $\{aa_1, aa_2, \dots, aa_k\}$ также взаимно просты с m .

Предположим, что $aa_i \equiv aa_j \pmod{m}$, тогда $a_i \equiv a_j \pmod{m}$. Следовательно, числа aa_i также представители всех классов вычетов, взаимно простых с m . Тогда каждое aa_i сравнимо с одним и только одним a_j , т. е.

$$aa_1 \equiv a_\alpha \pmod{m}, \quad aa_2 \equiv a_\beta \pmod{m}, \dots, \quad aa_\mu \equiv a_\theta \pmod{m}, \quad \text{где } 1 \leq \alpha, \beta, \theta \leq k.$$

Перемножив последние k сравнений, имеем $a_1 \cdots a_k a^k \equiv a_\alpha \cdots a_\theta \pmod{m}$. Заметим, что $a_\alpha, \dots, a_\theta$ — это те же числа, что и a_1, \dots, a_k , только в другом порядке (см. пример 1.37). После сокращения (в силу $D(a_i, m) = 1$) получаем нужное сравнение. ■

Пример 1.37. Проиллюстрируем доказательство теоремы 1.38. Пусть $a = 4, m = 7$. Тогда

$$\begin{aligned} 1 * 4 &\equiv 4 \pmod{7}, & 2 * 4 &\equiv 1 \pmod{7}, & 3 * 4 &\equiv 5 \pmod{7}, \\ 4 * 4 &\equiv 2 \pmod{7}, & 5 * 4 &\equiv 6 \pmod{7}, & 6 * 4 &\equiv 3 \pmod{7}. \end{aligned} \quad (1.57)$$

На рисунке 1.3 приведена графическая иллюстрация соотношений (1.57).

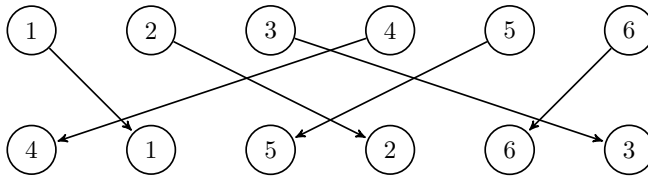


Рис. 1.3.

Следствие 1.15 («малая» теорема Ферма). Если p — простое число, то

$$a^p \equiv a \pmod{p}.$$

 **Замечание 1.28**

Этот частный случай установил Пьер Ферма в XVII в., однако доказательства не привел. Первое известное доказательство принадлежит Готфриду Вильгельму Лейбницу, но эта работа не была опубликована.

Полностью теорема 1.38 была доказана Леонардом Эйлером. А идея приведенного здесь доказательства принадлежит шотландскому математику Джеймсу Айвори и датируется 1806 г.

Следствие 1.16. Для простого числа p справедливо

$$(a + b)^p \equiv a^p + b^p \pmod{p}.$$

Доказательство. Достаточно заметить, что

$$(a + b)^p \equiv a + b \pmod{p}, \quad a^p \equiv a \pmod{p} \quad \text{и} \quad b^p \equiv b \pmod{p}.$$

■

Пример 1.38. Найдём остаток от деления $25^{11^{35}}$ на 34.

Вычислим $\varphi(34) = \varphi(2 \cdot 17) = \varphi(2) \cdot \varphi(17) = 1 \cdot 16 = 16$. Так как $D(25, 34) = 1$, то согласно теореме 1.38 (Эйлера — Ферма):

$$25^{16} \equiv 1 \pmod{34}.$$

Далее имеем $11^{35} = 16k + b$, где b — неизвестный остаток. Если число b известно, то в \mathbb{Z}_{34} справедливо

$$25^{11^{35}} = 25^{16k+b} = 25^{16k} \cdot 25^b = (25^{16})^k \cdot 25^b = 25^b.$$

Найдём теперь остаток b . Так как

$$D(11, 16) = 1 \quad \text{и} \quad \varphi(16) = \varphi(2^4) = 2^4 - 2^3 = 8,$$

то согласно теореме 1.38:

$$11^8 \equiv 1 \pmod{16}.$$

Тогда в \mathbb{Z}_{16} справедливо

$$b = 11^{35} = 11^{32} \cdot 11^3 = (11^8)^4 \cdot 11^3 = 1 \cdot 11^3 = 1331 = 3.$$

Решаем теперь исходную задачу. Имеем в \mathbb{Z}_{34} :

$$25^{11^{35}} = 25^3 \pmod{34} = 15625 = 19 \pmod{34}.$$

1.3.4. Линейные сравнения

Определение 1.26. *Линейным сравнением*¹ называют уравнение вида:

$$ax \equiv b \pmod{m}, \text{ где } a, b, x, m \in \mathbb{Z}. \quad (1.58)$$

В качестве решений уравнения (1.58) рассмотрим классы вычетов по модулю m , тогда сравнение может иметь не более m решений. Установим критерий разрешимости сравнения (1.58).

Теорема 1.39. Справедливо следующее:

- 1) пусть $d = D(a, m)$. Сравнение (1.58) разрешимо тогда и только тогда, когда $d \mid b$;
- 2) при этом существует d классов решений

$$\left\{ x_0, x_0 + \frac{m}{d}, x_0 + 2\frac{m}{d}, \dots, x_0 + (d-1)\frac{m}{d} \right\}. \quad (1.59)$$

Доказательство. Очевидно, что разрешимость сравнения (1.58) эквивалентна разрешимости диофантового уравнения $ax - my = b$. Применяя теорему 1.21, устанавливаем п. 1.

Далее, все решения соответствующего диофантового уравнения (а нас интересует только неизвестное x) описываются согласно (1.44) следующим образом: $x = x_0 - \frac{m}{d}t$. Различных по модулю m среди них ровно d , т. е. п. 2 установлен. ■

Следствие 1.17. Если p является простым числом, то сравнение $ax \equiv b \pmod{p}$ всегда имеет единственное решение (кроме случая $p \mid a$).

Таким образом (см. теорему 1.30), в случае простого модуля для любого ненулевого $a \in \mathbb{Z}_p$ существует обратный элемент a^{-1} , т. е. \mathbb{Z}_p является полем. Такие конечные поля называют *полями Галуа* и обозначают $GF(p)$

На основании теоремы 1.39 опишем алгоритм решения линейного сравнения.

Алгоритм 1.17. Решение линейного сравнения

Алгоритмом 1.16 решаем диофантово уравнение $ax - my = b$

$d \leftarrow D(a, m)$

if $d \mid b$ **then**

$x_0 \leftarrow$ любое решение диофантова уравнения

Согласно теореме 1.39 получаем все решения сравнения (1.58) в виде (1.59):

¹ По аналогии с линейными уравнениями в элементарной алгебре.

```

for  $i \leftarrow 1, d$  do
     $x_0 \leftarrow x_0 + m/d$ 
end for
else
    сравнение (1.58) не имеет решений.
end if

```

Пример 1.39. Решить сравнение $378x \equiv 90 \pmod{120}$.
Следуя теореме 1.39, решим диофантово уравнение

$$378x - 120y = 90. \quad (1.60)$$

Воспользуемся алгоритмом 1.16. Запишем протокол работы расширенного алгоритма Евклида для чисел 378 и 120 (таблица 1.22).

Таблица 1.22.

	378	120	18	12	6	0
q			3	6	1	2
x	1	0	1	-6	7	-20
y	0	1	-3	19	-22	63

Таким образом,

$$D(378, 120) = 6 = 378 \cdot 7 + 120 \cdot (-22) \Rightarrow 90 = 378 \cdot 105 - 120 \cdot 330.$$

Уравнение разрешимо, так как $90 : 6$, при этом $x_0 = 105$. Согласно теореме 1.39 существует $d = 6$ классов решений по модулю 120, получающихся из x_0 сдвигом на $m/d = 120/6 = 20$:

$$105, 125, 145, 165, 185, 205.$$

Замечание 1.29

В качестве x_0 в примере 1.39 вместо 105 можно взять любое решение диофантова уравнения (1.60):

$$x = x_0 + \frac{b}{d}t = 105 + 20t, \quad t \in \mathbb{Z},$$

например -15 . Добавляя последовательно 20, получаем те же шесть классов решений по модулю 120:

$$-15 \equiv 105(120), 5 \equiv 125(120), 25 \equiv 145(120), 45 \equiv 165(120), 65 \equiv 185(120), 85 \equiv 205(120).$$

Пример 1.40. Вычислить $\frac{46}{65}$ по модулю 71.

Найдем $x = 65^{-1} \pmod{71}$. Перепишем последнее равенство в виде $65x = 1 \pmod{71}$. Решим это сравнение.

Воспользуемся алгоритмом 1.16. Запишем протокол работы расширенного алгоритма Евклида для чисел 65 и 71 (таблица 1.23).

Таблица 1.23.

	65	71	65	6	5	1	0
q			0	1	10	1	5
x	1	0	1	-1	11	-12	71
y	0	1	0	1	-10	11	-65

Таким образом,

$$D(71, 65) = 1 = 65 \cdot (-12) + 71 \cdot 11 = 65 \cdot (-12) - 71 \cdot (-11).$$

Отсюда $x_1 = -12 \pmod{71} = 59 = 65^{-1}$. Вычисляем

$$46 \cdot 65^{-1} \pmod{71} = 46 \cdot 59 \pmod{71} = 2714 \pmod{71} = 16 \pmod{71}.$$

Рассмотрим еще один способ решения сравнения (1.58), принадлежащий Л. Эйлеру.

Будем считать, что числа a и m взаимно просты. К этой ситуации всегда можно привести сравнение, разделив a , b и m на $D(a, m)$. Согласно теореме 1.38:

$$a^{\varphi(m)} \equiv 1 \pmod{m}, \quad a^{\varphi(m)}b \equiv b \pmod{m}, \quad a(a^{\varphi(m)-1}b) \equiv b \pmod{m}.$$

Тогда

$$x^* = (a^{\varphi(m)-1}b) \pmod{m} \tag{1.61}$$

является решением (1.58).

Замечание 1.30

Недостаток такого подхода в том, что число a нужно возводить в степень $\varphi(m)$. Вычисления упрощаются, если возводить в степень в кольце \mathbb{Z}_m (см. замечание 1.9 к алгоритму 1.11).

Пример 1.41. Решить сравнение

$$11x \equiv 15 \pmod{24}.$$

Здесь $D(11, 24) = 1$, а $\varphi(24) = 8$. Согласно (1.61) нужно вычислить 11^7 .

Имеем (в Z_{34}):

$$11^2 = 1 \implies 11^6 = 1 \implies 11^7 = 11 \implies \text{в } Z_{34} \quad x^* = 11 \cdot 15 = -3.$$

В случае простого модуля решение (а оно в этом случае единственно) сравнения (1.58) можно получить в явном виде. Справедлива теорема.

Теорема 1.40. Пусть p — простое число, $0 < a < p$. Тогда сравнение $ax \equiv b \pmod{p}$ имеет решение:

$$x \equiv b(-1)^{a-1} \frac{C_p^a}{p} \pmod{p}. \quad (1.62)$$

Доказательство. Запишем набор из a тривиальных сравнений:

$$p-1 \equiv -1 \pmod{p}, \quad p-2 \equiv -2 \pmod{p}, \quad \dots, \quad p-(a-1) \equiv -(a-1) \pmod{p}, \quad b \equiv b \pmod{p}.$$

Перемножив эти a сравнений, получим

$$b(-1)^{a-1}(p-1)(p-2) \dots (p-(a-1)) \equiv b \times 1 \times 2 \dots \times (a-1) \pmod{p}. \quad (1.63)$$

Домножим и разделим левую часть сравнения (1.63) на $a!$. Получим

$$b(-1)^{a-1} a! \frac{(p-1)(p-2) \dots (p-(a-1))}{a!} \equiv b \times 1 \times 2 \dots \times (a-1) \pmod{p}. \quad (1.64)$$

Так как по условию теоремы $0 < a < p$ и p — простое число, то

$$D(1 \times 2 \dots \times (a-1), p) = 1.$$

Разделим сравнение (1.64) почленно на $(a-1)!$. Согласно следствию 1.12 к теореме 1.31 получаем требуемое. ■

Пример 1.42. Решим сравнение $8x \equiv 2 \pmod{13}$. Вычисляем решение согласно (1.62):

$$x \equiv 2(-1)^7 \frac{C_{13}^8}{13} (13) \equiv -\frac{2}{13} C_{13}^8 (13) \equiv -198 (13) \equiv 10 (13).$$

С помощью линейных сравнений установим критерий простоты числа.

Теорема 1.41 (Вильсон). Число $p \in \mathbb{Z}$ является простым тогда и только тогда, когда $(p-1)! \equiv -1 \pmod{p}$.

Доказательство. Пусть p — простое число. Не умаляя общности, считаем, что $p > 3$. Тогда для любого $a \in \mathbb{Z}$, не делящегося на p , сравнение

$$ax \equiv 1 \pmod{p} \quad (1.65)$$

имеет единственный класс решений $x \equiv b(p)$, где b — некоторый вычет из $\{1, \dots, p-1\}$. Заметим, что $b \neq 0$, в противном случае если $x \equiv 0(p)$, то $p \mid x$.

Рассмотрим все пары $\{a, b\}$, взаимно простые с p , из той же системы вычетов, что и решения (1.65) такие, что $ab \equiv 1(p)$.

Пусть $a = b$. Имеем $a^2 \equiv 1(p)$, тогда $(a+1)(a-1) \div p$, следовательно, $a \equiv \pm 1(p)$, т. е. $a = 1$ или $a = -1$.

Тогда для пар $\{a, b\}$ таких, что $2 \leq a, b \leq p-2$, справедливо $a \neq b$. Таких пар всего $(p-3)/2$.

Перемножив $(p-3)/2$ сравнения вида $ab \equiv 1(\text{mod } p)$, получим

$$2 \cdot 3 \cdot \dots \cdot (p-2) \equiv 1 \pmod{p}.$$

Домножив на тривиальное сравнение $p-1 \equiv -1(\text{mod } p)$, получим критерий Вильсона.

Пусть теперь p — составное число. Тогда $D((p-1)!, p) > 1$, следовательно, $(p-1)! + 1$ не может делиться на p . ■

Историческая справка

А. Вильсон (1714–1786) — шотландский астроном и математик, профессор астрономии в Глазго.

Теорема 1.41 впервые была сформулирована арабским ученым Ибн аль-Хайсамом около 1000 г. Первое доказательство опубликовано Лагранжем в 1771 г. Есть мнение, что Лейбниц знал о результате еще столетием раньше, но никогда не публиковал его.

1.3.5. Китайская теорема об остатках

Перейдем к системам сравнений первой степени. Рассмотрим важный частный случай, когда модули сравнений системы попарно взаимно просты.

Теорема 1.42 (китайская теорема об остатках). Система сравнений

$$x \equiv c_1(m_1), \quad x \equiv c_2(m_2), \quad \dots, \quad x \equiv c_k(m_k), \quad (1.66)$$

где $D(m_i, m_j) = 1$ для $i \neq j$ имеет единственное решение (по модулю $m_1 m_2 \dots m_k$).

Доказательство. Введем обозначения:

$$M = \prod_{i=1}^k m_i, \quad M_l = \frac{M}{m_l}, \quad l \in 1 : k.$$

Так как $D(M_l, m_l) = 1$ для всех $l \in 1 \dots k$, то согласно теореме 1.39 каждое сравнение $M_l x \equiv 1(m_l)$ имеет единственное решение (x_l) .

Тогда легко построить решение системы (1.66):

$$x^* \equiv \sum_{l=1}^k M_l x_l c_l \pmod{M}. \quad (1.67)$$

Подставив x^* в каждое сравнение (1.66), убеждаемся, что (1.67) — решение системы. Подставим для примера x^* в первое уравнение (1.66). Тогда

$$M_2 x_2 c_2 + M_3 x_3 c_3 + \dots + M_k x_k c_k \equiv 0 \pmod{m_1},$$

но $M_1 x_1 \equiv 1 \pmod{m_1}$, следовательно, $M_1 x_1 c_1 \equiv c_1 \pmod{m_1}$ и $x^* \equiv c_1 \pmod{m_1}$.

Покажем, что решение x^* единственно по модулю M . Предположим, что x — другое решение системы (1.66). Тогда выполнены сравнения

$$x^* - x \equiv 0 \pmod{m_1}, \dots, x^* - x \equiv 0 \pmod{m_k},$$

откуда

$$(x^* - x) : m_1, (x^* - x) : m_2, \dots, (x^* - x) : m_k.$$

По следствию к теореме 1.6 получаем, что

$$(x^* - x) : \prod_{i=1}^k m_i, \quad x^* \equiv x \pmod{M}.$$

Последнее сравнение и завершает доказательство. ■

Определение 1.27. Набор c_1, \dots, c_k называют *китайским кодом* числа x^* .

Историческая справка

Теорема 1.42 впервые была описана в трактате китайского математика Сунь Цзы «Математическое руководство» предположительно в III в. н. э. О самом авторе ничего не известно, кроме того, что он является автором этой книги; годы его жизни устанавливались историками науки на основе анализа текста трактата.

Пример 1.43. Найдем наименьшее натуральное число x , удовлетворяющее условиям:

$$x \equiv 2 \pmod{29}, \quad x \equiv 12 \pmod{20}, \quad x \equiv 20 \pmod{23}, \quad x \equiv 4 \pmod{13}.$$

Согласно обозначениям теоремы 1.66 имеем:

$$M = m_1 m_2 m_3 m_4 = 29 \cdot 20 \cdot 23 \cdot 13 = 173\,420,$$

$$M_1 = m_2 m_3 m_4 = 20 \cdot 23 \cdot 13 = 5\,980,$$

$$M_2 = m_1 m_3 m_4 = 29 \cdot 23 \cdot 13 = 8\,671,$$

$$M_3 = m_1 m_2 m_4 = 29 \cdot 20 \cdot 13 = 7\,540,$$

$$M_4 = m_1 m_2 m_3 = 29 \cdot 20 \cdot 23 = 13\,340.$$

Решим сравнение

$$5\,980x_1 \equiv 1 \pmod{29}.$$

Воспользуемся алгоритмом 1.16. Запишем протокол работы расширенного алгоритма Евклида для чисел 5 980 и 29 (таблица 1.24).

Таблица 1.24.

	5 980	29	6	5	1	0
q			206	4	1	5
x	1	0	1	-4	5	-29
y	0	1	-206	825	-1 031	5 980

Таким образом,

$$D(5\,980, 29) = 1 = 5\,980 \cdot 5 + 29 \cdot (-1\,031).$$

Откуда $x_1 = 5$, $M_1 x_1 = 29\,900$.

Решим сравнение

$$8\,671x_2 \equiv 1 \pmod{20}.$$

Применив алгоритм 1.16, запишем протокол работы расширенного алгоритма Евклида для чисел 8 671 и 20 (таблица 1.25).

Таким образом,

$$D(8\,671, 20) = 1 = 8\,671 \cdot (-9) + 20 \cdot 3\,920.$$

Откуда $x_2 = -9 = 11(20)$, $M_2 x_2 = 95\,381$.

Таблица 1.25.

	8 671	20	11	9	2	1	0
q			433	1	1	4	2
x	1	0	1	-1	2	-9	20
y	0	1	-433	434	-867	3 902	-8 671

Решим сравнение

$$7\,540x_3 \equiv 1 \pmod{23}.$$

Используя алгоритм 1.16, запишем протокол работы расширенного

алгоритма Евклида для чисел 7 540 и 23 (таблица 1.26).

Таблица 1.26.

	7 540	23	19	1	4	1	0
q			327	1	4	1	3
x	1	0	1	-1	5	-6	23
y	0	1	-327	328	-1 639	1 967	-7 540

Таким образом,

$$D(7\,540, 23) = 1 = 7\,540 \cdot (-6) + 23 \cdot 1\,967.$$

Откуда $x_3 = -6 = 17(23)$, $M_3 x_3 = 128\,180$.

Решим сравнение

$$13\,340 x_4 \equiv 1 \pmod{13}.$$

Применив алгоритм 1.16, запишем протокол работы расширенного алгоритма Евклида для чисел 13 340 и 13 (таблица 1.27).

Таблица 1.27.

	13 340	13	2	1	0
q			1 026	6	2
x	1	0	1	-6	13
y	0	1	-1 026	6 157	-13 340

Таким образом,

$$D(13\,340, 13) = 1 = 13\,340 \cdot (-6) + 13 \cdot 6\,157.$$

Откуда $x_4 = -6 = 7(13)$, $M_4 x_4 = 93\,380$.

Находим решение x^* :

$$\begin{aligned} x^* &= (M_1 x_1 c_1 + M_2 x_2 c_2 + M_3 x_3 c_3 + M_4 x_4 c_4) \pmod{M} = \\ &= (29\,900 \cdot 2 + 95\,381 \cdot 12 + 128\,180 \cdot 20 + 93\,380 \cdot 4) \pmod{173\,420} = \\ &= 152\,832. \end{aligned}$$

Замечание 1.31

Основную трудоемкость представляет вычисление значений $M_1 x_1, \dots, M_k x_k$. Однако для фиксированной системы модулей m_1, \dots, m_k они вычисляются только один раз и не зависят от китайского кода самого числа.

1.3.6. Система остаточных классов

Рассмотренная выше китайская теорема об остатках (теорема 1.42) является основой одной из «непозиционных систем счисления», так называемой системой остаточных классов¹ или «модулярной арифметикой».

Современные исследования по параллельным вычислениям показали, что в рамках обычной позиционной системы счисления значительного ускорения выполнения арифметических операций добиться невозможно. Это объясняется тем, что в таких системах значение разряда любого числа (кроме младшего) зависит не только от значения одноименных операндов, но и от всех предшествующих разрядов, т. е. позиционные системы обладают *последовательной структурой*.

Сегодня все большее внимание привлекают системы счисления, обладающие способностями к параллельной обработке информации. Такими возможностями обладает система остаточных классов, которая позволяют реализовать идею распараллеливания операций на уровне выполнения элементарных арифметических действий.

В системе остаточных классов согласно теореме 1.42 каждое число в диапазоне $0 \leq N \leq M - 1$ однозначно представляется вектором своего китайского кода (c_1, \dots, c_k) — см. определение 1.27.

Набор модулей (p_1, \dots, p_k) называют *базисом* системы. Представление самого числа в системе обычно записывают следующим образом:

$$N = (c_1 | \dots | c_k)_{(p_1 | \dots | p_k)}.$$

Выполнение арифметических операций в RNS производится поразрядно, *независимо* по каждому из модулей, что позволяет распараллелить алгоритмы при выполнении арифметических операций. Это избавляет от необходимости «занимать» или «переносить» единицу старшего разряда.

$$X = (x_1 | \dots | x_k)_{(p_1 | \dots | p_k)}, \quad Y = (y_1 | \dots | y_k)_{(p_1 | \dots | p_k)}.$$

Тогда

$$\begin{aligned} X + Y &= ((x_1 + y_1) \bmod p_1 | \dots | (x_k + y_k) \bmod p_k)_{(p_1 | \dots | p_k)}, \\ XY &= (x_1 y_1 \bmod p_1 | \dots | x_k y_k \bmod p_k)_{(p_1 | \dots | p_k)}. \end{aligned} \quad (1.68)$$

Систему используемых модулей (p_1, \dots, p_k) подбирают под конкретную задачу. Например, для представления 32-битных чисел достаточно взять систему модулей: $(7, 11, 13, 17, 19, 23, 29, 31)$ — все модули простые, следо-

¹ Оригинальное английское название — Residue Number System (RNS).

вательно, попарно взаимнопросты и

$$M = \prod_{i=1}^k p_i = 6685349671 > 4294967296 = 2^{32}.$$

Каждый из модулей не превышает 5 бит, таким образом поразрядные операции сложения и умножения будут производиться над 5-битными числами.

Пример 1.44. Рассмотрим арифметические действия с числами 13 и 11 в системе остаточных классов с базисом $(3, 7, 8)$. Тогда согласно (1.68) имеем:

$$\begin{aligned} 13 &= (1|6|5)_{(3|7|8)}, \quad 11 = (2|4|3)_{(3|7|8)}, \\ 13 + 11 &= (0|3|0)_{(3|7|8)}, \quad 13 \cdot 11 = (2|3|7)_{(3|7|8)}. \end{aligned}$$

В настоящее время система остаточных классов активно применяется в цифровой обработке сигналов, криптографии, обработке изображений и т. д.

Из недостатков RNS можно отметить следующие:

- а) ограниченный диапазон представления чисел;
- б) сложные алгоритмы деления;
- в) трудности в обнаружении переполнения.

Для более детального изучения данной темы можно рекомендовать монографию [69] (на английском языке). В ней содержится обширная информация о применении RNS с большим количеством подробных примеров.

1.3.7. Применение теоремы Эйлера в криптографии. Система шифрования RSA

Эволюция телекоммуникаций привела к повышению роли алгоритмов шифровки информации и развитию нового направления в криптографии — науке о способах преобразования (шифрования) информации с целью ее защиты от незаконных пользователей. Центральным понятием этого направления является понятие односторонней функции.

Определение 1.28. Функцию f называют *односторонней*, если существует эффективный (полиномиальный) алгоритм вычисления ее значений $f(x)$, но для решения уравнения $f(x) = a$ (обращения функции f) эффективного алгоритма нет.

Несмотря на то что построенные на понятии односторонней функции криптографические системы уже работают и понятия электронной подписи и электронных денег вошли в экономическую практику и закреплены законодательно, математического доказательства того, что эти функции действительно односторонние, пока не получено. Поэтому неизвестно, существуют ли односторонние функции.

Электронная подпись и электронные деньги основаны на так называемой функции с секретом (или ловушкой).

Определение 1.29. *Функцией с секретом K* называют функцию f_K , зависящую от параметра K и обладающую тремя свойствами:

- 1) при любом K существует эффективный (полиномиальный) алгоритм вычисления f_K ;
- 2) при неизвестном K не существует эффективного алгоритма обращения f_K ;
- 3) при известном K существует эффективный алгоритм обращения f_K .

Наиболее известной и популярной односторонней функцией является функция $f(x) = x^e \bmod m$, лежащая в основе шифра RSA (Rivest, Shamir, Adleman). Рассмотрим эту систему шифрования более подробно.

Историческая справка

Основы криптографии с открытыми ключами были выдвинуты Уитфилдом Диффи и Мартином Хеллманом. Идея состояла в том, что ключи можно использовать парами — ключ шифрования и ключ дешифрования, и что невозможно получить один ключ из другого.

Диффи и Хеллман впервые представили этот подход в [73] (1976), а спустя год появился первый алгоритм шифрования — RSA, основанный на этом подходе.

Еще до выхода из печати статья авторов системы RSA была послана известному популяризатору математики Мартину Гарднеру, который в 1977 г. в журнале Scientific American опубликовал статью [76], посвященную этой системе шифрования. В русском переводе название статьи Гарднера звучит так: «Новый вид шифра, на расшифровку которого потребуются миллионы лет».

Именно эта статья Гарднера сыграла важнейшую роль в распространении информации об RSA, привлекла к криптографии внимание широких кругов неспециалистов и фактически способствовала бурному прогрессу этой области, произошедшему в последующие 20 лет.

Будем считать, что шифруемые сообщения представлены (закодированы) длинными целыми числами.

Пусть m и e — натуральные числа. Функция f , осуществляющая шифрование, определяется следующим образом:

$$f(x) = x^e \bmod m.$$

Для дешифрования сообщения нужно решить сравнение

$$x^e \equiv a \pmod{m}.$$

Для возведения x в степень e (в кольце \mathbb{Z}_m) есть эффективный алгоритм 1.11. Для решения же обратной задачи — нахождения x по e и a — такого алгоритма предположительно нет. Поэтому f — односторонняя функция.

Оказывается, при некоторых условиях, налагаемых на m и e , функцию f можно использовать как функцию с секретом. Найдем эти условия.

Пусть число e взаимно просто с $\varphi(m)$, где φ — функция Эйлера (см. определение 1.23), а d — решение сравнения $de \equiv 1 \pmod{\varphi(m)}$. Тогда

$$de = 1 + k\varphi(m), \quad k \in \mathbb{Z}, \quad 1 \leq d < \varphi(m).$$

Так как $D(e, \varphi(m)) = 1$, то согласно теореме 1.39 существует единственный класс решений $d \pmod{\varphi(m)}$.

Авторы шифра RSA предложили выбирать число m в виде произведения двух простых множителей p и q , примерно одинаковых по значению. Тогда

$$\varphi(m) = \varphi(pq) = \varphi(p) \cdot \varphi(q) = (p-1)(q-1)$$

и условия взаимной простоты e с $\varphi(m)$ сводятся к условиям взаимной простоты e с $p-1$ и с $q-1$.

Покажем, что число d может служить секретом.

Теорема 1.43. Для любого $x \in \mathbb{Z}_m$, $x^{ed} \equiv x \pmod{m}$.

Доказательство. В силу выбора чисел d, e и m

$$de = 1 + k \cdot \varphi(m) = 1 + k(p-1)(q-1)$$

при некотором k . Если $x \not\equiv 0 \pmod{p}$, то по следствию 1.15 имеем:

$$x^{ed} \equiv x \left(x^{p-1}\right)^{k(q-1)} \equiv x \cdot 1^{k(q-1)} \equiv x \pmod{p}.$$

Если $x \equiv 0 \pmod{p}$, то сравнение $x^{ed} \equiv x \pmod{p}$, очевидно, выполняется. Таким образом,

$$\forall x \in \mathbb{Z}_m, \quad x^{ed} \equiv x \pmod{p}.$$

Аналогично получаем

$$\forall x \in \mathbb{Z}_m, \quad x^{ed} \equiv x \pmod{q}.$$

Применяя пункт 4 теоремы 1.27, получаем требуемое. ■

Итак, лицо, заинтересованное в организации шифрованной переписки с помощью шифра RSA, выбирает два достаточно больших простых числа p и q и перемножает их: $m = pq$. Затем выбирается число e , взаимно простое с $p - 1$ и $q - 1$; вычисляется $\varphi(m)$ и d .

Числа m и e публикуются, а число d держится в секрете. Теперь любой может отправить шифрованное сообщение лицу, опубликовавшему m и e , расшифровать которое сможет только последний.

Замечание 1.32

Для вычисления функции $f(x) = x^e \bmod m$ достаточно знать лишь числа e и m . Именно они составляют открытый ключ для шифрования. А вот для вычисления обратной функции требуется знать число d , оно и является «секретом».

Казалось бы, ничего не стоит, зная число m , разложить его на простые сомножители, вычислить затем значение $\varphi(m)$ и определить нужное число d . Все шаги этого вычисления могут быть реализованы достаточно быстро, за исключением первого. Именно разложение числа на простые множители и составляет наиболее трудоемкую часть вычислений.

В теории чисел, несмотря на многолетнюю ее историю и интенсивные поиски в течение последних 20 лет, эффективный алгоритм разложения натуральных чисел на множители так и не найден. Конечно, можно, перебирая все простые числа до \sqrt{m} и деля на них m , найти требуемое разложение. Но, учитывая, что количество простых чисел в этом промежутке, асимптотически равно

$$\frac{2\sqrt{m}}{\ln m},$$

находим, что при m , записываемом 100 десятичными цифрами, найдется не менее $4 \cdot 10^{42}$ простых чисел, на которые придется делить m при разложении его на множители.

Очень грубые прикидки показывают, что компьютеру, выполняющему миллион делений в секунду, для разложения числа $m > 10^{99}$ таким способом на простые сомножители потребуется не менее чем 10^{35} лет. Известны и более эффективные способы разложения целых чисел на множители, чем перебор простых делителей, но и они работают очень медленно.

Для иллюстрации своего метода авторы RSA зашифровали фразу «The magic words are squeamish ossifrage», предварительно заменив ее символы наборами из двух десятичных цифр: 00 — пробел, 01 — a, 02 — b, ..., 26 — z.

При этом m было равно числу ..., которое получилось перемножением простых чисел, записываемых соответственно 64 и 65 десятичными знаками. Число e взяли равным 9007.

Это сообщение ждало своей расшифровки 17 лет и ожидание завершилось в 1994 г., когда D. Atkins, M. Graff, A. K. Lenstra и P. C. Leyland сообщили о дешифровке фразы. Для дешифровки сообщения пришлось искать разложение на множители 129-значного десятичного числа. Это было непросто, поскольку для разложения числа на множители до сих пор не найдено эффективного алгоритма. Все известные алгоритмы, в том числе и использованный в решении метод квадратичного решета, основаны на методе полного перебора.

Интересно, что нахождение этого разложения стало, по-видимому, первым серьезным сетевым проектом. В работе, возглавляемой четверьмя авторами проекта и

продолжавшейся 220 дней, участвовало 660 человек и 1600 компьютеров, объединенных сетью Интернет.

При практическом применении шифров с открытым ключом в настоящее время используются простые числа до 400 и более цифр в десятичной записи. В результате вскрыть этот шифр оказывается невозможно (в обозримое время).

Рассмотрим работу алгоритма RSA на примере.

Пример 1.45.

✓ **Генерация ключей.** Выбираем пару простых чисел $p = 3$, $q = 11$.

Тогда

$$m = pq = 3 \cdot 11 = 33; \quad (p - 1)(q - 1) = 2 \cdot 10 = 20.$$

Выбираем число e , взаимно простое с $\varphi(m) = 20$, например $e = 7$. Тогда открытая часть ключа $(m, e) = (33, 7)$. Вычислим закрытую часть ключа d . Имеем

$$7d \equiv 1 \pmod{20}, \quad \text{тогда } d = 3 (7 \cdot 3 = 21 \pmod{20} = 1).$$

✓ **Шифрование сообщения.** Зашифруем слово LIFE. Шифруем сообщение S .

$$S = [S_1, S_2, S_3, S_4], \quad \text{где } S_1 = L = 12, S_2 = I = 9, S_3 = F = 6, S_4 = E = 5.$$

Тогда кодированное сообщение имеет вид $C = [C_1, C_2, C_3, C_4]$, где

$$C_1 = S_1^7 \pmod{33} = 12^7 \pmod{33} = 12,$$

$$C_2 = S_2^7 \pmod{33} = 9^7 \pmod{33} = 15,$$

$$C_3 = S_3^7 \pmod{33} = 6^7 \pmod{33} = 30,$$

$$C_4 = S_4^7 \pmod{33} = 5^7 \pmod{33} = 14.$$

✓ **Декодирование сообщения.** Декодируем принятое сообщение. Имеем:

$$D_1 = C_1^3 \pmod{33} = 12^3 \pmod{33} = 1728 \pmod{33} = 12,$$

$$D_2 = C_2^3 \pmod{33} = 15^3 \pmod{33} = 3375 \pmod{33} = 9,$$

$$D_3 = C_3^3 \pmod{33} = 30^3 \pmod{33} = 27000 \pmod{33} = 6,$$

$$D_4 = C_4^3 \pmod{33} = 14^3 \pmod{33} = 2744 \pmod{33} = 5.$$

Рассмотри еще один пример работы алгоритма RSA.

Пример 1.46. Русское слово из четырех букв, закодированное с помощью алгоритма RSA с открытым ключом $e = 19$, $m = 46$. Зашифрованное сообщение имеет вид: (5, 12, 16, 26). Буквам русского алфавита соответствуют числа в диапазоне от 2 до 32 (исключены буквы «Ё» и «Ъ»).

Нужно подобрать закрытую часть ключа и дешифровать исходное слово.

Определяем закрытую часть ключа d :

$$\varphi(m) = \varphi(2 \cdot 23) = 22, 19d \equiv 1 \pmod{22}, d = 7 = (1, 1, 1)_2.$$

Используя алгоритм 1.11 быстрого возведения в степень по модулю, получаем:

$$\begin{aligned} 1) c &= 1, & 2) c &= 1^2 \cdot 5 = 5, & 3) c &= 5^2 \cdot 5 = 125(46) = 33, \\ 4) c &= 33^2 \cdot 5 = 5445(46) = 17. \end{aligned}$$

Первая буква сообщения «П» (17). Далее, аналогично:

$$\begin{aligned} 1) c &= 1, & 2) c &= 1^2 \cdot 12 = 12, & 3) c &= 12^2 \cdot 12 = 1728(46) = 26, \\ 4) c &= 26^2 \cdot 12 = 8112(46) = 16. \end{aligned}$$

Вторая буква сообщения «О» (16). Далее:

$$\begin{aligned} 1) c &= 1, & 2) c &= 1^2 \cdot 16 = 16, & 3) c &= 16^2 \cdot 16 = 4096(46) = 2, \\ 4) c &= 2^2 \cdot 16 = 64(46) = 18. \end{aligned}$$

Третья буква сообщения «Р» (18). Определяем последнюю букву слова:

$$\begin{aligned} 1) c &= 1, & 2) c &= 1^2 \cdot 26 = 26, & 3) c &= 26^2 \cdot 26 = 17576(46) = 4, \\ 4) c &= 4^2 \cdot 26 = 416(46) = 2. \end{aligned}$$

Четвертая буква сообщения «А» (2). Слово «пора».

1.3.8. Простейшие атаки на систему RSA

Появление новой системы шифрования RSA привело к многочисленным попыткам ее взлома, она проверяется «на прочность» многими исследователями с момента первой публикации. Хотя четверть века исследований привели к большому числу крайне интересных атак, ни одна из них не представляет большой угрозы. Они скорее иллюстрируют опасность неправильного использования RSA.

Рассмотрим несколько простейших вариантов атак на RSA, которые позволяют взломать ее, не используя прямого разложения модуля m на произведение двух простых чисел.

1.3.8.1. Схема с нотариусом

- **Начальные условия.** Известны открытая часть ключа нотариуса (m, e) .
- **Задача.** Получить подпись нотариуса на сообщении S .
- **Решение.** Противник выбирает *затемняющий множитель* $r \in \mathbb{Z}_m$ и $r \neq 0$, затем вычисляет

$$\widehat{S} = r^e S \pmod{m} \quad (1.69)$$

и отправляет это сообщение на подпись нотариусу. Если нотариус подписывает это сообщение:

$$\widehat{C} = \widehat{S}^d \pmod{m} = r^{ed} S^d \pmod{m} = r S^d \pmod{m},$$

то противник, легко сняв затемняющий множитель r , получает подпись сообщения $S^d \pmod{m}$.

Замечание 1.33

Из-за использования затемняющего множителя этот метод иногда называют ослеплением (*blinding*). В данном контексте ослепление является способом атаки на RSA, однако, как будет показано ниже, это полезное свойство для предоставления анонимности электронных платежей: «электронные деньги» позволяют покупать товар, не указывая конкретно покупателя.

1.3.8.2. Метод бесключевого чтения RSA, или циклическая атака

Рассмотрим еще один пример алгоритма такого рода — метод бесключевого чтения RSA. Этот алгоритм известен еще под названием *циклическая атака*.

- **Начальные условия.** Противнику стали известны открытая часть ключа (m, e) и закодированное сообщение C .
- **Задача.** Восстановить исходное сообщение S .
- **Решение.** Противник подбирает число j , для которого выполняется следующее соотношение:

$$C^{e^j} \pmod{m} = C.$$

Таким образом, противник просто проводит j раз шифрование на открытом ключе перехваченного закодированного сообщения C . Это выглядит следующим образом:

$$(C^e)^e \dots)^e \pmod{m} = C^{e^j} \pmod{m}.$$

Найдя такое j , противник вычисляет $C^{e^{j-1}} \pmod{m}$ (т. е. $j - 1$ раз повторяет операцию шифрования) — это значение и есть исходное сообщение S . Это следует из того, что

$$C^{e^j} \pmod{m} = (C^{e^{j-1}})^e \pmod{m} = C = S^e \pmod{m}.$$

Пример 1.47. Рассмотрим следующие параметры шифра RSA [80]:

$$p = 983, \quad q = 563, \quad m = pq = 553\,429.$$

Выберем для открытой части ключа $e = 49$, а сообщение $S = 123\,456$. Тогда закодированное (перехваченное) сообщение C имеет вид:

$$C = S^{49} \pmod{m} = 1\,603.$$

Проводим циклическую атаку (таблица 1.28).

Таблица 1.28.

j	$C^{e^j} \pmod{m}$
...
7	$C^{49^7} \pmod{m} = 85\,978$
8	$C^{49^8} \pmod{m} = 123\,456$
9	$C^{49^9} \pmod{m} = 1\,603 = C$

Откуда получаем

$$S = C^{49^8} \pmod{m} = 123\,456.$$

1.3.9. Другие примеры использования RSA

1.3.9.1. Электронная подпись

Электронная подпись существенно отличается от привычной. Ее смысл — так зашифровать передаваемое сообщение, чтобы расшифровать его можно было только с помощью открытой части кода, принадлежащей

лицу, «поставившему подпись». Таким образом проверяется принадлежность письма предполагаемому автору.

Как следует из теоремы 1.43, операции зашифровки и расшифровки различаются только показателем степени и, следовательно, коммутируют:

$$x = (x^e)^d \bmod m = x^{ed} \bmod m = x^{de} \bmod m = (x^d)^e \bmod m = x.$$

На этом свойстве и основан прием, называемый *электронной подписью*.

1. Отправитель кодирует сообщение x своим закрытым ключом d : $y = x^d \bmod m$ и посылает адресату пару $\{x, y\}$, т. е. подписанное сообщение.

2. Адресат декодирует подпись сообщения с помощью открытой части кода e и m , т. е. вычисляет $\bar{x} = y^e \bmod m$. Если $\bar{x} = x$, то сообщение правильное, если же $\bar{x} \neq x$, то сообщение было искажено при передаче или фальсифицировано.

1.3.9.2. Электронные деньги

Всюду ниже под электронными деньгами будем понимать электронные платежные средства, обеспечивающие *неотслеживаемость*. Понятие *неотслеживаемости*, по-видимому, не может быть формализовано и будет пояснено на конкретном примере протокола.

Для работы с электронными деньгами разрабатываются специальные криптографические протоколы, которые называют *протоколами электронных платежей*. В таком протоколе задействованы три участника, которых будем называть: *банк, покупатель и продавец*. Покупатель и продавец, каждый в отдельности, имеют счет в банке, и покупатель желает заплатить продавцу за товар или услугу.

В платежной системе используются три основные транзакции:

- 1) снятие со счета;
- 2) платеж;
- 3) депозит.

В транзакции снятия со счета покупатель получает подписанную банком электронную банкноту на затребованную сумму. При этом счет покупателя уменьшается на эту сумму.

В транзакции платежа покупатель передает банкноту продавцу и указывает сумму платежа. Продавец, в свою очередь, передает эту информацию банку, который проверяет подлинность банкноты. Если банкнота подлинная, банк проверяет, не была ли она потрачена ранее. Если нет, то банк

заносит банкноту в специальный регистр, зачисляет требуемую сумму на счет продавца, уведомляет продавца об этом и, если достоинство банкноты выше, чем сумма платежа, возвращает покупателю «сдачу» (через продавца).

С помощью транзакции депозита покупатель может положить «сдачу» на свой счет в банке.

Замечание 1.34

Транзакция депозита в данном примере не рассматривается. Подробнее можно ознакомиться в [10].

Безопасность банка основывается на невозможности подделать банковскую подпись для создания фальшивой банкноты или более общим образом на невозможности, получив набор подлинных электронных банкнот, подделать подпись еще хотя бы для одной банкноты.

Для неотслеживаемости покупателя необходимо, чтобы банк, получив банкноту в транзакции платежа, не мог установить, кому она была выдана. То же относится и к «сдаче». Это, казалось бы, парадоксальное требование удовлетворяется с помощью схемы так называемой затемненной (слепой) подписи: в транзакции снятия со счета банк подписывает не банкноту, а некоторую «абракадабру», из которой покупатель восстанавливает подписанную банкноту. Таким образом, неотслеживаемость гарантируется тем, что банк просто не знает, что именно он подписал.

Рассмотрим простейший вариант платежной системы, в которой используется затемненная подпись, соответствующая схеме подписи RSA.

Банк выбирает открытую (m, e) и закрытую d части ключа и публикует открытую часть, а также некоторую одностороннюю функцию $f : \mathbb{Z}_m \rightarrow \mathbb{Z}_m$, назначение которой станет ясно из дальнейшего. Ключ $\{(m, e), d\}$ используется банком исключительно для создания электронных банкнот, т. е. устанавливается соглашение о том, что электронной подписи, сгенерированной на этом ключе, соответствует электронная банкнота достоинством, скажем, в 1 сантик.

✓ **Транзакция снятия со счета.** Покупатель выбирает случайное число $n \in \mathbb{Z}_m$ и вычисляет $f(n)$. Ему нужно получить подпись банка на этой банкноте, т. е. значение $f^d(n) \bmod m$. Но просто послать значение $f(n)$ банку покупатель не может, поскольку для снятия денег со счета он должен идентифицировать себя. Поэтому если банк получает $f(n)$, он в дальнейшем всегда узнает данную банкноту и неотслеживаемость будет потеряна.

Решение проблемы состоит в использовании затемненной подписи («схема с нотариусом» — (1.69)): покупатель выбирает случайное число $r \in \mathbb{Z}_m, r \neq 0$, затем вычисляет $f(n)r^e \bmod m$ и посылает это значение банку. Как уже отмечалось ранее, множитель r^e часто называют *затемняю-*

щим множителем. Банк подписывает это число, т. е. вычисляет значение

$$(f(n)r^e)^d \equiv f(n)^d r \pmod{m},$$

и возвращает его покупателю. Покупатель делением на r «снимает» затемняющий множитель и получает подписанную банкноту $(n, f^d(n) \pmod{m})$.

✓ **Транзакция платежа.** Покупатель передает продавцу электронную банкноту $(n, f^d(n) \pmod{m})$. В принципе, продавец может проверить подлинность любой банкноты (n, s) самостоятельно. Для этого достаточно вычислить $f(n)$ и проверить, что при этом $f(n) \equiv s^e \pmod{m}$. Но дело в том, что электронные банкноты, как и любую другую информацию, представленную в электронной форме, легко копировать. Поэтому нечестный покупатель может заплатить одной и той же электронной банкнотой многократно.

Для предотвращения подобного злоупотребления продавец передает банкноту на проверку банку. Банк проверяет по специальному регистру, не была ли эта банкнота потрачена ранее, и если нет, то зачисляет 1 сантик на счет продавца и уведомляет его об этом.

Безопасность банка в этой системе электронных платежей основывается на вере в стойкость схемы электронной подписи RSA. Применение функции f в этой конструкции необходимо ввиду известного свойства мультипликативности схемы RSA: если s_1 и s_2 — подписи для m_1 и m_2 соответственно, то

$$s_1 s_2 \equiv m_1^d m_2^d \pmod{m}$$

есть подпись для $m_1 m_2$. Поэтому если бы в системе электронных платежей использовались банкноты вида $(n, n^d \pmod{m})$, то из двух подлинных банкнот всегда можно было бы изготовить третью.

Неотслеживаемость клиентов в данной системе абсолютна. Все, что остается у банка от транзакции снятия со счета $f^d(n)r \pmod{m}$, — это значение, которое благодаря затемняющему множителю r представляет собой просто случайное число из \mathbb{Z}_m . Поэтому у банка нет информации о том, какую именно банкноту он выдал данному клиенту.

В рассмотренном случае банк выдает банкноты только достоинством в 1 сантик и все платежи должны быть кратны этой величине. Оказывается, можно реализовать и более гибкую систему. С подробностями можно ознакомиться в уже указанной монографии [10].

Задачи и вопросы

1. Пусть $\varphi(m)$ — функция Эйлера (см. определение 1.23).
 - а) вычислите $\varphi(6^{10}27^7)$;

б) найдите три различных решения уравнения $\varphi(x) = 4^9$.

2. Используя теорему Эйлера — Ферма, найдите остаток от деления $43^{3^{53}}$ на 60.

3. Найдите наименьшее натуральное число x , удовлетворяющее условиям:

$$x \equiv 11 \pmod{12}, \quad x \equiv 20 \pmod{31}, \quad x \equiv 1 \pmod{29}, \quad x \equiv 9 \pmod{13}.$$

4. Дано кольцо вычетов \mathbb{Z}_{15} и несколько его элементов: $\{2, 3, 7, 9, 0, 6, 12\}$.

а) Выпишите из указанных все делители нуля.

б) Сколько из перечисленных элементов обратимы? Найдите обратный к наименьшему из них.

5. Вычислите $\frac{3}{37}$ в кольце вычетов по модулю 87.

6. Если в игре с разноцветными шариками в мешочке располагать их по 15 в ряд, в мешочке останутся 4 шарика. Если шарика располагать по 8 в ряд, то 3 шарика останутся в мешочке. Если каждый ряд будет содержать 23 шарика, то 10 шариков останутся в мешочке. Какое наименьшее количество шариков первоначально могло находиться в мешочке? Какое было при этом количество рядов по 15, 8 и 23 шарика в каждом?

7. Приведите пример технологии распараллеливания арифметических операций: работа без переноса в следующий разряд.

8. Есть открытая часть ключа RSA: (e, m) . Зашифруйте число N .

9. На чем основана криптостойкость системы шифрования RSA?

10. Есть открытая часть ключа RSA: (e, m) и зашифрованное сообщение M . Предложите способ дешифровки сообщения M .

11. Пусть $m = 35$ и $e = 7$ — открытая часть ключа RSA. Найдите закрытую часть ключа d .

12. Поставьте электронную подпись на сообщении, заданном числом 27, используя код RSA с открытыми частями $m = 17 \cdot 23$ и $e = 125$.

13. Русское слово из четырех букв закодированного с помощью алгоритма RSA с открытым ключом $e = 19, m = 46$. Зашифрованное сообщение имеет вид: $(5, 12, 16, 26)$. Подберите закрытую часть ключа и дешифруйте исходное слово. Буквам русского алфавита соответствуют числа в диапазоне от 2 до 32 (исключены буквы «Ё» и «Ъ»).

1.4. Арифметика многочленов

1.4.1. Основные операции и свойства

Определение 1.30. Алгебраическое выражение с переменной x называют *многочленом над полем K* , если оно может быть представлено в виде

$$P_n(x) = p_n x^n + p_{n-1} x^{n-1} + \dots + p_1 x + p_0, \text{ где } p_i \in K. \quad (1.70)$$

При этом используются следующие обозначения:

- $K[x]$ — множество многочленов от переменной x над полем K ;
- $l(P_n(x))$ — старший коэффициент многочлена, $l(P_n(x)) = p_n \neq 0$;
- $\deg P_n(x)$ — степень многочлена, $\deg P_n(x) = n$.

Если в формуле (1.70) все p_i равны нулю, то такой многочлен называют *нулевым*. Степень нулевого многочлена не определена (иногда ее удобно полагать равной нулю, см. теорему 1.45).

Арифметические операции с многочленами в некотором смысле проще операций с натуральными числами, так как операции межразрядного переноса для многочленов отсутствуют.

Многочлен $P_n(x)$ степени n определяется упорядоченным набором своих $n + 1$ коэффициентов: $\{p_n, p_{n-1}, \dots, p_0\}$.

Замечание 1.35

Между многочленами и позиционными записями натуральных чисел есть очевидная аналогия, которая распространяется и на алгоритмы выполнения арифметических операций над многочленами.

В дальнейшем, когда это не вызывает разночтений, вместо $P_n(x)$ будем писать P_n или просто P .

Сложение и умножение многочлена на число осуществляется как сложение и вычитание векторов.

Для рассматриваемых в этом разделе алгоритмов введем следующие обозначения:

$$P_n(x) = p_n x^n + \dots + p_0, \quad Q_m(x) = q_m x^m + \dots + q_0, \quad n \geq m$$

— исходные многочлены,

$$S_n(x) = s_k x^k + \dots + s_0$$

— результат работы алгоритма.

Алгоритм 1.18. Сложение двух многочленов

```

for  $i \leftarrow 0, m$  do
   $s_i \leftarrow p_i + q_i$ 
end for
for  $i \leftarrow m + 1, n$  do
   $s_i \leftarrow p_i$ 
end for

```

Произведение многочленов P_n степени n и Q_m степени m вычисляется по формуле

$$S_{n+m} = P_n Q_m = \sum_{k=0}^{n+m} \left(\sum_{i+j=k} p_i q_j \right) x^k. \quad (1.71)$$

Для простоты вычислений выражение в правой части (1.71) удобно переписать в виде:

$$\sum_{k=0}^m \sum_{j=0}^k p_j q_{k-j} x^k + \sum_{k=m+1}^n \sum_{j=k-m}^k p_j q_{k-j} x^k + \sum_{k=n+1}^{n+m} \sum_{j=k-m}^n p_j q_{k-j} x^k. \quad (1.72)$$

Из (1.72) очевидно, что

$$l(PQ) = l(P)l(Q), \quad \deg PQ = \deg P + \deg Q.$$

Запишем алгоритм умножения многочленов на основе равенства (1.72).

Алгоритм 1.19. Умножение многочленов

```

for  $k \leftarrow 0, n + m$  do
   $s_k \leftarrow 0$ 
end for
for  $k \leftarrow 0, m$  do
  for  $j \leftarrow 0, k$  do
     $s_k \leftarrow s_k + p_j q_{k-j}$ 
  end for
end for
for  $k \leftarrow m + 1, n$  do
  for  $j \leftarrow k - m, k$  do
     $s_k \leftarrow s_k + p_j q_{k-j}$ 
  end for
end for
for  $k \leftarrow n + 1, n + m$  do
  for  $j \leftarrow k - m, n$  do
     $s_k \leftarrow s_k + p_j q_{k-j}$ 
  end for
end for

```

end for
end for

При таком определении сложения и умножения многочленов очевидна следующая теорема.

Теорема 1.44. Множество многочленов $K[x]$ есть ассоциативное коммутативное кольцо с единицей.

Упражнение 1.7. Покажите, что кольцо многочленов $\mathbb{R}[x]$ не является полем.

Для многочленов в $K[x]$ справедливо представление, аналогичное (1.1) для чисел из \mathbb{Z} .

Теорема 1.45. Пусть $A(x), B(x) \in K[x]$, тогда если $B(x) \neq \text{const}$, то существуют и единственны многочлены $Q(x)$ и $R(x)$ такие, что справедливо представление

$$A(x) = B(x)Q(x) + R(x), \quad 0 \leq \deg R < \deg B. \quad (1.73)$$

Здесь полагаем, что степень нулевого многочлена равна 0. Как и ранее для чисел, многочлен $Q(x)$ называют частным от деления $A(x)$ на $B(x)$, а $R(x)$ — соответственно остатком, $R(x) = \langle A(x) \rangle_{B(x)}$.

Доказательство. Существование многочленов Q и R в представлении (1.73) можно обосновать конструктивно. Построим для этого алгоритм 1.20, аналогичный делению чисел «уголком».

Алгоритм 1.20. Первый алгоритм деления многочленов

Инициализация:

$R \leftarrow A$

$Q \leftarrow 0$

// Вычисляем обратное значение старшего коэффициента многочлена делителя $l(B)^{-1}$

if $l(B) \neq 1$ **then**

$\alpha \leftarrow 1/l(B)$

else

$\alpha \leftarrow 1$

end if

while $\deg R \geq \deg B$ **do**

$T \leftarrow l(R) \cdot \alpha \cdot x^{\deg R - \deg B}$

$Q \leftarrow Q + T$

$R \leftarrow R - T \cdot B$

end while

 **Замечание 1.36**

Очевидно, что после каждого шага цикла в алгоритме 1.20 степень R уменьшается, но при этом равенство $A = BQ + R$ сохраняется. Когда степень R станет меньше степени B , алгоритм 1.20 свою работу заканчивает.

Единственность представления (1.73) докажем от противного.

Предположим, что $A = BQ_1 + R_1 = BQ_2 + R_2$. Тогда после вычитания равенств получим $B(Q_1 - Q_2) = R_2 - R_1$. Степень левой части не меньше $\deg B$, если только Q_1 не совпадает с Q_2 ; степень же правой части меньше $\deg B$, так как $\deg R_1 < \deg B$ и $\deg R_2 < \deg B$.

Итак, равенство будет справедливо только при $Q_1 = Q_2$ и, следовательно, $R_1 = R_2$. ■

Представление (1.73) для многочленов над конечными полями играет важную роль в криптографии и будет использоваться в дальнейшем в разделе 1.6.5.

Рассмотрим несколько примеров деления многочленов с остатком в \mathbb{Z}_m для различных простых m .

Пример 1.48. Используя алгоритм 1.20, найдем представление (1.73) для многочленов

$$A(x) = x^5 + 2x^3 + x^2 + 2, \quad B(x) = 2x^3 + 2x^2 + x + 2$$

над полем \mathbb{Z}_3 .

Для вычислений понадобятся таблицы арифметических действий для \mathbb{Z}_3 (таблица 1.29a и 1.29b).

Таблица 1.29.

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

(a)

×	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

(b)

 **Замечание 1.37**

Для сложения и вычитания в кольце вычетов \mathbb{Z}_m можно обойтись и без таблицы сложения, просто выполнив действия в \mathbb{Z} и взяв результат из наименьшей положительной системы вычетов (см. пример 1.30). Например, для \mathbb{Z}_3

$$1 - 2 = -1 = 2 \pmod{3}$$

1 $R = A$, $Q = 0$. Вычислим предварительно $1/l(B) = 2^{-1} = 2$.
Тогда

$$l(R)/l(B) = 2, T = 2x^{\deg R - \deg B} = 2x^2, Q = 2x^2, R = 2x^4 + 2.$$

Вычисление $R = R - TB$ покажем подробно:

$$\begin{array}{r} x^5 + \quad 2x^3 + x^2 \\ x^5 + x^4 + 2x^3 + x^2 \\ \hline 2x^4 \end{array} + 2 \left| \begin{array}{r} 2x^3 + 2x^2 + x + 2 \\ 2x^2 \end{array} \right.$$

2 $R = 2x^4 + 2$, $Q = 2x^2$, $\deg R - \deg B = 1$.

$$l(R)/l(B) = 2 \cdot 2 = 1, T = x^{\deg R - \deg B} = x, Q = 2x^2 + x, R = x^3 + 2x^2 + x + 2.$$

Вычисляем $R = R - TB$.

$$\begin{array}{r} 2x^4 \\ 2x^4 + 2x^3 + x^2 + 2x \\ \hline x^3 + 2x^2 + x + 2 \end{array} + 2 \left| \begin{array}{r} 2x^3 + 2x^2 + x + 2 \\ x \end{array} \right.$$

3 $R = x^3 + 2x^2 + x + 2$, $Q = 2x^2 + x + 2$, $\deg R - \deg B = 0$.

$$l(R)/l(B) = 1 \cdot 2 = 2, T = 2x^{\deg R - \deg B} = 2, Q = 2x^2 + x + 2, R = x^2 + 2x + 1.$$

Вычисляем $R = R - TB$.

$$\begin{array}{r} x^3 + 2x^2 + x + 2 \\ x^3 + x^2 + 2x + 1 \\ \hline x^2 + 2x + 1 \end{array} + 2 \left| \begin{array}{r} 2x^3 + 2x^2 + x + 2 \\ 2 \end{array} \right.$$

Окончательно получаем представление

$$x^5 + 2x^3 + x^2 + 2 = (2x^3 + 2x^2 + x + 2)(2x^2 + x + 2) + x^2 + 2x + 1.$$

Пример 1.49. Найдем представление (1.73) для многочленов

$$x^5 + x^4 + 1, \quad x^4 + x^2 + 1$$

над полем \mathbb{Z}_2

Все действия будем выполнять в \mathbb{Z}_2 . Поэтому нам потребуются таблицы арифметических действий (таблица 1.30a и 1.30b).

Таблица 1.30.

+	0	1
0	0	1
1	1	0

(a)

×	0	1
0	0	0
1	0	1

(b)

Заметим, что из таблицы 1.30а следует, что $1 - 1 = 1$. Вычисляем предварительно $1/l(B) = 1^{-1} = 1$. Далее имеем

$$\textcircled{1} \quad R = x^5 + x^4 + 1, \quad Q = 0, \quad \deg R - \deg B = 1.$$

$$l(R)/l(B) = 1 \cdot 1 = 1, \quad T = x^{\deg R - \deg B} = x, \quad Q = x.$$

Вычисляем $R = R - TB$

$$\begin{array}{r} x^5 + x^4 \qquad \qquad \qquad + 1 \quad \Big| \quad x^4 + x^2 + 1 \\ x^5 \qquad \qquad + x^3 \qquad + x \quad \Big| \quad x \\ \hline x^4 + x^3 \qquad \qquad + x + 1 \end{array}$$

$$\textcircled{2} \quad R = x^4 + x^3 + x + 1, \quad Q = x, \quad \deg R - \deg B = 0.$$

$$l(R)/l(B) = 1 \cdot 1 = 1, \quad T = x^{\deg R - \deg B} = 1, \quad Q = x + 1.$$

Вычисляем $R = R - TB$

$$\begin{array}{r} x^4 + x^3 \qquad \qquad + x + 1 \quad \Big| \quad x^4 + x^2 + 1 \\ x^4 \qquad \qquad + x^2 \qquad + 1 \quad \Big| \quad 1 \\ \hline x^3 + x^2 + x \end{array}$$

Окончательно получаем представление

$$x^5 + x^4 + 1 = (x^4 + x^2 + 1)(x + 1) + x^3 + x^2 + x.$$

Пример 1.50. Найдем представление (1.73) для многочленов

$$4x^5 + 4x^4 + x^3 + 4x^2 + x + 1, \quad 4x^3 + 2x^2 + 2x + 2$$

над полем \mathbb{Z}_5 .

Используем таблицы арифметических действий (таблица 1.31а и 1.31б).

Предварительно вычисляем $1/l(B) = 4^{-1} = 4$. Далее имеем

$$\textcircled{1} \quad R = 4x^5 + 4x^4 + x^3 + 4x^2 + x + 1, \quad Q = 0, \quad \deg R - \deg B = 2.$$

$$l(R)/l(B) = 4 \cdot 4 = 1, \quad T = x^{\deg R - \deg B} = x^2, \quad Q = x^2.$$

Таблица 1.31.

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

(a)

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

(b)

Вычисляем $R = R - TB$

$$\begin{array}{r} 4x^5 + 4x^4 + x^3 + 4x^2 + x + 1 \\ 4x^5 + 2x^4 + 2x^3 + 2x^2 \\ \hline 2x^4 + 4x^3 + 2x^2 + x + 1 \end{array} \left| \begin{array}{l} 4x^3 + 2x^2 + 2x + 2 \\ x^2 \end{array} \right.$$

$$\textcircled{2} \quad R = 2x^4 + 4x^3 + 2x^2 + x + 1, \quad Q = x^2, \quad \deg R - \deg B = 1.$$

$$l(R)/l(B) = 2 \cdot 4 = 3, \quad T = x^{\deg R - \deg B} = 3x, \quad Q = x^2 + 3x.$$

Вычисляем $R = R - TB$

$$\begin{array}{r} 2x^4 + 4x^3 + 2x^2 + x + 1 \\ 2x^4 + x^3 + x^2 + x \\ \hline 3x^3 + x^2 + 1 \end{array} \left| \begin{array}{l} 4x^3 + 2x^2 + 2x + 2 \\ 3x \end{array} \right.$$

$$\textcircled{3} \quad R = 3x^3 + x^2 + 1, \quad Q = x^2 + 3x, \quad \deg R - \deg B = 0.$$

$$l(R)/l(B) = 3 \cdot 4 = 2, \quad T = x^{\deg R - \deg B} = 2, \quad Q = x^2 + 3x + 2.$$

Вычисляем $R = R - TB$

$$\begin{array}{r} 3x^3 + x^2 + 1 \\ 3x^3 + 4x^2 + 4x + 4 \\ \hline 2x^2 + x + 2 \end{array} \left| \begin{array}{l} 4x^3 + 2x^2 + 2x + 2 \\ 3x \end{array} \right.$$


Окончательно получаем представление

$$4x^5 + 4x^4 + x^3 + 4x^2 + x + 1 = (4x^3 + 2x^2 + 2x + 2)(x^2 + 3x + 2) + 2x^2 + x + 2.$$

Запишем алгоритм деления многочленов в другом виде.

Для многочленов $A(x) = a_n x^n + \dots + a_0$ и $B(x) = b_m x^m + \dots + b_0$ требуется получить представление:

$$A(x) = B(x)(q_{n-m}x^{n-m} + \dots + q_0) + r_{m-1}x^{m-1} + \dots + r_0. \quad (1.74)$$

 **Замечание 1.38**

В отличие от алгоритма 1.20, коэффициенты многочлена остатка r_i будем хранить в коэффициентах многочлена делимого a_i .

Алгоритм 1.21. Второй алгоритм деления многочленов

Инициализация:

$$k \leftarrow \deg A - \deg B = n - m$$

// Вычисляем обратное значение старшего коэффициента многочлена делителя $l(B)^{-1}$

if $b_m \neq 1$ **then**

$$\alpha \leftarrow 1/b_m$$

else

$$\alpha \leftarrow 1$$

end if

while $k \geq 0$ **do**

$$q_k \leftarrow \alpha \cdot a_{m+k}$$

for $j \leftarrow m + k - 1, k$ **do**

// пересчитываем коэффициенты a_j

$$a_j \leftarrow a_j - q_k \cdot b_{j-k}$$

end for

end while

Пример 1.51. Используя алгоритм 1.21, разделим с остатком над полем \mathbb{R} многочлен $x^4 + x^3 - 3x^2 - 4x - 1$ на $x^3 + x^2 - x - 1$. Протокол работы алгоритма приведен в таблице 1.32. Имеем $n = 4, m = 3$. Тогда

Таблица 1.32.

k	q_k	Формула пересчета	a_4	a_3	a_2	a_1	a_0
			1	1	-3	-4	-1
1	$q_1 = a_4/b_3 = 1$	$a_j = a_j - b_{j-1}, j = 3, 2, 1$		0	-2	-3	
0	$q_0 = a_3/b_3 = 0$	$a_j = a_j, j = 2, 1, 0$			-2	-3	-1

Таким образом получаем представление:

$$x^4 + x^3 - 3x^2 - 4x - 1 = (x^3 + x^2 - x - 1)(x + 0) + (-2x^2 - 3x - 1).$$

1.4.2. Схема Горнера

Рассмотрим результат деления многочлена $P(x) \in K[x]$ на многочлен $Q(x) = x - \alpha \in K[x]$:

$$P(x) = (x - \alpha)Q(x) + R(x), \text{ где } \deg R(x) < \deg(x - \alpha) = 1,$$

т. е. $\deg R(x) = 0$ и $R(x) = \text{const} = r$.


 **Замечание 1.39**

| $\deg Q(x) = \deg P(x) - 1$.

Теорема 1.46 (Безу). Остаток от деления многочлена $P(x)$ на $x - \alpha$ равен значению многочлена $P(\alpha)$.

Доказательство. $P(x) = (x - \alpha)Q(x) + r$, следовательно, $P(\alpha) = r$. ■

Следствие 1.18. Многочлен $x - \alpha$ является делителем многочлена $P(x)$ тогда и только тогда, когда $P(\alpha) = 0$.

 **Замечание 1.40**

| Теорема Безу связывает два разных взгляда на многочлены — алгебраический (деление многочленов, остаток) и аналитический (вычисление значений многочлена как функции r).

Согласно теореме Безу в основу вычисления многочлена может быть положен алгоритм деления многочлена на двучлен. Этот алгоритм называют *схемой Горнера*. Рассмотрим его подробнее как частный случай алгоритма 1.21, когда $Q(x) = x - \alpha$.

Имеем $m = 1$, $b_1 = 1$, $b_0 = -\alpha$. Тогда получаем следующую схему

$$q_k = a_{k+1}, \quad a_k = a_k - a_{k+1}b_0 = a_k + a_{k+1}\alpha, \quad k = n - 1, \dots, 0. \quad (1.75)$$

При этом значение остатка находится в a_0 .

Схема (1.75) иногда может быть неудобна, так как в ней изменяются коэффициенты исходного многочлена $A_n(x)$.

Модифицируем схему (1.75) так, чтобы их сохранить. В представлении (1.74) коэффициенты частного будем хранить в r_{n-1}, \dots, r_0 , а остатка — в r_{-1} . Тогда схема (1.75) преобразуется к виду

$$r_{n-1} = a_n, \quad r_{k-1} = a_k - a_{k+1}b_0 = a_k - r_k b_0 = a_k + r_k \alpha, \quad k = n - 1, \dots, 0. \quad (1.76)$$

Обычно схемой Горнера называют именно схему (1.76).

Схема Горнера эквивалентна представлению многочлена $A(x)$ в виде

$$A(x) = a_n x^n + \dots + a_0 = (\dots((a_n x + a_{n-1})x + a_{n-2})x + \dots + a_1)x + a_0.$$

При ручных вычислениях промежуточные результаты схемы Горнера удобно помещать в следующей таблице:

a_n	a_{n-1}	a_{n-2}	\dots	a_1	a_0
r_{n-1}	r_{n-2}	r_{n-3}	\dots	r_0	$r_{-1} = s_0 = p(\alpha)$

Каждое следующее вычисление использует только значение α и числа, помещенные в соседних (левой и верхней) клетках таблицы:

\dots	a_k	\dots
r_k	$r_{k-1} = a_k + \alpha r_k$	\dots

Следствие 1.19. Схему Горнера можно применять многократно: к частному $Q(x)$ от деления $A(x)$ на $x - \alpha$, затем к получившемуся частному $Q_1(x)$ и т. д.

В результате получим коэффициенты разложения многочлена $P(x)$ по степеням $x - \alpha$ (от младших степеней к старшим):

$$A(x) = (x - \alpha)Q(x) + s_0 = (x - \alpha)[(x - \alpha)Q_1(x) + s_1] + s_0 = (x - \alpha)^2Q_1(x) + s_1(x - \alpha) + s_0 = \dots = s_n(x - \alpha)^n + s_{n-1}(x - \alpha)^{n-1} + \dots + s_1(x - \alpha) + s_0.$$

При выполнении вычислений вручную протокол вычислений удобно вести в «треугольной» таблице (таблица 1.33).

Таблица 1.33.

a_n	a_{n-1}	a_{n-2}	\dots	a_1	a_0
r_{n-1}	r_{n-2}	\dots	\dots	r_0	$r_{-1} = s_0$
c_{n-2}	\dots	\dots	\dots	$c_{-1} = s_1$	
\dots	\dots	\dots	\dots		
\dots	\dots	s_{n-2}			
\dots	s_{n-1}				
s_n					

Таблица заполняется слева направо и сверху вниз по правилу, указанному ранее.

Пример 1.52 (схема Горнера). Найдем остаток от деления многочлена

$$A(x) = x^4 + 2x^3 - 3x^2 + x - 5$$

на двучлен $x - 2$.

Согласно схеме Горнера имеем (таблица 1.34):

$$A(x) = x^4 + 2x^3 - 3x^2 + x - 5 = (x^3 + 4x^2 + 5x + 11)(x - 2) + 17.$$

Таблица 1.34.

a_4	a_3	a_2	a_1	a_0
1	2	-3	1	-5
1	4	5	11	17
r_3	r_2	r_1	r_0	s_0

Таблица 1.35.

1	2	-3	1	-5
1	4	5	11	17
1	6	17	45	
1	8	33		
1	10			
1				

Используя обобщенную схему Горнера, найдем разложение многочлена

$$A(x) = x^4 + 2x^3 - 3x^2 + x - 5$$

по степеням $x-2$. Последовательно применяя схему Горнера, получаем (таблица 1.35):

$$A(x) = x^4 + 2x^3 - 3x^2 + x - 5 = (x-2)^4 + 10(x-2)^3 + 33(x-2)^2 + 45(x-2) + 17.$$

Упражнение 1.8. Напишите алгоритм вычисления коэффициентов многочлена $S(x) : s_0, \dots, s_n$, полученного из многочлена $A(x)$ с коэффициентами a_n, a_{n-1}, \dots, a_0 разложением по степеням $x - \alpha$.

1.4.3. Алгоритм Евклида для многочленов. Линейное представление НОД

Алгоритм Евклида для многочленов не отличается от алгоритма деления чисел при соответствующей интерпретации входящих в него переменных и операций.

Сравним определения деления нацело, лежащие в основе алгоритма Евклида, для чисел и многочленов.

$a = bq + r$	$A(x) = B(x)Q(x) + R(x)$
$0 \leq r < b $	$0 \leq \deg R(x) < \deg B(x)$
q — частное, r — остаток	$Q(x)$ — частное, $R(x)$ — остаток

Видно, что определения отличаются лишь характеристиками (абсолютным значением числа, степенью многочлена), используемыми для нахождения остатка и частного.

Несмотря на то что частное и остаток при делении как целых чисел, так и многочленов определяются однозначно, выбор наибольшего общего делителя многочленов неоднозначен.

Это связано с тем, что НОД определяется как общий делитель наибольшей степени, а таких многочленов может быть бесконечно много.

Например, если $G(x) = D(A(x), B(x))$ для $A, B \in \mathbb{R}[x]$, то $\alpha G(x)$ при $\alpha \neq 0$ также будет наибольшим общим делителем этих многочленов. Можно было бы среди всех наибольших общих делителей выбрать многочлен с единичным старшим коэффициентом, тогда НОД будет определен единственным образом. Однако для вычислений это не всегда удобно.

Например, можно рассмотреть модификацию классического алгоритма Евклида с домножением промежуточных результатов на ненулевую константу, что позволит, находя НОД целочисленных многочленов, избежать дробных коэффициентов.

Пример 1.53. Найдем $D(2x^2 + x - 1, 3x^2 + 2x - 1)$.

- 1) $D(2x^2 + x - 1, 3x^2 + 2x - 1) = D(6x^2 + 3x + 3, 3x^2 + 2x - 1)$;
- 2) $D(6x^2 + 3x - 3, 3x^2 + 2x - 1) = D(-x - 1, 3x^2 + 2x - 1)$;
- 3) $D(-x - 1, 3x^2 + 2x - 1) = D(-x - 1, 0)$;
- 4) $D(-x - 1, 0) = D(x + 1, 0) = x + 1$.

Определение 1.31. Многочлены P и G называют *взаимно простыми*, если они не имеют отличных от константы общих делителей.

Для нахождения линейного представления наибольшего общего делителя многочленов можно использовать аналог расширенного алгоритма Евклида для чисел (алгоритм 1.15). Рассмотрим примеры.

Пример 1.54. С помощью расширенного алгоритма Евклида найдем $D(x^4 + x^3 - 3x^2 - 4x - 1, x^3 + x^2 - x - 1)$ и его линейное представление над полем \mathbb{R} . Протокол работы алгоритма запишем в виде таблицы 1.36:

Следовательно,

$$D(x^4 + x^3 - 3x^2 - 4x - 1, x^3 + x^2 - x - 1) = -\frac{3}{4}(x + 1).$$

Произведя нормировку (умножив на $-4/3$), получаем

$$D(x^4 + x^3 - 3x^2 - 4x - 1, x^3 + x^2 - x - 1) = x + 1.$$

Таблица 1.36.

	$x^4 + x^3 - 3x^2 - 4x - 1$	$x^3 + x^2 - x - 1$	$-2x^2 - 3x - 1$
$q(x)$			x
$u(x)$	1	0	1
$v(x)$	0	1	$-x$
	$-3/4(x + 1)$	0	
$q(x)$	$1/4(-2x + 1)$	$4/3(2x + 1)$	
$u(x)$	$-1/4(-2x + 1)$	$-4/3(x^2 - 1)$	
$v(x)$	$1/4(-2x^2 + x + 4)$	$4/3(x^3 - 3x - 1)$	

Таким образом:

$$(x^4 + x^3 - 3x^2 - 4x - 1) \left(\frac{x}{2} - \frac{1}{4} \right) + (x^3 + x^2 - x - 1) \left(-\frac{x^2}{2} + \frac{x}{4} + 1 \right) = -\frac{3}{4}(x+1).$$

Произведя нормировку (умножив на $-4/3$), получаем

$$(x^4 + x^3 - 3x^2 - 4x - 1) \left(\frac{-2x + 1}{3} \right) + (x^3 + x^2 - x - 1) \left(\frac{2x^2 - x - 4}{3} \right) = x + 1.$$

Пример 1.55. Найти $D(x^5 + x^4 + 1, x^4 + x^2 + 1)$ и его линейное представление над полем \mathbb{Z}_2 . Запишем протокол работы расширенного алгоритма Евклида (таблица 1.37).

Таблица 1.37.

	$x^5 + x^4 + 1$	$x^4 + x^2 + 1$	$x^3 + x^2 + x$	$x^2 + x + 1$	0
$q(x)$			$x + 1$	$x + 1$	x
$u(x)$	1	0	1	$x + 1$	$x^2 + x + 1$
$v(x)$	0	1	$x + 1$	x^2	$x^3 + x + 1$

Тогда

$$(x^5 + x^4 + 1)(x + 1) + (x^4 + x^2 + 1)(x^2) = x^2 + x + 1.$$

1.4.4. Интерполяционная формула Лагранжа

Аналогично алгоритму Евклида на многочлены переносится и китайская теорема об остатках.

Теорема 1.47. Рассмотрим систему сравнений

$$P(x) \equiv R_1(x) \pmod{M_1(x)}, \dots, P(x) \equiv R_n(x) \pmod{M_n(x)}.$$

Если многочлены $M_1(x), \dots, M_n(x)$ попарно взаимно просты, то

$$P(x) \equiv \sum_{i=1}^n C_i(x) D_i(x) R_i(x) \pmod{M(x)}, \text{ где}$$

$$M(x) = \prod_{i=1}^n M_i(x), C_i(x) = \frac{M(x)}{M_i(x)}, D_i(x) C_i(x) \equiv 1 \pmod{M_i(x)}.$$

Доказательство. Доказательство теоремы идентично доказательству аналогичной теоремы для чисел (см. теорему 1.42).

Решение сравнения $D_i(x) C_i(x) \equiv 1 \pmod{M_i(x)}$ осуществляется применением расширенного алгоритма Евклида к взаимно простым многочленам $C_i(x), M_i(x)$: $C_i(x)U(x) + M_i(x)V(x) = \text{const} \neq 0$, тогда

$$\frac{C_i(x)U(x)}{\text{const}} \equiv 1 \pmod{M_i(x)},$$

откуда

$$D_i(x) \equiv \frac{U(x)}{\text{const}} \pmod{M_i(x)}.$$

■

Теорема 1.48 (интерполяционная формула Лагранжа). Пусть $\deg P = n$ и известны значения многочлена $P(x)$ в $n + 1$ различных точках:

$$P(x_0) = y_0, P(x_1) = y_1, \dots, P(x_n) = y_n,$$

тогда

$$P(x) = \sum_{i=0}^n \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} y_i. \quad (1.77)$$

Доказательство. Интерпретируем задачу в терминах арифметики многочленов.

По теореме Безу $P(x_i) = y_i$ тогда и только тогда, когда $P(x) \equiv y_i \pmod{(x - x_i)}$.

Многочлены $M_i(x) = x - x_i$ попарно взаимно просты, и $P(x)$ можно найти применением китайской теоремы об остатках:

$$R_i(x) = y_i; M(x) = \prod_{k=0}^n (x - x_k); C_i(x) = \frac{M(x)}{M_i(x)} = \prod_{k \neq i} (x - x_k),$$

тогда $D_i(x)$ удовлетворяет уравнению

$$C_i(x)D_i(x) + M_i(x)B(x) = 1, \quad \deg D_i < \deg M_i = \deg(x - x_i) = 1, \quad (1.78)$$

откуда $\deg D_i = 0$, т. е. $D_i(x) = \text{const}$. Подставив в (1.78) $x = x_i$, получим

$$C_i(x_i)D_i + M_i(x_i)B(x_i) = 1,$$

но $M_i(x_i) = x_i - x_i = 0$, следовательно, $D_i = \frac{1}{C_i(x_i)}$. Отсюда по китайской теореме об остатках

$$P(x) \equiv \sum_{i=0}^n C_i(x)D_i(x)R_i(x) = \sum_{i=0}^n \frac{C_i(x)}{C_i(x_i)} y_i = \sum_{i=0}^n y_i \prod_{i \neq k} \left(\frac{x - x_i}{x_i - x_k} \right).$$

■

Пример 1.56. По формуле Лагранжа (1.77) найти интерполяционный многочлен $P(x)$, удовлетворяющий условиям:

$$P(-2) = 23, \quad P(1) = -1, \quad P(3) = 13.$$

Имеем

$$\begin{aligned} P(x) &= 23 \frac{(x-1)(x-3)}{(-2-1)(-2-3)} - \frac{(x+2)(x-3)}{(1+2)(1-3)} + 13 \frac{(x+2)(x-1)}{(3+2)(3-1)} = \\ &= \frac{23}{15}(x^2 - 4x + 3) + \frac{1}{6}(x^2 - x - 6) + \frac{13}{10}(x^2 + x - 2) = \\ &= 3x^2 - 5x + 1. \end{aligned}$$

1.4.5. Разложение многочлена на свободные от квадратов множители

Определение 1.32. Многочлен $P(x)$ из кольца $K[x]$ называют *неприводимым* (*примитивным*) над полем K , если не существует нетривиальных, т. е. отличных от константы, многочленов $P_1(x)$ и $P_2(x)$ из $K[x]$ таких, что $P(x) = P_1(x)P_2(x)$.

Пример 1.57. Многочлены $P(x) = x^2 + 1$, $Q(x) = x^2 + x + 2$ являются неприводимыми над полем \mathbb{R} , многочлены $T(x) = x^3 + 3x^2 + 9x + 6$ и $S(x) = x^4 + 2x^3 - 8x^2 - 2x + 3$ неприводимы над полем \mathbb{Q} (см. теорему 1.51).

Аналогично определению 1.6 для чисел можно ввести соответствующее определение для многочленов.

Определение 1.33. Многочлен $P(x)$ называется *свободным от квадратов*, если не существует многочлена $Q(x)$, $\deg Q > 0$ такого, что $Q^2(x) \mid P(x)$.

Представление многочлена $P(x) \in K[x]$ в виде

$$P(x) = P_1(x)P_2^2(x)P_3^3(x) \cdots P_n^n(x), \quad P_1, P_2, \dots, P_n \in K[x],$$

где $P_i(x)$ — произведение различных свободных от квадратов многочленов, причем P_i взаимно прост с P_j ($j \neq i$), называют разложением многочлена $P(x)$ на *свободные от квадратов множители* P_1, P_2, \dots, P_n .

Алгоритм разложения на свободные от квадратов множители основан на использовании алгоритма Евклида.

В данном подразделе будем рассматривать многочлены над полем вещественных чисел.

Лемма 1.9. Если $P = Q^k M$, где $P, Q, M \in \mathbb{R}[x]$, а Q неприводим в $\mathbb{R}[x]$ и взаимно прост с M , то производная P' может быть записана в виде $Q^{k-1} N$, где Q и N взаимно просты.

Доказательство.

$$P' = (Q^k M)' = k Q^{k-1} Q' M + Q^k M' = Q^{k-1} (kMQ' + QM').$$

Пусть $N = kMQ' + QM'$.

Докажем, что Q и N взаимно просты. Так как Q неприводим в $\mathbb{R}[x]$, то взаимная простота Q и N равносильна тому, что N не делится на Q . Если бы многочлен N делился на Q , то на Q делился бы и многочлен MQ' . Но Q' на Q не делится, так как $\deg Q' < \deg Q$, поэтому на Q должен делиться многочлен M , что противоречит условию. ■

Следствие 1.20. Если $P = Q_1^{k_1} \cdots Q_n^{k_n}$, где Q_1, \dots, Q_n неприводимы в $\mathbb{R}[x]$, то $P' = Q_1^{k_1-1} \cdots Q_n^{k_n-1} Q$, где Q взаимно прост с Q_1, \dots, Q_n .

Рассмотрим алгоритм разложения многочлена на свободные от квадратов множители.

Алгоритм 1.22. Разложение многочлена на свободные от квадратов множители

$j \leftarrow 1$

while $P \neq \text{const}$ **do**

$R \leftarrow D(P, P')$

$S \leftarrow \frac{P}{R}$

$M \leftarrow D(S, P')$

// $P = Q_1^{k_1} \cdots Q_n^{k_n}$

// $R = Q_1^{k_1-1} \cdots Q_n^{k_n-1}$

// $S = Q_1 \cdots Q_n$

// $M = Q_{i_1} \cdots Q_{i_s} \mid k_{i_p} > 1$

$$P_j \leftarrow \frac{S}{M} \quad // P_j = Q_{i_1} \cdots Q_{i_s} | k_{i_p} = 1$$

$$j \leftarrow j + 1$$

// повторяем алгоритм для многочлена, у которого степени неприводимых множителей уменьшены на 1

$$P \leftarrow R$$

end while

1.4.6. Разложение многочленов на неприводимые множители

Неприводимые многочлены являются аналогом простых чисел для многочленов. Возможность разложения многочлена на множители зависит от поля, над которым рассматривается многочлен, т. е. какому полю принадлежат коэффициенты.

Пример 1.58.

1) Многочлен $P(x) = x^2 - 2$ является неприводимым над \mathbb{Q} — полем рациональных чисел, но приводимым над \mathbb{R} — полем вещественных чисел:

$$x^2 - 2 = (x - \sqrt{2})(x + \sqrt{2}).$$

2) Многочлен $P(x) = x^4 + 1$ также является неприводимым над \mathbb{Q} — полем рациональных чисел, но приводимым над \mathbb{R} — полем вещественных чисел:

$$x^4 + 1 = x^4 + 2x^2 + 1 - 2x^2 = (x^2 + 1)^2 - (\sqrt{2}x)^2 = (x^2 - \sqrt{2}x + 1)(x^2 + \sqrt{2}x + 1).$$

3) Многочлен $P(x) = x^2 + 1$ является неприводимым над \mathbb{R} — полем вещественных чисел, но приводимым над \mathbb{C} — полем комплексных чисел:

$$x^2 + 1 = (x + i)(x - i).$$

4) Многочлен $P(x) = x^2 + 1$ также приводим над полем вычетов по модулю два \mathbb{Z}_2 :

$$x^2 + 1 = (x + 1)(x + 1).$$

Очевидно, многочлены вида $x - a$ неразложимы над любым полем. Это простейшие примитивные многочлены — линейные многочлены или двучлены.

1.4.6.1. Приводимость над полем рациональных чисел

Наибольший интерес в рамках курса дискретной математики представляют многочлены с целыми и рациональными коэффициентами. Разложимость многочлена с рациональными коэффициентами над полем рациональных чисел \mathbb{Q} легко сводится к разложимости многочленов с целыми коэффициентами над \mathbb{Q} вынесением за скобки наибольшего общего знаменателя всех дробных коэффициентов многочлена.

Рассмотрим сначала вопрос о возможности выделения у многочлена линейного множителя.

По теореме Безу (см. теорему 1.46) $P(x) = (x - a)P_1(x)$ тогда и только тогда, когда $P(a) = 0$.

Возьмем многочлен с целыми коэффициентами

$$P(x) = a_n x^n + \dots + a_1 x + a_0.$$

Найдем условия, при которых рациональное число p/q будет его корнем. Положим, что дробь p/q несократима и подставим ее в многочлен $P(x)$:

$$a_n \left(\frac{p}{q}\right)^n + \dots + a_1 \frac{p}{q} + a_0 = 0 \Leftrightarrow a_n p^n + \dots + a_0 q^n = 0.$$

Правая часть (равная нулю) очевидно делится на p . По условию теоремы все члены левой части, кроме последнего, делятся на p . Тогда и последнее слагаемое должно делиться на p , но так как q на p не делится, то свободный член a_0 делится на p . Аналогично доказываем, что старший член a_n многочлена P делится на q .

Таким образом доказана следующая теорема.

Теорема 1.49. Для существования линейного множителя $(x - p/q)$ у многочлена P с целыми коэффициентами необходимо, чтобы свободный член многочлена делился на p , а коэффициент старшего члена — на q .

Пример 1.59. Многочлен $P(x) = x^5 + 5x^3 + x^2 + 6x + 2$ нельзя разложить над полем \mathbb{Q} так, чтобы один из множителей был линейным.

Действительно, по теореме 1.49 в качестве возможных множителей достаточно проверить четыре: $x - 2$, $x + 2$, $x - 1$ и $x + 1$, но ни один из них не подходит. Заметим, что этот многочлен приводим над \mathbb{Q} :

$$P(x) = (x^3 + 3x + 1)(x^2 + 2).$$

Представляет интерес следующая теорема.

Теорема 1.50. Если многочлен с целыми коэффициентами раскладывается на множители над \mathbb{Q} , то он может быть разложен на множители с целыми коэффициентами.

Пример 1.60. Многочлен $6x^2 - x - 2$ раскладывается на множители

$$6x^2 - x - 2 = 6 \left(x + \frac{1}{2} \right) \left(x - \frac{2}{3} \right).$$

Это разложение можно переписать так, что коэффициенты многочленов-множителей будут целыми:

$$6x^2 - x - 2 = (2x + 1)(3x - 2).$$

Задача 1.1. Докажите теорему 1.50 самостоятельно.

Рассмотрим критерий неприводимости многочлена над \mathbb{Q} .

Теорема 1.51 (критерий Эйзенштейна). Если все коэффициенты многочлена с целыми коэффициентами кроме коэффициента старшего члена делятся на простое число p , а свободный член не делится на p^2 , то многочлен неприводим над \mathbb{Q} .

Доказательство. От противного. Пусть имеется требуемое разложение, в котором $k, m \geq 1$. По теореме 1.50 коэффициенты b_i и c_j можно считать целыми:

$$a_n x^n + \dots + a_1 x + a_0 = (b_k x^k + \dots + b_1 x + b_0)(c_m x^m + \dots + c_1 x + c_0),$$

тогда по формуле умножения многочленов (1.71) получаем

$$\begin{aligned} a_0 &= b_0 c_0, \\ a_1 &= b_1 c_0 + b_0 c_1, \\ a_2 &= b_2 c_0 + b_1 c_1 + b_0 c_2, \\ &\dots, \\ a_i &= b_i c_0 + \dots + b_0 c_i, \\ &\dots \end{aligned}$$

Из первого равенства следует, что ровно один из множителей b_0, c_0 делится на p (если бы делились оба, то их произведение делилось бы на p^2). Пусть для определенности это будет b_0 .

Тогда из второго равенства получим, что b_1 делится на p , и т. д. Дойдя до равенства с левой частью a_k (или a_m , в зависимости от того, что будет

больше, k или m , но в любом случае $k, m < n$), получим, что все коэффициенты b_i делятся на p , а значит, p можно вынести за скобку. Последнее означает, что все коэффициенты исходного многочлена делятся на p , а это противоречит условию. ■

Пример 1.61. Многочлен $x^5 + 2x^4 + 4x^3 + 6x + 2$ неразложим над \mathbb{Q} по критерию Эйзенштейна для $p = 2$.

1.4.6.2. Связь неприводимости многочленов с целыми коэффициентами над \mathbb{Z}_p и \mathbb{Q}

Теорема 1.52. Если многочлен с целыми коэффициентами и со старшим членом, не кратным простому числу p , приводим над полем \mathbb{Q} , то он приводим и над полем \mathbb{Z}_p .

Доказательство. Рассмотрим разложение многочлена над полем \mathbb{Q} . По теореме 1.50 его можно преобразовать в разложение с целыми коэффициентами.

Перейдем в полученном равенстве к модульной арифметике, заменив все коэффициенты остатками от деления на p . Справедливость равенства при этом не пострадает. Осталось объяснить, почему степени множителей сохраняются (иначе один из многочленов-множителей мог бы превратиться в константу, и разложения многочлена на нетривиальные множители не получилось бы).

Исчезновение старшего члена множителя означало бы, что его коэффициент делится нацело на p . Но тогда и старший коэффициент исходного многочлена делился бы на p , что противоречит условию. ■

Пример 1.62. Рассмотрим разложение из примера 1.59:

$$x^5 + 5x^3 + x^2 + 6x + 2 = (x^3 + 3x + 1)(x^2 + 2).$$

Рассмотрим это равенство последовательно как равенство в \mathbb{Z}_2 , \mathbb{Z}_3 , \mathbb{Z}_5 . Получим равенства, справедливость которых легко проверить умножением:

$$\text{в } \mathbb{Z}_2: x^5 + x^3 + x^2 = (x^3 + x + 1)x^2,$$

$$\text{в } \mathbb{Z}_3: x^5 + 2x^3 + x^2 + 2 = (x^3 + 1)(x^2 + 2),$$

$$\text{в } \mathbb{Z}_5: x^5 + x^2 + x + 2 = (x^3 + 3x + 1)(x^2 + 2).$$

Наибольший интерес для применения теоремы 1.52 имеет следующее ее следствие.

Следствие 1.21. Если старший член многочлена с целыми коэффициентами не делится на простое число p и если многочлен неприводим как многочлен над \mathbb{Z}_p , то он неприводим над \mathbb{Q} .

Доказательство. От противного. Если бы многочлен был приводим над \mathbb{Q} , то по теореме 1.52 он был бы приводим и над \mathbb{Z}_p , что не так. ■

Полученное следствие дает простой и достаточно мощный метод проверки многочлена на неприводимость над \mathbb{Q} .

Надо последовательно рассматривать его как многочлен над \mathbb{Z}_p для последовательно возрастающих простых чисел p . Неприводимые многочлены над полем вычетов, степень которых ограничена сверху, можно перечислить явно (перебором и проверкой неделимости на все многочлены меньшей степени). Если таким образом исходный многочлен будет сведен к неприводимому, то и неприводимость исходного доказана.

Теорема 1.52 дает только достаточные условия. Поэтому если исходный многочлен оказывается приводимым над \mathbb{Z}_p для достаточно большого числа простых чисел p , то это еще не означает, что он не может оказаться приводимым над \mathbb{Q} .

Пример 1.63. Рассмотрим все неприводимые над \mathbb{Z}_2 нетривиальные многочлены степени не выше двух:

$$P_1(x) = x,$$

$$P_2(x) = x + 1(x - 1, -x - 1, -x + 1 - \text{ это тот же самый двучлен}),$$

$$P_3(x) = x^2 + 1.$$

Пример 1.64. Является ли многочлен $x^3 + 7x^2 + 4x + 3$ неприводимым над \mathbb{Q} ?

Рассмотрим этот многочлен как многочлен над \mathbb{Z}_2 . Получим многочлен $x^3 + x^2 + 1$, который является неприводимым, так как при делении на x и $x + 1$ дает 1 (не делится на P_1 и P_2 из примера 1.63).

Проверка делимости на $P_3(x) = x^2 + 1$ не требуется, так как в случае делимости был бы получен другой делитель с меньшей степенью. Из решения следует, что и любой многочлен вида

$$(2n + 1)x^3 + (2m + 1)x^2 + 2kx + 2s + 1$$

будет неприводимым над \mathbb{Q} для любых целых неотрицательных n, m, k, s .

Задачи и вопросы

1. Пусть $P(x)$ и $Q(x)$ — многочлены с коэффициентами из \mathbb{Z}_6 степени больше либо равной 1. Верно ли, что:

а) степень произведения $P(x)Q(x)$ равна сумме степеней исходных многочленов;

б) степень остатка от деления $P(x)$ на $Q(x)$ меньше степени $P(x)$.

2. Найдите наибольший общий делитель многочленов и его линейное представление над полем вещественных чисел:

а) $x^4 + 2x^3 - x^2 - 4x - 2$ и $x^4 + x^3 - x^2 - 2x - 2$;

б) $x^5 + 3x^4 + x^3 + x^2 + 3x + 1$ и $x^4 + 2x^3 + x + 2$.

3. Найдите наибольший общий делитель многочленов и его линейное представление над полем $GF(2)$:

а) $x^5 + x + 1$ и $x^4 + x^3 + 1$;

б) $x^5 + x^3 + x$ и $x^4 + x + 1$.

4. Пусть $P(x)$ — интерполяционный многочлен Лагранжа, построенный по точкам $(x_1, y_1), \dots, (x_n, y_n)$. Всегда ли справедливо утверждение: $\deg P = n - 1$?

5. По формуле Лагранжа (1.77) найдите интерполяционный многочлен $Q(x)$, удовлетворяющий условиям:

а) $Q(-5) = -38, Q(2) = -38, Q(1) = -2, Q(-1) = 10, Q(-2) = 34$;

б) $Q(-3) = 9, Q(2) = -41, Q(-4) = -29, Q(-2) = 7, Q(-1) = 1$.

6. Многочлены $P(x)$ и $Q(x)$ с целыми коэффициентами не взаимно просты. Верно ли, что:

а) $P(x)$ неприводим на множестве многочленов с целыми коэффициентами;

б) $P(x) + Q(x)$ и $P(x) - Q(x)$ взаимно просты.

7. Дано кольцо многочленов $\mathbb{Z}_5[x]$. Разложите на неприводимые множители (и обоснуйте истинность разложения) многочлен $x^3 + 4x^2 + 2x + 3$.

8. Пусть $P(x)Q(x) \equiv 1 \pmod{M(x)}$, где $P(x), Q(x), M(x)$ — многочлены с рациональными коэффициентами степени больше 1. Докажите, что $P(x)$ взаимно прост с $M(x)$.

1.5. Базисы Гребнера

1.5.1. Задачи на доказательство и многочлены

Для многочленов одной переменной расширенный алгоритм Евклида позволяет решить задачу, можно ли выразить данный многочлен $P(x)$ как линейную комбинацию заданных многочленов $P_i(x)$ над тем же полем, коэффициентами которой являются многочлены над тем же полем:

$$P(x) = Q_1(x)P_1(x) + Q_2(x)P_2(x) + \dots + Q_n(x)P_n(x),$$

где $P(x), P_1(x), \dots, P_n(x), Q_1(x), \dots, Q_n(x) \in K[x]$.

Можно ли решить аналогичную задачу для многочленов от нескольких переменных?

Если бы такой алгоритм существовал, то некоторые классы задач на доказательство могли бы быть решены вычислением по алгоритму.

Например, в геометрических задачах на доказательство как условия, так и вывод можно описать равенствами нулю многочленов от нескольких переменных. Поэтому если бы удалось выразить подобным образом «многочлен вывода» через «многочлены условий», то равенство нулю последних влекло бы равенство нулю первого, т. е. из справедливости условий следовала бы справедливость утверждения теоремы.

Пример 1.65. Доказать, что если из заданной точки M провести прямую, пересекающую окружность, то произведение расстояний от этой точки до точек пересечения A и B прямой с окружностью будет постоянным: $AM \cdot BM = \text{const}$ (рисунок 1.4).

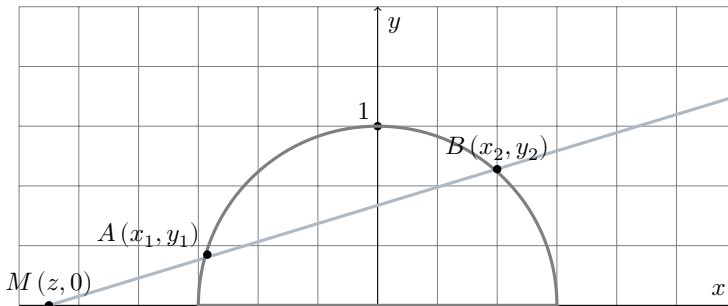


Рис. 1.4.

Введем координаты с началом в центре окружности (см. рисунок 1.4), тогда условия можно записать в виде равенства нулю трех многочленов.

Радиус окружности примем за 1, отметим точку M на оси абсцисс с координатой z . Обозначив координаты точки A как (x_1, y_1) , точки B —

(x_2, y_2) , получим условия в виде равенства нулю таких многочленов:

$$P_1(x_1, y_1) = 0 \Leftrightarrow x_1^2 + y_1^2 - 1 = 0 - \text{ точка } A \text{ лежит на окружности;}$$

$$P_2(x_2, y_2) = 0 \Leftrightarrow x_2^2 + y_2^2 - 1 = 0 - \text{ точка } B \text{ лежит на окружности;}$$

$$P_3(x_1, y_1, x_2, y_2, z) = 0 \Leftrightarrow y_1(x_2 - z) = y_2(x_1 - z) - \text{ точка } M \text{ лежит на прямой } AB.$$

Чтобы записать вывод, надо взять две другие точки A' и B' , записать для них те же условия, а затем приравнять указанные в утверждении произведения. Можно также взять точки A' и B' на диаметре, тогда расстояния будут $1 - r$ и $1 + r$, если точка M лежит внутри круга радиусом 1, и $r - 1$ и $r + 1$, если точка лежит вне этого круга. Здесь $r = |z|$. Тогда произведение расстояний будет $|1 - r^2|$, а вывод может быть записан так:

$$\sqrt{(z - x_1)^2 + y_1^2} \sqrt{(z - x_2)^2 + y_2^2} = |1 - z^2|$$

или после возведения в квадрат (что не нарушает эквивалентности, так как обе части неотрицательны):

$$((z - x_1)^2 + y_1^2) ((z - x_2)^2 + y_2^2) = (1 - z^2)^2.$$

Переносим все члены в одну часть, получим условие в виде равенства нулю многочлена от нескольких переменных:

$$P(x_1, y_1, x_2, y_2, z) = 0 \Leftrightarrow ((z - x_1)^2 + y_1^2) ((z - x_2)^2 + y_2^2) - (1 - z^2)^2 = 0.$$

Если удастся выразить P через P_1, P_2, P_3 , как указано ранее, то при любых значениях переменных выполнение условий обеспечивает выполнение утверждения. Тем самым, утверждение будет доказано.

1.5.2. Случай многочленов одной переменной

Проанализируем решение задачи о возможности выражения одного многочлена через другие для случая многочленов одной переменной.

В этом случае, как известно из расширенного алгоритма Евклида для многочленов, многочлен P можно выразить через многочлены P_1, \dots, P_n тогда и только тогда, когда P делится на наибольший общий делитель многочленов P_1, \dots, P_n .

Пример 1.66. Многочлен $P = x^3 + x - 2$ выражается через $P_1 = x^2 - 1$ и $P_2 = (x - 1)^2$, так как делится на многочлен $x - 1$, который является НОД

многочленов P_1 и P_2 :

$$x^3 + x - 2 = A(x)(x^2 - 1) + B(x)(x - 1)^2, \text{ где } A(x) = x + 1, B(x) = -1.$$

В случае если необходимо выразить многочлен P через единственный многочлен P_1 , то задача упрощается — надо поделить P на P_1 «уголком» (см. алгоритм 1.20): если получится остаток ноль, то многочлен P выражается через многочлен P_1 (будем говорить, редуцируется к нулю через P_1).

Алгоритм Евклида позволяет также решить задачу с несколькими многочленами:

1) сначала находим $D(P_1, \dots, P_n)$, используя алгоритм деления многочленов с остатком;

2) затем делим многочлен P на $D(P_1, \dots, P_n)$: если остаток равен нулю, то многочлен P выражается через многочлены P_1, \dots, P_n , иначе — нет.

Пример 1.67. В примере 1.66 деление P_1 на P_2 дает в остатке многочлен $P_3 = 2x - 2$, на который делится многочлен P . Поэтому если к исходным многочленам P_1 и P_2 добавить P_3 , то многочлен P , как будем говорить в дальнейшем, может быть «редуцирован к нулю по системе многочленов» P_1, P_2, P_3 . При этом для редукции будет использован только многочлен P_3 , отсутствующий в исходной системе многочленов.

1.5.3. Понятие идеала

Аналогично определению 1.30 дадим формальное определение для многочлена нескольких переменных.

Определение 1.34. Конечную сумму вида

$$P(x_1, x_2, \dots, x_n) = \sum_{J=\{j_1, j_2, \dots, j_n\}} c_J x_1^{j_1} x_2^{j_2} \dots x_n^{j_n}, \quad c_J \in K,$$

называют *многочленом от n переменных над полем K* , а набор из целых неотрицательных чисел $J = \{j_1, j_2, \dots, j_n\}$ — *мультииндексом*.

Многочлен вида $c x_1^{j_1} x_2^{j_2} \dots x_n^{j_n}$ называют *одночленом*. Степенью (ненулевого) одночлена называют целое число $|J| = j_1 + j_2 + \dots + j_n$. Степенью *многочлена* называют максимальную из степеней его одночленов, тождественный ноль не имеет степени.

Множество многочленов от n переменных над полем K обозначают $K[x_1, x_2, \dots, x_n]$.

Замечание 1.41

Если в определении 1.34 допустить отрицательные степени, то полученное выражение называют многочленом Лорана.

Для многочленов от n переменных легко доказать теорему, аналогичную теореме 1.44.

Теорема 1.53. Множество многочленов $K[x_1, x_2, \dots, x_n]$ есть ассоциативное коммутативное кольцо с единицей.

Исходная задача может быть интерпретирована по-другому.

Для многочленов $P_1, \dots, P_n \in K[x_1, x_2, \dots, x_n]$ строим множество многочленов вида:

$$I = \{Q_1P_1 + Q_2P_2 + \dots + Q_nP_n \mid Q_1, \dots, Q_n \in K[x_1, x_2, \dots, x_n]\}. \quad (1.79)$$

Требуется определить принадлежность многочлена P этому множеству.

Многочлены P_1, \dots, P_n называют *базисом* I (несмотря на то, что они не обязательно образуют минимальное множество).

Пример 1.68. В примере 1.67 базис образуют как многочлены P_1, P_2 , так и многочлены P_1, P_2, P_3 . Кроме того, базис образует и единственный многочлен P_3 .

Так построенное множество многочленов образует новую математическую структуру — идеал во множестве всех многочленов $K[x_1, x_2, \dots, x_n]$.

Определение 1.35. Непустое подмножество I кольца M называют *идеалом* этого множества, если разность двух элементов этого подмножества и умножение его элемента на любой элемент множества M не выводят за пределы I :

- 1) $a \in I, b \in I \Rightarrow a - b \in I$,
- 2) $a \in I, m \in M \Rightarrow am \in I$.

Упражнение 1.9. Докажите, что для любых P_1, \dots, P_n принадлежащих $K[x_1, \dots, x_n]$, множество I , определенное в (1.79), есть идеал в кольце многочленов $K[x_1, \dots, x_n]$.

Историческая справка

Понятие идеала (или в первоначальной терминологии идеального числа) было введено в 1847 г. немецким математиком Эрнстом Куммером для одного частного случая числовых полей в попытке доказать великую теорему Ферма.

1.5.3.1. Идеалы в кольце многочленов одной переменной

В случае многочленов одной переменной множество I , определенное в (1.79), совпадает с множеством, которое образуют многочлены, кратные $D(P_1, \dots, P_n)$.

Процедура определения принадлежности многочлена одной переменной P идеалу, определяемому многочленами P_1, \dots, P_n , крайне проста, если среди набора есть НОД этих многочленов R . Редукция осуществляется вычитанием из P многочлена R , умноженного на многочлен Q — частное многочленов P и R (предварительно можно сделать редукцию исходного многочлена и по другим многочленам P_1, \dots, P_n). Если же среди многочленов не содержится их НОД, то подобная редукция не даст результата.

Поэтому для применения того же приема базис идеала I придется расширить, добавив НОД многочленов (или, что то же самое, остатки от деления многочленов P_i друг на друга — рано или поздно НОД среди них появится).

Теорема 1.54. Добавление к базису идеала остатков от деления многочленов идеала друг на друга не меняет идеала.

Доказательство. Заметим, что если $R(x) \equiv P_j(x) \pmod{P_i(x)}$, т. е. $R(x)$ есть остаток от деления $P_j(x)$ на $P_i(x)$, то существует многочлен $Q(x)$ — частное, такой что $R(x) = P_j(x) - Q(x)P_i(x)$.

По свойству идеала умножение многочлена, принадлежащего идеалу, на любой многочлен дает многочлен идеала, а вычитание многочленов идеала не выводит за его пределы. Так что $P_i(x) \in I, P_j(x) \in I \Rightarrow R(x) \in I$. ■

1.5.4. Линейные многочлены многих переменных.

Понятие базиса Гребнера

Рассмотрим решение задачи о разложении многочлена для линейных многочленов многих переменных. В этом случае для построения базиса исходной системы и осуществления редукции по нему можно применить метод Гаусса.

Пример 1.69. Можно ли выразить многочлен

$$P(x, y, z, u) = 3y - 2u + 5z - 4 + 9x$$

через многочлены:

$$P_1(x, y, z, u) = 2 - u + 2z - y + 5x, \quad P_2(x, y, z, u) = -z - u + 2y + 3x - 5,$$

$$P_3(x, y, z, u) = 4x - 3y - 2z + 3 - u, \quad P_4(x, y, z, u) = z - u + 6x - 6y + 10?$$

Расположим одночлены каждого многочлена в лексикографическом порядке (далее в учебнике будет дано формальное определение лексикографического порядка):

$$P(x, y, z, u) = 9x + 3y + 5z - 2u - 4,$$

$$P_1(x, y, z, u) = 5x - y + 2z - u + 2, \quad P_2(x, y, z, u) = 3x + 2y - z - u - 5,$$

$$P_3(x, y, z, u) = 4x - 3y - 2z - u + 3, \quad P_4(x, y, z, u) = 6x - 6y + z - u + 10.$$

Если не внести в набор P_1, P_2, P_3, P_4 изменения, то редуцировать по нему многочлен P не удастся, так как после первой редукции, например через многочлен P_1 , коэффициент при x станет равным нулю, в то время как среди оставшихся будут ненулевые, и дальнейшая редукция будет невозможна.

Сделаем типичное для метода Гаусса элементарное преобразование (в дальнейшем такое преобразование будем называть построением S -многочлена): построим многочлен

$$S(P_1, P_2) = 3P_1 - 5P_2 = -13y + 11z + 2u + 31,$$

исключив старший член. Добавим полученный S -многочлен к системе как P_5 (предварительно попробовав редуцировать через остальные, но здесь редукции не будет, так как все остальные многочлены содержат член со старшей переменной x).

Далее построим $S(P_1, P_3) = 4P_1 - 5P_3 = 11y + 18z + u - 7$. Редуцируя его через многочлен P_5 (через другие это невозможно), получим многочлен $71z + 7u + 50$ (после домножения на подходящую константу, чтобы иметь дело с целыми коэффициентами). Поскольку более он не редуцируется, обозначим его P_6 и добавим к системе.

Затем построим $S(P_1, P_4) = 6P_1 - 5P_4 = 24y + 7z - u - 38$. Редуцируем его через P_5 и P_6 . После первого редуцирования (и домножения на удобный коэффициент) получим $71z + 7u + 50$, который равен P_6 и поэтому редуцируется через него к нулю.

Теперь построим S -многочлены для P_2 и многочленов, больших его по номеру (все S -многочлены для P_1 уже построены).

Получим $S(P_2, P_3) = 4P_2 - 3P_3 = 17y + 2z - u - 29$. Редуцируем его через P_5 и получим (после домножения на соответствующий коэффициент) $71z + 7u + 50$, который редуцируется к нулю через P_6 .

Далее, $S(P_2, P_4) = 2P_2 - P_4 = 10y - 3z - u - 20$. Редуцируем его через P_5 и получим опять $71z + 7u + 50$, который редуцируется к нулю через P_6 .

Наконец, $S(P_3, P_4) = 3P_3 - 2P_2 = 3y - 8z - u - 11$. Редуцируем его через P_5 и P_6 и получим ноль:

$$3y - 8z - u - 11 - \left(-\frac{3}{13}\right)(-13y + 11z + 2u + 31). \quad (1.80)$$

Домножив (1.80) на числовой коэффициент 13, позволяющий сделать коэффициенты результата целыми, получим равенство

$$-68z - 7u - 50 - 71z - 7u - 50 - (-1)(71z + 7u + 50) = 0.$$

Других S -многочленов построить нельзя. Построенную таким образом систему $\{P_1, P_2, P_3, P_4, P_5, P_6\}$ называют базисом Гребнера для идеала, определяемого исходной системой $\{P_1, P_2, P_3, P_4\}$.

Базис получил специальное название (базис Гребнера) потому, что через него любой многочлен идеала редуцируется к нулю. Этот базис не обязан содержать минимальное число элементов, но при возможности количество элементов следует делать меньше. В данном случае два из трех многочленов P_1, P_2, P_3 можно удалить, так как они выражаются через оставшиеся три многочлена системы (что ясно из их построения).

Оставим базис $\{P_1, P_5, P_6\}$.

Теперь для решения вопроса, можно ли выразить P через исходные многочлены, достаточно проверить редуцируемость P к нулю по базису $\{P_1, P_5, P_6\}$.

После редуцирования P через P_1 получаем $Q_1 = 24y + 7z - u - 38$. После редуцирования Q_1 через P_2 (и домножения на удобный коэффициент) имеем $71z + 7u + 50$, редуцируемый к нулю через P_6 .

Таким образом, исходный вопрос решен положительно: P выражается через P_1, P_2, P_3, P_4 .

Упражнение 1.10. Найдите многочлены Q_1, Q_2, Q_3, Q_4 — коэффициенты разложения P по P_1, P_2, P_3, P_4 .

Замечание 1.42

В полученном алгоритме нетрудно видеть алгоритм Гаусса для определения ранга расширенной системы (если добавление уравнения $P = 0$ не меняет ранга системы, значит, $P = 0$ не влияет на множество решений системы и выражается как линейная комбинация других уравнений).

Полученные приемы обобщает на случай системы многочленов многих переменных алгоритм Бухбергера. Алгоритм Бухбергера позволяет получить базис Гребнера идеала, заданного произвольным конечным набором многочленов.

Если базис Гребнера построен, то для решения вопроса о принадлежности многочлена идеалу его достаточно редуцировать по многочленам этой системы.

1.5.5. Алгоритм Бухбергера

Для формулировки алгоритма Бухбергера нам понадобится аккуратно определить понятия:

- 1) главного члена многочлена с несколькими переменными;
- 2) редукции многочлена;
- 3) S -многочлена.

1.5.5.1. Главный член многочлена с несколькими переменными

Одночлены многочлена с несколькими переменными можно упорядочить по-разному. Рассмотрим один из способов.

Приведем многочлен к нормальному виду, т. е. раскроем скобки и приведем подобные члены. Теперь расположим все множители одночлена в лексикографическом порядке, считая, например, переменную x_1 главной, x_2 следующей по старшинству и т. д.

Наконец, упорядочим все одночлены в лексикографическом порядке. Так, если два одночлена включают переменную x_1 в разных степенях, то старшим будем считать одночлен с большей степенью переменной. Если степени главной переменной одинаковы, то сравниваем следующие по старшинству переменные и т. д.

Определение 1.36. Одночлен $x_1^{k_1} x_2^{k_2} \dots x_n^{k_n}$ считается старше одночлена $x_1^{m_1} x_2^{m_2} \dots x_n^{m_n}$, если существует i , при котором $k_i > m_i$, а все предыдущие степени одинаковы: $k_j = m_j$ при $j < i$.

По аналогии с многочленами одной переменной можно обозначить

$$\deg(x_1^{k_1} x_2^{k_2} \dots x_n^{k_n}) = (k_1, k_2, \dots, k_n).$$

Тогда порядок на множестве одночленов определяется лексикографическим порядком на наборах, задающих степени многочленов.

Иногда одночлен $x_1^{k_1} x_2^{k_2} \dots x_n^{k_n}$ записывают в виде x^k , имея в виду:

$$x = (x_1, x_2, \dots, x_n), \quad k = (k_1, k_2, \dots, k_n).$$

Пример 1.70. Упорядочим одночлены по убыванию их старшинства в многочлене $P(x, y, z) = y^2z + z^4 + xy^3$ (здесь x соответствует старшей переменной x_1 , y — переменной x_2 , z — переменной x_3).

Первые два члена не включают x , т. е. соответствующий показатель степени при x в этих одночленах равен нулю, а у последнего одночлена — единице, значит, он и есть главный. Аналогично член y^2z старше z^4 . Получаем

$$P(x, y, z) = xy^3 + y^2z + z^4.$$

Если обозначить старший член многочлена P (как и для многочленов одной переменной) $l(P)$, то в примере 1.70 $l(P) = xy^3$, а результат упорядочения может быть записан так:

$$\deg(xy^3) > \deg(y^2z) > \deg(z^4).$$

Упражнение 1.11. Упорядочьте по старшинству члены следующего многочлена:

$$P(x_1, x_2, x_3, x_4) = x_1x_3^4x_4^2 + x_3^4x_4^2 + x_1^2x_2^2x_4^2 + x_1^2x_2^2x_3^4.$$

1.5.5.2. Редукция многочлена с несколькими переменными

Определение 1.37. Если старший член многочлена P делится на старший член другого многочлена Q и дает в результате одночлен t , то операцию $P - tQ$ называют *редукцией* многочлена P с помощью Q .

Теорема 1.55. Замена редуцируемого базисного многочлена P многочленом P^* , полученным в результате редукции по другим многочленам базиса, не меняет идеала.

Идея доказательства.

При однократной редукции P по многочлену Q получаем $P^* = P - tQ$, где t — некоторый одночлен. Отсюда $P = P^* + tQ$, т. е. исходный многочлен может быть выражен через многочлены нового базиса. Аналогичное утверждение справедливо и при дальнейших редукциях.

Замечание 1.43

Поскольку на делимость многочленов умножение на ненулевую константу не влияет, редукцией можно называть и операцию вида $\alpha(P - tQ)$ с любой константой $\alpha \neq 0$.

Пример 1.71. Многочлен $P(x, y, z) = xy^3 + y^2z + z^4$ из примера 1.70 редуцируется с помощью многочлена $P_1(x, y, z) = 3y^2 + y + z^5$ к многочлену

$$y^2z + z^4 - 1/3xy^2 - 1/3xyz^5,$$

или (после умножения на 3) к многочлену

$$P_2(x, y, z) = -xy^2 - xyz^5 + 3y^2z + 3z^4,$$

старший член полученного многочлена меньше, чем был раньше (хотя членов стало больше).

Полученный многочлен P_2 можно еще раз редуцировать с помощью P_1 :

$$P_3(x, y, z) = -3xyz^5 + xy + xz^5 + 9y^2z + 9z^4.$$

Теперь его старший член меньше старшего члена многочлена P_1 и дальнейшая редукция невозможна.

Это решение можно записать в виде деления «уголком». Используя обобщение алгоритма 1.20 (см. пример 1.48), получаем:

$$\begin{array}{r|l} xy^3 + y^2z + z^4 & 3y^2 + y + z^5 \\ \hline xy^3 + \frac{1}{3}xy^2 + \frac{1}{3}xyz^5 & \frac{1}{3}xy - \frac{1}{9}x \\ \hline -\frac{1}{3}xy^2 - \frac{1}{3}xyz^5 + y^2z + z^4 & \\ \hline -\frac{1}{3}xy^2 - \frac{1}{9}xy - \frac{1}{9}xz^5 & \\ \hline -\frac{1}{3}xyz^5 + \frac{1}{9}xy + \frac{1}{9}xz^5 + y^2z + z^4 & \end{array}$$

Таким образом видно, что редукция одного многочлена по другому является обобщением алгоритма деления многочленов на многочлены нескольких переменных.

Запишем алгоритм редукции многочлена многих переменных по базису Гребнера.

Нам потребуется вспомогательный алгоритм редукции многочлена P по Q . Результат работы алгоритма 1.23 — редуцированный многочлен сохраняется в P . Напомним, что $l(P)$ и $l(Q)$ — обозначение старших коэффициентов P и Q соответственно.

Алгоритм 1.23. Редукция многочлена P по Q

while $l(P)$ делится на $l(Q)$ **do**

$$P \leftarrow P - \frac{l(P)}{l(Q)} \cdot Q$$

end while

Редукция многочлена по базису Гребнера сводится к редукции многочлена по каждому из многочленов базиса. Упорядочим многочлены базиса

по убыванию их степени. Тогда сначала многочлен редуцируется по первому многочлену базиса, затем — по второму и т. д.

Результат работы алгоритма 1.24 — редуцированный многочлен сохраняется в многочлене P .

Алгоритм 1.24. Редукция многочлена P по базису Гребнера P_1, \dots, P_n

```

while есть многочлен  $P_i$ , по которому редуцируется  $P$  do
   $Q \leftarrow$  первый из многочленов  $P_i$ , у которого  $l(P)$  делится на  $l(P_i)$ 
   $P \leftarrow$  результат редукции  $P$  по  $Q$  (алгоритм 1.23)
end while

```

Из примера 1.71 видно, что в процессе редукции степени переменных младших членов быстро растут. Может ли быть так, что процесс редукции не остановится? Нет.

Докажем предварительно вспомогательное утверждение.

Лемма 1.10. При редукции степень одночлена уменьшается.

Доказательство. После домножения делителя на частное от деления их старших коэффициентов степени делимого и делителя сравниваются, но другие одночлены делителя, как и делимого, будут иметь меньшую степень. После вычитания старшие члены уничтожатся, старшим членом станет один из одночленов меньшей степени, и степень многочлена уменьшится. ■

Для дальнейших рассуждений полезно использовать геометрическую интерпретацию степеней одночленов (мономов).

Сопоставим одночлену от n переменных $x_1^{k_1} x_2^{k_2} \dots x_n^{k_n}$ точку с координатами (k_1, k_2, \dots, k_n) (если переменная в одночлене отсутствует, соответствующий показатель степени берется нулевым). Заметим, что координаты являются неотрицательными целыми числами.

Пример 1.72. На рисунке 1.5а показаны графические изображения степеней всех одночленов многочлена $x^4 y + 2x^3 y^2 - x^3 + 5x^2 y^3 + y^2$.

Теорема 1.56. Процесс редукции заканчивается за конечное число шагов.

Доказательство. Докажем теорему для многочленов двух переменных. Для многочленов большего числа переменных доказательство аналогичное. Заметим, что чем меньше степень одночлена, тем левее лежит точка, изображающая его степень. Если же точки лежат на одной вертикали, то меньшие степени изображаются точками, лежащими ниже.

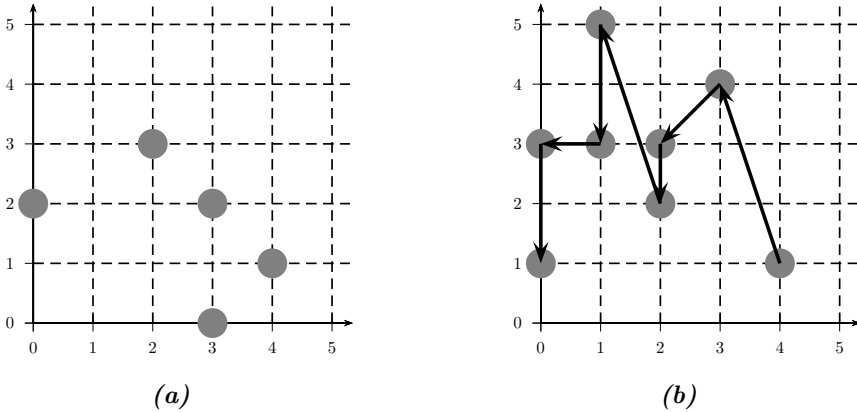


Рис. 1.5.

Таким образом, если проследить за движением точки, изображающей степень старшего члена многочлена при его редукции, то получится траектория, состоящая из перемещения точки влево (и, возможно, одновременно вверх или вниз) либо вниз по вертикали (рисунок 1.5b).

Поскольку координаты точки целые и неотрицательные, траектория может состоять только из конечного числа звеньев, что и означает окончание процесса редукции за конечное число шагов. ■

1.5.5.3. Построение S -многочленов

Определение 1.38. Для двух многочленов P и Q определим S -многочлен следующим образом:

$S(P, Q) = Pq - Qp$, где $l(P) = pd, l(Q) = qd, d = D(l(P), l(Q))$ — нетривиальный НОД старших членов многочленов P и Q .

Если $D(l(P), l(Q)) = \text{const}$, то говорят, что многочлены не имеют *зацепления*.

Теорема 1.57. Добавление к базису идеала S -многочленов, построенных по базисным многочленам, не меняет идеала.

Доказательство. Если многочлен $R = S(P, Q)$, то $R = Pq - Qp$, где p и q — некоторые одночлены. Значит, R принадлежит идеалу, которому принадлежат P и Q . ■

Пример 1.73. Многочлен $Q = x^3y + x^3$ не редуцируется с помощью многочлена $P = x^2y^4 + x^2y^2$, но имеет с ним зацепление. Тогда S -многочлен для них будет $S(Q, P) = y^3Q - xP = x^3y^3 - x^3y^2$.

Заметим, что степени обоих одночленов результата превышают степени исходных многочленов. Таким образом, при использовании S -многочленов

для построения базисов Гребнера нужно обосновать конечность процесса добавления этих многочленов к исходному набору.

1.5.5.4. Построение базиса Гребнера

Пусть дана система m многочленов от n переменных P_1, P_2, \dots, P_m . Она определяет идеал

$$I = \{Q_1P_1 + Q_2P_2 + \dots + Q_nP_m \mid Q_1, \dots, Q_m \in K[x_1, x_2, \dots, x_m]\}.$$

Требуется построить базис Гребнера этого идеала.

Определение 1.39. *Базисом Гребнера* называют такой базис идеала, для которого принадлежность многочлена идеалу определяется редукцией по многочленам базиса.

Алгоритм 1.25. Алгоритм Бухбергера

```

while существуют многочлены, имеющие зацепление do
    // проводим поиск только среди тех пар, которые еще не рассматривались
    Для найденного зацепления строим  $S$ -многочлен
    Редуцируем его по всем многочленам системы
    // подойдет любая последовательность редукций
    if многочлен редуцировался к ненулевому многочлену then
        добавляем его к системе
    end if
end while

```

Для дальнейших рассуждений потребуется вспомогательная лемма.

Лемма 1.11. Любой мономиальный идеал имеет конечный базис.

Доказательство. Проиллюстрируем доказательство на многочленах двух переменных. Для многочленов многих переменных доказательство проводится аналогично.

Сначала заметим, что каждый одночлен (моном) порождает все одночлены, лежащие правее (умножение на первую переменную) и/или выше (умножение на вторую переменную) (рисунок 1.6a). Все эти одночлены делятся на данный моном.

Геометрически задача означает, что имеется некоторая область, лежащая в первом квадранте и являющаяся объединением «углов» (рисунок 1.6b). Требуется найти конечное покрытие этой области такими «углами».

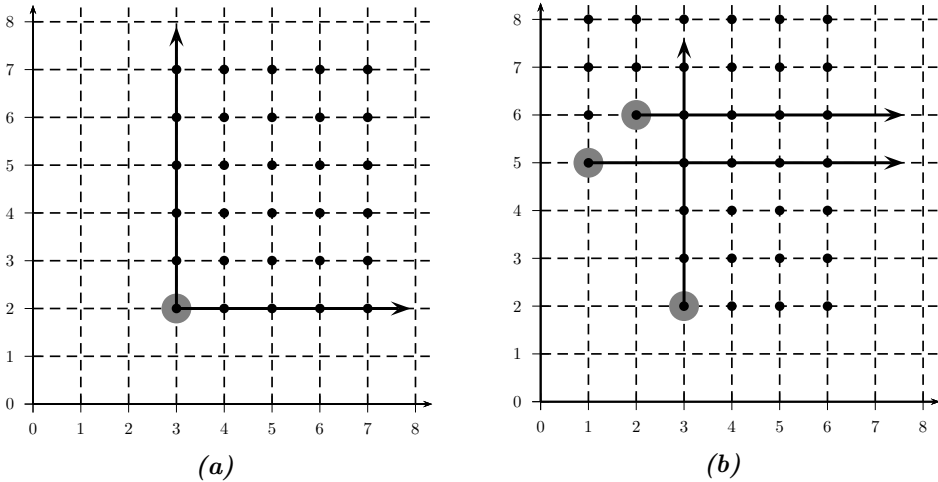


Рис. 1.6.

Попробуем, наоборот, построить область из бесконечного числа «углов». После того как поставлен первый «угол», следующий нужно ставить или левее, или ниже (иначе область не изменится) (рисунок 1.6b).

Но, как и в случае описанной выше траектории «левее или ниже», число шагов, меняющих область, конечно, поэтому любая область подобного вида имеет конечное покрытие «углами». ■

Докажем теперь основное утверждение.

Теорема 1.58. Предложенный алгоритм 1.25 строит базис Гребнера для любого конечного набора многочленов от нескольких переменных.

Доказательство. Требуется доказать, что:

- 1) алгоритм заканчивает свою работу за конечное число шагов (как видно из примера 1.73 степень S -многочлена может превышать степени исходных многочленов, поэтому это утверждение не очевидно);
- 2) полученный базис является базисом Гребнера.

Доказательство первого пункта будем вести от противного, т. е. положим, что применение операций построения S -многочленов и редукции привело к бесконечному множеству многочленов.

Проанализируем множество их старших членов (мономов) и докажем, что если рассмотреть идеал, построенный на них (мономиальный идеал), то в нем можно будет построить конечный базис.

Установим, что процесс построения S -многочленов с редукцией по уже имеющимся многочленам не может продолжаться бесконечно. От противного.

Пусть описанные процедуры порождают бесконечное множество нередуцируемых друг через друга многочленов. Рассмотрим конечный базис множества их старших членов — мономов. Возьмем многочлены, старший член которых не вошел в этот базис. По лемме их старшие коэффициенты делятся на старшие члены некоторых многочленов, вошедших в базис, а значит, соответствующие им многочлены должны редуцироваться по выделенным таким образом многочленам, что противоречит нередуцируемости построенных многочленов.

Докажем редуцируемость любого многочлена идеала по построенному базису. От противного.

Пусть имеется многочлен $P = \sum H_i P_i$, который нельзя редуцировать по построенному базису. Старший член такого многочлена не должен делиться ни на один из старших членов, построенных по алгоритму Бухбергера многочленов P_i . Это означает, что в представленной сумме старшие члены уничтожились и получившийся после приведения одночленов старший член не делится ни на одного из старших членов P_i . Будем считать, что из всех представлений выбрано то, для которого степень самого старшего члена правой части наименьшая из возможных:

$$P = \sum H_i P_i. \quad (1.81)$$

Покажем, что эту сумму можно преобразовать так, что степень старшего члена уменьшится, тем самым придя к противоречию, доказывающему утверждение теоремы.

Разобьем сумму на две: первая будет содержать только члены старшей степени, вторая — члены меньшей степени (поэтому в дальнейшем ее можно не рассматривать).

Первую сумму можно разбить еще на две суммы, выделив у многочленов H_i старший член. Тогда первая из этих сумм

$$P = \sum l(H_i) P_i, \quad (1.82)$$

а вторая будет содержать члены меньшей степени, и поэтому ее можно, как и в предыдущем случае, не рассматривать.

Оказывается, вследствие сокращения старших членов сумму (1.82) можно переписать в виде

$$\sum l_{ij} S(l(H_i) P_i, l(H_j) P_j), \quad (1.83)$$

где $l(H_i)$ — старший член многочлена H_i (моном); l_{ij} — числовые коэффициенты (см. задачу 1.2).

Заметим, что при построении S -многочлена для многочленов P и Q одинаковой степени в выражении $S(P, Q) = Pq - Qp$ p и q — числа, поэтому степень S -многочлена будет меньше степеней исходных многочленов. Значит, у полученной суммы (1.83) степень старшего члена меньше, чем у суммы (1.82).

Выражение $S(l(H_i)P_i, l(H_j)P_j)$ можно записать так: $x^m S(P_i, P_j)$, где

$$x = (x_1, x_2, \dots, x_n), \quad k = (k_1, k_2, \dots, k_n)$$

(см. задачу 1.3).

Поскольку все нередуцируемые S -многочлены уже включены в базис, многочлены $S(P_i, P_j)$ редуцируются к нулю, а значит, их можно представить в виде суммы $\sum G_s P_s$, степень старшего члена которой совпадает со степенью $S(P_i, P_j)$ (см. задачу 1.4).

Подставив полученные выражения в сумму (1.83), получим разложение, аналогичное разложению (1.81), но со старшим членом меньшей степени. Добавив две другие суммы, снова получим сумму того же вида (1.81) и снова со старшим членом меньшей степени, что противоречит предположению о выборе разложения. ■

Задача 1.2. Обозначим через x^k одночлен вида

$$x^k = x_1^{k_1} x_2^{k_2} \dots x_n^{k_n},$$

пусть $f_i = a_i x^k + \dots$

Выразите коэффициенты l_{ij} через b_i так, чтобы выполнялось равенство

$$\sum b_i f_i = \sum l_{ij} S(f_i, f_j),$$

если известно, что старшие коэффициенты в сумме уничтожаются, а b_i и l_{ij} — числа.

Задача 1.3. Докажите, что $S(M_i P_i, M_j P_j)$ делится на $S(P_i, P_j)$, где M_i — мономы.

Задача 1.4. Докажите, что если многочлен P редуцируется к нулю по многочленам P_i , его можно представить в виде суммы $\sum G_i P_i$, степень которой совпадает со степенью P .

Замечание 1.44

† Задачи 1.2 и 1.3 — задачи повышенной сложности.

В процессе построения базисов Гребнера, как отмечалось ранее, могут появиться многочлены больших степеней, что затрудняет решение задач вручную. Применение пакетов символьных вычислений позволяет выполнять символьные вычисления с многочленами больших степеней.

Вот как выглядит решение задачи из начала раздела (см. пример 1.65) с помощью пакета Maple (в текст внесены комментарии к решению).

Пример 1.74 (построение базиса Гребнера).

```
>P:=((z-x1)^2+y1^2)*((z-x2)^2+y2^2)-(1-z^2)^2;
```

$$P := ((z - x_1)^2 + y_1^2)((z - x_2)^2 + y_2^2) - (1 - z^2)^2,$$

так записывается многочлен, принадлежность которого к идеалу нужно установить.

```
>with(Groebner):G:=[x1^2+y1^2-1,x2^2+y2^2-1,y1*(x2-z)-
y2*(x1-z),(y1-y2)*w-1];
```

так описывается команда подключения операций, используемых при построении базисов Гребнера, и указывается набор многочленов, определяющих идеал.

Обратите внимание, что кроме трех многочленов, описывающих принадлежность точек окружностям и одной прямой, появилось четвертое условие $(y_1 - y_2)w - 1$, которое указывает на то, что ординаты точек y_1 и y_2 не могут быть одинаковыми, иначе прямая, проведенная через них, не пересечет ось x (для задания этого условия в виде равенства нулю некоторого многочлена пришлось ввести новую переменную w).

Подобные условия в геометрических задачах мы часто не замечаем, принимая их как очевидные. Однако без этого условия решения мы не получим, что легко проверить.

```
>GB:=gbasis(G,plex(x1,x2,y1,y2,z,w));
```

$$\begin{aligned} GB := & [8z^2y_2^3w + 4z^2y_2^2 + 4wy_2 + 4wy_2z^2 + 4z^2y_2^4w^2 - 4w^2z^2y_2^2 + z^4 - 2z^3 + \\ & + 1, wy_1 - wy_2 - 1, 3y_2 - 2y_1z^2 + y_1 + 4y_1z^2y_2^2 + y_1z^4 - y_2z^2 + \\ & + 4y_2^2w - 2y_2z^2 + 4z^2y_2^4w + 4y_2^3z^2 - 4wz^2y_2^2, -zx_2 + 2wy_2 - z^2 + \\ & + 1 + x_2z^3 + 2z^2y_2^3w + 2z^2y_2^2 - 2wy_2z^2, 2x_2wy_2 + z^2x_2 + x_2 + \\ & + 2y_2^3zw + 2y_2^2z - 2zwy_2 - 2z, 4y_1zy_2^2 - 3y_1z + 2x_2y_2 - 2x_2y_2z^2 + \\ & + 2y_1x_2 + y_2z + y_1z^3 - y_2z^3, x_2^2 + y_2^2 - 1, x_1z^2 - x_1 + z^2x_2 - x_2 - \\ & - 2zy_1y_2, 2y_2x_1 - 2x_2y_2z^2 + 2x_2y_2 + 4y_1zy_2^2 + y_1z^3 - y_1z - y_2z^3 - \\ & - y_2z, x_1x_2 - x_1z - zx_2 + y_1y_2 + 1, x_1^2 + y_1^2 - 1]. \end{aligned}$$

Эта процедура вычисляет базис Гребнера, учитывая лексикографический порядок, явно определяемый перечислением переменных. В базисе десять многочленов

```
>reduce(P,GB,plex(x1,x2,y1,y2,z,w));
```

0

Последняя процедура выполняет редуцирование исходного многочлена по построенному базису. В силу того, что он редуцировался к нулю, можно заключить, что утверждение теоремы истинно и доказано.

Замечание 1.45

При практических вычислениях порядок, в котором идут переменные, может существенно повлиять на трудоемкость вычислений.

Например, если в примере 1.74 переменную z сделать старшей, то базис Гребнера будет существенно проще.

```
>GB:=gbasis(G,plex(z,x1,x2,y1,y2,w));
```

```
GB := [wy1 - wy2 - 1, x2^2 + y2^2 - 1, x1^2 + y1^2 - 1, z + x1wy2 - x2wy2 - x2]
```

```
>reduce(P,GB,plex(z,x1,x2,y1,y2,w));
```

0

Базис Гребнера можно уменьшить и другими способами. Рассмотрим два приема, которые позволяют упростить базис (самостоятельно докажете корректность таких упрощений).

Прием 1. Если старший член одного из многочленов базиса делится на старший член другого, то первый из многочленов можно удалить из базиса.

Прием 2. Если нестарший член одного многочлена базиса делится на старший член другого, то первый многочлен можно упростить, редуцировав его по второму многочлену (относительно выделенного нестаршего члена), т. е. домножить второй многочлен на некоторый одночлен и результат вычесть из исходного многочлена так, чтобы выделенный одночлен уничтожился.

Пример 1.75. В примерах 1.73 и 1.74 не все исходные многочлены присутствуют в построенных базисах Гребнера.

В обоих примерах многочлен, описывающий принадлежность точек прямой, был изменен.

Задачи и вопросы

1. Сформулируйте следующую геометрическую задачу в терминах многочленов. В треугольнике ABC точка D лежит на прямой AC , а точка E — на прямой BC , так что $AD = BE$. F — точка пересечения DE и AB . Докажите, что $FD \cdot AC = EF \cdot BC$.

2. Определите, можно ли выразить многочлен $q = -3x^3 - 5x^2 - x^4 + 5$ через многочлены $p_1 = x^3 - 2x - 3$ и $p_2 = x^4 - 1$. Если можно, то найдите выражение.

3. Изобразите степени мономов многочлена

$$x^3 + 3xy^2 - y^2 + 2x - xy^3 - x^2 - xy + xy^4 - 1$$

точками плоскости.

4. Упорядочьте одночлены многочлена

$$x^3 + 3xy^2 - y^2 + 2x - xy^3 - x^2 - xy + xy^4 - 1$$

по степеням.

5. Найдите конечный базис мономиального идеала, порожденного бесконечным множеством одночленов $\{xy^n; x^2; y^n \mid n - \text{натуральное}\}$. Отметьте в координатной плоскости точки, соответствующие базисным мономам.

6. Редуцируйте многочлен

$$P(x, y) = x^3 - x^2 + xy^4 - xy^3 + 3xy^2 - xy + 2x - y^2 - 1$$

по базису Гребнера из задачи 9. Редуцируется ли многочлен к нулю?

7. Постройте S -многочлен для многочленов

$$P(x, y) = 2x^2y^2 + x^2y \text{ и } Q(x, y) = 3xy^4 - xy^3.$$

8. Постройте базис Гребнера для идеала, построенного по следующим многочленам $\{x^2 - 1, (x - 1)y, (x + 1)z\}$.

9. Упростите следующий базис Гребнера:

$$\{x^2y^2 - y^2 - 1, x^2 + xy^3 + xy + y^2 + 1, xy^2 + x, y^4 + 2y^2 + 1\}.$$

10. Докажите, что добавление к базису многочленов, полученных редукцией, или S -многочленов не меняет идеала, определяемого этим базисом.

1.6. Теория множеств и комбинаторика

Комбинаторика изучает вопросы о том, сколько различных комбинаций, подчиненных тем или иным условиям, можно составить из заданных объектов и как перечислить эти объекты.

Прежде чем перейти к систематическому изложению, рассмотрим несколько классических задач, отражающих ее базовые идеи.

1.6.1. Вводные задачи

1.6.1.1. Принцип умножения

Принцип умножения особенно ярко проявился в открытии Д. И. Менделеевым Периодической таблицы химических элементов. Пытаясь найти скрытую закономерность в свойствах элементов, ученый выделил базовые (возможно, скрытые от непосредственного восприятия) признаки, комбинации которых характеризуют свойства элементов.

Более формально принцип умножения связан с перечислением (в виде таблицы) всех комбинаций двух признаков различной природы. Эта идея привела к понятию декартова произведения множеств (признаков).

$$A \times B = \{(a, b) | a \in A, b \in B\}.$$

Задача 1.5. Сколько существует автомобильных номеров, составленных из трех произвольных букв русского алфавита и четырех десятичных цифр.

Решение. Автомобильный номер можно интерпретировать как один из элементов декартова произведения трехбуквенных «слов» $\{AAA; AAB; \dots\}$ и четырехзначных наборов цифр $\{0000; 0001; \dots\}$.

С другой стороны, множество трехбуквенных слов может быть представлено как комбинация буквы и двухбуквенного слова и т. д. В результате можно рассмотреть множество всех потенциально допустимых номеров автомобилей как декартово произведение семи множеств:

$$A \times A \times A \times D \times D \times D \times D = A^3 D^4,$$

где A — алфавит (множество букв русского языка); D — множество десятичных цифр.

Понятно, что написанная формула может рассматриваться как формула для подсчета числа элементов декартова произведения множеств:

$$N = |A|^3 \times |D|^4 = 33^3 \times 10^4 = 359\,370\,000$$

— число всех автомобильных номеров.

1.6.1.2. Принцип сложения

Принцип сложения тоже можно иллюстрировать замечательным открытием — изобретением телескопа Г. Галилеем.

Г. Галилей, используя комбинаторные рассуждения, «переоткрыл» устройство телескопа, которое являлось в то время коммерческим секретом.

Удивительно, но сохранилась запись рассуждений Галилея об этом открытии:

Мои рассуждения были такими.

Устройство должно иметь либо одно стекло, либо больше. Оно не может состоять из единственного стекла, потому что стекло может быть выпуклым, вогнутым или иметь параллельные грани. Но последнее из упомянутых стекол не меняет изображения, вогнутое же уменьшает предметы, а выпуклое увеличивает, но делает их в то же время неотчетливыми и искаженными.

Перейдя к устройству с двумя стеклами и помня, что стекло с параллельными гранями ничего не меняет, я сделал вывод, что нужный эффект не может быть достигнут комбинированием этого стекла с другими.

Таким образом, мне осталось исследовать взаимоположение выпуклого и вогнутого стекол.

В этих рассуждениях Галилей сначала разбивает все возможные комбинации стекол (варианты устройства телескопа) на группы по числу стекол в устройстве. Затем в каждой из групп перебирает комбинации стекол различного вида.

Идея разбить все варианты на группы, каждая из которых состоит из «одинаково устроенных» комбинаций, в теории прикладных алгоритмов получила название «разделяй и властвуй» (divide and conquer).

Принцип сложения лежит в основе построения рекуррентных формул.

Задача 1.6. Вывести рекуррентную формулу для подсчета C_n^k — числа k -элементных подмножеств n -элементного множества (читается как «число сочетаний из n по k »).

Решение. Выделим из n элементов один и разобьем все k -элементные подмножества на два класса:

- содержащие выделенный элемент;

- не содержащие его.

Первых будет C_{n-1}^{k-1} , так как один элемент уже выбран, и из оставшихся $n - 1$ элементов надо выбрать еще $k - 1$ элемент.

Вторых будет C_{n-1}^k , так как один элемент запрещается выбирать, и надо выбрать все k элементов из оставшихся $n - 1$. Так как любое подмножество (сочетание) принадлежит либо одному, либо другому классу и классы не пересекаются, получаем рекуррентную формулу:

$$C_n^k = C_{n-1}^{k-1} + C_{n-1}^k.$$

Процесс вычислений по этому рекуррентному соотношению можно представить в виде пирамиды, которую называют *треугольником Паскаля* (рисунок 1.7).

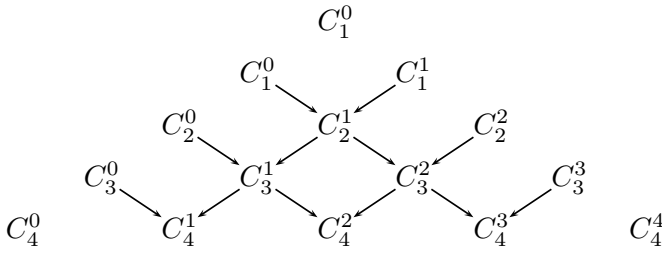


Рис. 1.7.

В этом треугольнике каждое число по выведенной рекуррентной формуле есть сумма двух, стоящих над ним, а по границам треугольника стоят единицы.

Задача 1.7 (о размене монет). Сколькими способами можно разменять один рубль монетами достоинством 50, 10 и 5 копеек.

Решение. Обозначим количество способов размена монеты достоинством k на монеты меньшего номинала как D_k .

Любой вариант размена можно записать, указав количество монет данного номинала. Например, $(1; 3; 4)$ соответствует размену

$$50 + 10 + 10 + 10 + 5 + 5 + 5 + 5.$$

При этом первым можно указать количество монет номинала 50, затем 10 и, наконец, 5 (будем считать порядок, в котором «выдаются» монеты после размена, несущественным, и разные комбинации в нашей постановке задачи отличаются только составом монет).

Тогда любой размен начинается либо с 50, либо с 10, либо с 5.

Соответственно, количество разменов оставшейся суммы для первого вида размена (в общем случае) будет D_{k-50} , для второго — D'_{k-10} , для третьего — D''_{k-5} , где D' обозначает размен без использования монет достоинством 50 копеек, а D'' — без использования монет достоинством 50 и 10 копеек. Тем самым D'' всегда равно 1.

Получаем формулы:

$$D_k = D_{k-50} + D'_{k-10} + 1, \quad D'_k = D'_{k-10} + 1,$$

по которым рекурсией (сверху вниз) можно получить искомое число разменов. Как легко заметить, $D_0 = 1$ — это соответствует размену $10 = 50 + 50$. Далее имеем:

$$\begin{aligned} D_{100} &= D_{50} + D'_{90} + 1 = D_0 + D'_{40} + 1 + D'_{80} + 1 + 1 = D'_{40} + D'_{80} + 4 = \\ &= D'_{30} + 1 + D'_{70} + 1 + 4 = D'_{30} + D'_{70} + 6 = D'_0 + D'_{40} + 12 = \\ &= D'_{40} + 13 = D'_0 + 17 = 18. \end{aligned}$$

1.6.1.3. Использование взаимно однозначного соответствия множеств

Рассмотрим примеры, когда вместо прямого подсчета количества элементов данного множества A можно установить взаимно однозначное соответствие между A и другим множеством B , количество элементов которого уже подсчитано или считается легче.

Задача 1.8. Какое из чисел больше: C_n^k или C_n^{n-k} ?

Решение. По определению C_n^k есть количество k -элементных подмножеств n -элементного множества. Например, C_5^2 показывает, сколькими способами можно выбрать два предмета из пяти различных.

Если обозначить последнее множество как $A = \{a, b, c, d, e\}$, то все варианты выбора легко перечислить явно: $ab, ac, ad, ae, bc, bd, be, cd, ce, de$. Всего их 10.

Теперь, положим, нас интересует вопрос о том, сколькими способами можно выбрать три предмета из пяти различных. Если обратить внимание на то, что, выбирая два элемента из пяти, мы оставляем три, то между подмножествами выбранных и остающихся элементов возникает естественное взаимно однозначное соответствие:

$$\begin{aligned} \{a; b\} - \{c; d; e\}, \{a; c\} - \{b; d; e\}, \{a; d\} - \{b; c; e\}, \{b; c\} - \{a; d; e\}, \\ \{b; d\} - \{a; c; e\} \text{ и т. д.} \end{aligned}$$

В итоге получаем равенство

$$C_n^k = C_n^{n-k}.$$

Задача 1.9 (о числе перестановок). Сколькими способами можно переставить между собой цифры числа 12345?

Решение. Рассуждения для решения этой задачи таковы: на первом месте может стоять любая из пяти цифр $\{1, 2, 3, 4, 5\}$, с каждой из этих цифр может комбинироваться одна из четырех, кроме той, которая уже поставлена, что дает 5×4 комбинаций. Далее с любой из этих комбинаций комбинируется любая цифра из трех оставшихся (кроме двух уже использованных) и т. д. По принципу умножения получаем $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5! = 120$.

Однако заметим, что принцип умножения в сформулированном выше виде неприменим, так как у нас нет декартова произведения множеств. Когда, например, на первом месте стоит 1, на втором месте могут стоять любые перестановки цифр $\{2, 3, 4, 5\}$, а когда на первом месте стоит 2, то на втором месте стоят уже элементы другого множества — множества перестановок цифр $\{1, 3, 4, 5\}$.

Но хотя эти множества и разные, легко понять, что число элементов в них одинаково. Для доказательства достаточно в каждой перестановке цифр первого множества поменять 2 на 1, и получится перестановка цифр второго множества. Аналогично делается и обратный переход.

Таким образом, все перестановки четырех элементов имеют одинаковое число элементов, и можно сопоставить их одному фиксированному множеству (из 24 элементов). После этого применение принципа умножения оправдано, и можно записать

$$P_n = nP_{n-1},$$

где P_n — число перестановок n элементов. По этой рекуррентной формуле легко находим, что

$$P_n = n!.$$

Задача 1.10. Доказать, что количество шестизначных номеров, у которых сумма первых трех цифр равна сумме оставшихся трех («счастливые» номера билетов), равно количеству шестизначных номеров с суммой цифр 27.

Решение. В этой задаче прямой подсчет достаточно труден для нахождения обоих количеств, однако доказательство равенства количества элементов обоих множеств элементарно.

Следующее сопоставление решает задачу:

$$(a; b; c; d; e; f) - (a; b; c; 9 - d; 9 - e; 9 - f).$$

Действительно, то, что сопоставление взаимно однозначно, очевидно. С другой стороны, сумма цифр второго билета в показанном сопоставлении равна

$$a + b + c + (9 - d) + (9 - e) + (9 - f) = 27 + [(a + b + c) - (d + e + f)].$$

Поэтому, если первый билет «счастливым», то

$$a + b + c = d + e + f$$

и второй билет имеет сумму цифр 27. Легко показать и обратное.

1.6.1.4. Различные способы рассуждений

Идею взаимно однозначного соответствия можно видоизменить так, чтобы она не только приводила к факту равномогности некоторых множеств, но и давала алгоритм для вычисления количества элементов этих множеств. Для этого можно применить несколько разных способов рассуждений к вычислению количества элементов одного и того же множества, а затем сравнить результаты.

Задача 1.11 (о футбольной команде). Сколькими способами из n претендентов можно выбрать футбольную команду из k человек, а в ней капитана?

Решение. Первый способ рассуждений. Выберем из n человек команду из k человек — это можно сделать C_n^k способами. Далее выберем из этих k человек капитана, что можно сделать k способами. По принципу умножения будем иметь kC_n^k команд, различающихся либо составом, либо капитанами.

Второй способ рассуждений. Выберем из n человек капитана, что можно сделать n способами. Далее из оставшихся человек выберем остальную часть команды, что можно сделать C_{n-1}^{k-1} способами. По принципу умножения получаем другую запись искомого количества nC_{n-1}^{k-1} . Заключаем, что

$$kC_n^k = nC_{n-1}^{k-1},$$

откуда получаем рекуррентную формулу для вычисления C_n^k :

$$C_n^k = \frac{n}{k} C_{n-1}^{k-1}.$$

Именно благодаря этому число сочетаний из n по k часто обозначают как $\binom{n}{k}$. Например, если $n = 5, k = 3$, получаем

$$C_5^3 = \frac{5}{3} \cdot \frac{4}{2} C_3^1.$$

Но $C_3^1 = 3$, что очевидно из определения C_n^k . Таким образом, $C_5^3 = 10$. В общем случае

$$C_n^k = \frac{n}{k} C_{n-1}^{k-1} = \frac{n}{k} \frac{n-1}{k-1} C_{n-2}^{k-2} = \dots = \frac{n}{k} \frac{n-1}{k-1} \dots \frac{n-k+1}{1}.$$

1.6.1.5. Принцип использования разных языков описания

Идея взаимно однозначного соответствия часто проявляется в форме переформулировки исходной задачи, «переводе» с одного языка описания на другой. Например, ранее мы уже столкнулись с языком теории множеств (подсчет элементов декартова произведения) и подсчета слов в некотором алфавите. Это два самых важных, но не единственных языка описания комбинаторных задач.

Часто переход от одного описания задачи к другому позволяет применить приемы, известные в терминах другой интерпретации. Таким образом, одним из важных приемов решения задач является установление взаимно однозначного соответствия между множествами комбинаций в разных по форме задачах.

Задача 1.12. Найти число двоичных последовательностей с k единицами и $n - k$ нулями.

Решение. Интерпретируем k -элементные подмножества n -элементного множества как двоичные последовательности. Для этого упорядочим элементы множества, и каждое подмножество закодируем набором из единиц и нулей, в зависимости от того, входит ли очередной элемент в подмножество или не входит. Например, для множества $\{a; b; c; d; e\}$ подмножество $\{b; c; e\}$ кодируется двоичным набором 01101.

Таким образом, число двоичных последовательностей с k единицами и $n - k$ нулями равно C_n^k . Заметим, что отсюда следует, что число всех подмножеств множества из n элементов равно 2^n , так как каждая цепочка есть элемент n -кратного декартова произведения B^n бинарных множеств $B = \{0; 1\}$.

Задача 1.13. Сколько существует способов представить число 10 в виде суммы пяти неотрицательных целых слагаемых (с учетом порядка сла-

гаемых, т. е. разложения $4 + 1 + 3 + 0 + 2$ и $2 + 0 + 3 + 4 + 1$ считаются разными).

Решение. Каждое из представлений можно закодировать набором нулей и единиц, где суммарное число единиц равно 10, а нули заменяют плюсы: 11110101110011 кодирует первое из приведенных выше разложений, а 11001110111101 — второе. В задаче 1.12 показано, что число таких двоичных наборов равно

$$\frac{14}{4} \cdot \frac{13}{3} \cdot \frac{12}{2} \cdot \frac{11}{1} = 1001.$$

Эту же задачу можно сформулировать по-другому. Например, так. В кондитерской продают пирожные пяти сортов. Надо купить 10 пирожных. Сколькими способами это можно сделать? Каждая покупка кодируется упорядоченным разложением 10 в сумму пяти неотрицательных слагаемых. Так, разложение $4 + 1 + 3 + 0 + 2$ кодирует покупку: 4 пирожных первого вида, 1 пирожное — второго и т. д. Таким образом, ответ в задаче такой же — 1001.

В общем случае эту задачу называют нахождением числа сочетаний с повторениями: имеется k видов элементов, каждого вида элементов бесконечное множество. Сколькими способами можно выбрать n элементов?

Ответ: C_{n+k-1}^{k-1} .

Часто процесс комбинирования можно свести к формальному оперированию алгебраическими объектами.

1.6.1.6. Принцип формального оперирования алгебраическими объектами вместо комбинаторных рассуждений

Задача 1.14. Рассмотрим следующее выражение, содержащее n множителей:

$$(x + y)(x + y)(x + y) \cdots (x + y).$$

Требуется раскрыть скобки и упростить выражение.

Решение. Посчитаем количество одночленов, содержащих k букв x (остальные $n - k$ букв будут y).

Каждый такой одночлен может быть закодирован двоичным набором из n цифр: 1 — если мы выбираем x из очередной скобки, 0 — если выбираем y . Таким образом, каждый одночлен, содержащий k букв x и $(n - k)$ букв y , т. е. одночлен $x^k y^{n-k}$, будет встречаться C_n^k раз. Итак, получаем формулу

$$(x + y)^n = x^n + C_n^{n-1} x^{n-1} y + \cdots + C_n^k x^k y^{n-k} + \cdots + y^n.$$

Эту формулу называют *биномом Ньютона*. С точки зрения комбинаторики важность этой формулы в том, что в ней комбинаторные по сути

объекты — числа сочетаний — становятся коэффициентами многочлена, с которым можно выполнять алгебраические операции, не задумываясь о комбинаторном смысле действий. Например, подставив вместо x и y единицы, получим тождество

$$(1 + 1)^n = 1^n + C_n^{n-1}1^{n-1}1 + \dots + C_n^k 1^k 1^{n-k} + \dots + 1^n = \sum_{k=0}^n C_n^k,$$

комбинаторный смысл которого в том, что все подмножества (которых 2^n), можно разбить на классы, в каждом из которых содержатся подмножества с одинаковым числом элементов (число подмножеств с количеством элементов k равно C_n^k). При подстановке $x = 1, y = -1$ получим равенство

$$\sum_k C_n^{2k-1} = \sum_k C_n^{2k},$$

для которого провести комбинаторное рассуждение уже не так просто.

Еще менее очевидный результат получается при подстановке 1 вместо x и y в формулу, полученную дифференцированием бинома Ньютона по x :

$$\begin{aligned} n(x + y)^{n-1} &= nx^{n-1} + (n-1)C_n^{n-1}x^{n-2}y + \dots + kC_n^k x^{k-1}y^{n-k} + \dots + 0, \\ n2^{n-1} &= n1^{n-1} + (n-1)C_n^{n-1}1^{n-2}1 + \dots + kC_n^k 1^{k-1}1^{n-k} + \dots + 0 = \\ &= \sum_{k=0}^n kC_n^k = \sum_{k=1}^n kC_n^k. \end{aligned}$$

Такой переход от оперирования комбинаторными объектами к действиям с функциями называют методом *производящих функций*. Подробно производящие функции рассмотрим в следующем подразделе.

1.6.1.7. Принцип Дирихле

Самая распространенная формулировка принципа Дирихле звучит следующим образом:

если в n клетках сидит больше n зайцев, то найдется клетка, в которой сидят по крайней мере два зайца.

В роли «зайцев» могут выступать различные предметы и математические объекты. Принцип Дирихле можно сформулировать и на языке множеств и отображений:

при любом отображении множества A , состоящего из $n+1$ элемента, в множество B из n элементов, найдутся два элемента множества A , имеющие один и тот же образ.

Несмотря на совершенную очевидность этого принципа, его применение является весьма эффективным методом решения задач, дающим во многих случаях наиболее простое и изящное решение.

Историческая справка

Дирихле Петер Густав Лежен (1805–1859) — выдающийся немецкий математик. Ученик Фурье. После смерти Гаусса возглавил его кафедру в Геттингенском университете. Сделал ряд крупных открытий в теории чисел и математическом анализе.

Задача 1.15. Имеются n целых чисел a_1, \dots, a_n . Доказать, что среди них найдутся одно или несколько чисел, сумма которых делится на n .

Рассмотрим n сумм:

$$\sum_{i=1}^k a_i, \quad 1 \leq k \leq n.$$

Если одна из этих сумм делится на n , то задача решена. Если же ни одна из этих сумм не делится на n , тогда по принципу Дирихле существуют суммы, принадлежащие одному классу вычетов по модулю n :

$$\sum_{i=1}^s a_i \equiv \sum_{j=1}^k a_j \pmod{n}.$$

Утверждение задачи следует из определения сравнения по модулю.

Задача 1.16. На клетчатой бумаге отметили пять точек, расположенных в узлах клеток. Доказать, что хотя бы один из отрезков, соединяющих эти точки, проходит через узел клетки.

Введем на клетчатой бумаге систему координат с началом координат в одном из узлов, осями, направленными вдоль линий сетки, и единичным отрезком, равным стороне клетки. Тогда все отмеченные точки будут иметь целочисленные координаты. Покажем, что найдутся две точки из пяти, у которых одна и та же четность координат (x, y) .

Возможно четыре варианта «четности» координат: (четная, четная), (четная, нечетная), (нечетная, четная), (нечетная, нечетная). Следовательно из пяти точек две будут иметь один вариант «четности». Обозначим их (x_1, y_1) и (x_2, y_2) .

Тогда середина отрезка, соединяющего эти две точки, имеет координаты

$$\left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right),$$

которые являются целыми числами в силу одинаковой четности x_1, x_2 и y_1, y_2 . Таким образом, середина этого отрезка лежит в узле сетки, т. е. данный отрезок является искомым.

Перейдем к более формальному описанию базовых понятий комбинаторики.

1.6.2. Размещения и сочетания

Определение 1.40. Способ расположения в *определенном порядке* некоторого числа элементов из заданного конечного множества S называют *размещением*. Иначе говоря, способ выбора этих элементов из S , когда *существенна* последовательность выбора. Если последовательность выбора *несущественна*, то такой способ называют *сочетанием*. При этом может допускаться или нет повторение элементов.

Пример 1.76. Пусть $S = \{a, b, c\}$. Будем выбирать из множества S по два элемента. Тогда:

- 1) размещения без повторений — $\{ab, ac, ba, bc, ca, cb\}$;
- 2) сочетания без повторений — $\{ab, ac, bc\}$;
- 3) размещения с повторениями — $\{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$;
- 4) сочетания с повторениями — $\{aa, ab, ac, bb, bc, cc\}$.

Введем следующие обозначения:

- число размещений без повторений из n элементов по k обозначим A_n^k ;
- число сочетаний без повторений обозначим C_n^k или $\binom{n}{k}$.

Определение 1.41. Размещение без повторений из n элементов по n называют *перестановкой* n элементов и обозначают P_n .

По определению положим $C_n^0 = 1$, $A_n^0 = 1$. Справедлива следующая теорема.

Теорема 1.59.

$$A_n^r = n(n-1)\cdots(n-r+1) = \frac{n!}{(n-r)!}, \quad C_n^r = \frac{A_n^r}{r!} = \frac{n!}{(n-r)!r!}.$$

Доказательство. Необходимо заполнить r позиций, выбирая из n -элементного множества без повторений. Первого кандидата можно выбрать n способами. Для каждого из этих способов второго кандидата можно выбрать $n-1$ способом. Для каждого из $n(n-1)$ способов третьего кандидата можно выбрать $n-2$ способами и т. д. Следовательно всего существует $n(n-1)(n-2)\cdots(n-r+1)$ способ выбора.

$$n! \simeq n^n \sqrt{2\pi n} \exp\left(-n + \frac{1}{12n} - \frac{1}{360n^2} + \dots\right) \text{ при } n \rightarrow \infty.$$

1.6.3. Биномиальные коэффициенты

Оказывается, что числа сочетаний C_n^k , полученные в предыдущем разделе, обладают целым рядом свойств, которые оказываются очень полезными при решении ряда комбинаторных задач. Числа сочетаний C_n^k называют также *биномиальными коэффициентами*. Смысл этого названия устанавливается следующей теоремой, известной также как формула *бинома Ньютона*.

Теорема 1.62.

$$(x + y)^n = \sum_{k=0}^n C_n^k x^k y^{n-k}. \quad (1.85)$$

Доказательство. Коэффициент при $x^k y^{n-k}$ равен числу способов, которыми при раскрытии скобок в $(x + y)^n$ из n скобок можно выбрать k скобок с x , а в остальных $n - k$ взять y . ■

Следствие 1.22. Справедливы следующие свойства C_n^k :

1) $\sum_{k=0}^n C_n^k = 2^n$;

2) $\sum_{k=0}^n (-1)^k C_n^k = 0$;

3) каждое n -элементное множество имеет ровно 2^n подмножеств (включая пустое подмножество и его самого).

Доказательство. Для доказательства п. 1 и 2 в (1.85) нужно положить соответственно $x = y = 1$ и $x = -1, y = 1$. Пункт 3 следует непосредственно из определения сочетания без повторов. ■

Замечание 1.48

Формула бинома Ньютона и результаты следствия 1.22 к теореме 1.62 обсуждались при изучении алгебраических объектов комбинаторными способами (см. задачу 1.14).

Теорема 1.63. Справедливы следующие равенства:

1) $C_n^k = C_n^{n-k}$,

2) $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$.

Доказательство. Каждой выборке k элементов из n однозначно соответствует выборка оставшихся $n - k$ элементов из n . Это доказывает п. 1. Для доказательства п. 2 рассмотрим n -элементное множество $S = \{s_1, s_2, \dots, s_n\}$ и его произвольный элемент s_i . Все выборки k элементов из n распадаются на две непересекающиеся группы.

1. Выборки, содержащие элемент s_i . Таких выборов всего C_{n-1}^{k-1} , так как выбираем из $n - 1$ кандидата (элемент s_i уже выбран) на оставшееся $k - 1$ место (одно место уже занято элементом s_i).

2. Выборки, не содержащие элемент s_i . Таких выборов всего C_{n-1}^k , так как выбираем из $n - 1$ кандидата (элемент s_i не может принадлежать выборке) на k мест.

Сложив оба числа, получаем требуемое. ■

Следствие 1.23 (треугольник Паскаля). Треугольник Паскаля построен по следующему правилу: его строку с номером i образуют коэффициенты разложения бинома $(x + y)^i$, т. е. числа $C_i^0, C_i^1, \dots, C_i^i$. Согласно теореме 1.63 треугольник имеет вид, представленный на рисунке 1.8.

			1							
			1	1						
			1	2	1					
			1	3	3	1				
			1	4	6	4	6			
			1	5	10	10	5	1		
			1	6	15	20	15	6	1	
			1	7	21	35	35	21	7	1
.....										

Рис. 1.8.

Замечание 1.49

| Результаты теоремы 1.63 обсуждаются в задачах 1.6 и 1.8.

Теорема 1.64 (тождество Коши).

$$C_{n+m}^k = \sum_{s=0}^k C_m^s C_n^{k-s}.$$

Доказательство. Предположим, что имеется m белых и n красных шаров. Из них нужно выбрать k любых шаров. Это можно сделать C_{n+m}^k способами.

Рассмотрим произвольную выборку, в которой s белых шаров ($s \geq 0$). Способов получить такую выборку C_m^s , и для каждого способа оставшиеся $k - s$ красных шаров можно выбрать C_n^{k-s} способами.

Поэтому всего способов выбрать k шаров из $m + n$ при фиксированном числе белых шаров s равно $C_m^s C_n^{k-s}$. Остается просуммировать по всем возможным значениям $s = 0, 1, \dots, k$. ■

Следствие 1.24.

$$C_{2n}^m = \sum_{s=0}^n (C_n^s)^2.$$

Доказательство. Достаточно применить теорему 1.64 при $m = k$ и $n = k$. ■

Приведем еще несколько свойств биномиальных коэффициентов для самостоятельного доказательства.

Теорема 1.65. Справедливы соотношения:

- 1) $C_n^k C_k^m = C_n^m C_{n-m}^{k-m}$,
- 2) $\sum_{n=0}^m n C_m^n = m 2^{m-1}$.

1.6.4. Кодирование с исправлением ошибок. Граница Хемминга

Определение 1.42. Неотрицательную вещественную функцию

$$d(a, b) : M \times M \rightarrow \mathbb{R}_+$$

называют *функцией расстояния* (или *метрикой*) на множестве M , если выполнены следующие условия (аксиомы метрики):

- 1) $d(a, b) \geq 0$, $d(a, b) = 0$ тогда и только тогда, когда $a = b$;
- 2) $d(a, b) = d(b, a)$;
- 3) $d(a, b) + d(b, c) \geq d(a, c)$.

Определение 1.43. Пусть на каждый кодовый символ a выделено m двоичных разрядов:

$$a = (a_1, a_2, \dots, a_m), \quad a_i = 0, 1, \quad i \in 1 : m.$$

Определим бинарную операцию на множестве кодовых символов:

$$a \oplus b = c = (c_1, c_2, \dots, c_m), \quad \text{где } c_i = a_i + b_i \pmod{2}.$$

Введем понятие расстояния между кодовыми символами: $d(a, b)$ — число позиций $i : a_i \neq b_i$. Функцию d называют *кодovým расстоянием Хемминга*.

Пример 1.78. $a = (0, 1, 1, 1), b = (1, 0, 1, 0)$, тогда $a \oplus b = (1, 1, 0, 1)$ и $d(a, b) = 3$.

Очевидно следующее утверждение.

Лемма 1.12. Введенная функция $d(a, b)$ является функцией расстояния.

Доказательство. Условия 1, 2 очевидны. Установим условие 3. Если $a_i = c_i$, то $d(a, c)$ не меняется, а выражение в левой части неравенства может возрасти. Если же $a_i \neq c_i$, то либо $a_i \neq b_i$, либо $b_i \neq c_i$. ■

Предположим, что при приеме кодируемых символов возможно появление ошибок не более чем в $r \leq m$ разрядах. Поставим задачу исправления ошибок в разрядах за счет уменьшения числа кодовых символов.

Пусть $A = \{a^{(1)}, a^{(2)}, \dots, a^{(p)}\}$ — множество выбранных кодовых символов, $p \leq 2^m$. Справедлива следующая теорема.

Теорема 1.66. Если $d(a^{(i)}, a^{(j)}) \geq 2r + 1$, то можно исправлять ошибки, возникающие не более чем в r разрядах.

Доказательство. Пусть при приеме символа $a^{(i)}$ не более чем в r разрядах возникли ошибки. Тогда принятый символ b попадает в окрестность символа $a^{(i)}$ радиусом r , т. е. справедливо $d(a^{(i)}, b) \leq r$. Обозначим соответствующую окрестность $A_i = \{a | d(a^{(i)}, a) \leq r\}$. Тогда $A_i \cap A_j = \emptyset$. Действительно, пусть $w \in A_i \cap A_j$, тогда

$$d(a^{(i)}, w) \leq r, \quad d(a^{(j)}, w) \leq r, \quad d(a^{(i)}, a^{(j)}) \leq d(a^{(i)}, w) + d(a^{(j)}, w) \leq 2r.$$

Полученное противоречие и доказывает утверждение. ■

Определение 1.44. Множество A_i назовем *областью декодирования* символа $a^{(i)}$. Любой принятый символ b , попадающий в A_i , отождествляем с $a^{(i)}$.

Сколько элементов в множестве A_i ? Это число способов выбора некоторых k ($0 \leq k \leq r$) компонент кодового символа $a^{(i)}$, т. е. число символов, отличающихся от символа $a^{(i)}$ в $0, 1, 2, \dots, r$ разрядах. Тогда

$$|A_i| = C_m^0 + C_m^1 + \dots + C_m^r = \sum_{k=0}^r C_m^k.$$

Теперь легко оценить число элементов в множестве выбранных символов:

$$A = \{a^{(1)}, a^{(2)}, \dots, a^{(p)}\}.$$

Так как всего 2^m кодируемых символов, то для p получаем оценку (*граница Хемминга*):

$$p \sum_{k=0}^r C_m^k \leq 2^m, \quad p \leq \frac{2^m}{\sum_{k=0}^r C_m^k}. \quad (1.86)$$

Пример 1.79. Пусть $m = 3$, $r = 1$. Тогда согласно (1.86):

$$p \leq \frac{2^3}{(C_3^0 + C_3^1)} = \frac{8}{1 + 3} = 2.$$

В качестве двух кодовых символов, удовлетворяющих условию теоремы 1.66, можно взять следующие:

$$a^{(0)} = (0, 0, 0), \quad a^{(1)} = (1, 1, 1),$$

при этом

$$A_0 = \{(0, 0, 1), (0, 1, 0), (1, 0, 0), (0, 0, 0)\},$$

$$A_1 = \{(1, 1, 0), (1, 0, 1), (0, 1, 1), (1, 1, 1)\}.$$

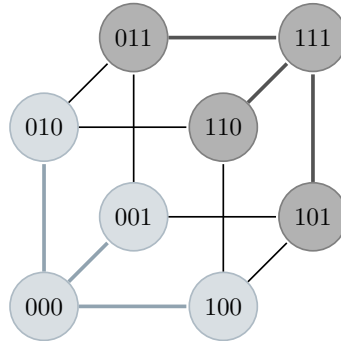


Рис. 1.9.

На рисунке 1.9 элементы области декодирования A_0 выделены светло серым цветом, а элементы области декодирования A_1 — темно серым.

Замечание 1.50

В общем случае вопрос о способе выбора элементов $a^{(i)}$, удовлетворяющих условию теоремы 1.66, является достаточно сложным и остается вне рамок данной книги. Подробно этот вопрос изучен в [4].

1.6.5. Полиномиальное кодирование

Будем решать задачу предыдущего подраздела (см. пример 1.79), не уменьшая количество кодовых слов согласно границе Хемминга, а увеличивая разрядность кода.

Определение 1.45. Код называют *линейным*, если для любых двух кодовых слов a и b их сумма $a \oplus b$, введенная в определении 1.43, также является кодовым словом и кодовым словом является λa — произведение кодового слова a на элемент поля $\lambda \in \{0, 1\}$.

Определение 1.46. *Вес Хемминга* $w(a)$ кодового слова a есть число его ненулевых компонент. *Минимальным кодовым расстоянием* кода называют величину d^* :

$$d^* = \min_{a \neq b} d(a, b),$$

где d — расстояние Хемминга (см. определение 1.43), минимум берется по всем кодовым словам.

Из определений 1.45 и 1.46 очевидно вытекает следующее утверждение.

Теорема 1.67. Для линейного кода минимальное кодовое расстояние d^* определяется из равенства

$$d^* = \min_{c \neq 0} w(c),$$

где минимум берется по всем кодовым словам, кроме нулевого.

Пример 1.80. Код из примера 1.79 является линейным и $d^* = 3$.

Определение 1.47. *Совершенный*, или *плотно упакованный*, код — код, для которого сферы (окрестности) одинакового радиуса вокруг кодовых слов, не пересекаясь, покрывают пространство всех двоичных векторов длины m . Совершенный код удовлетворяет границе Хемминга (1.86) с равенством.

Пример 1.81. Заметим, что код из примера 1.79 — совершенный. Однако на практике крайне редко удается построение совершенных кодов.

Определение 1.48. Линейный код называют *циклическим*, если для любого кодового слова $\{a_1, a_2, \dots, a_m\}$ циклическая перестановка символов $\{a_m, a_1, \dots, a_{m-1}\}$ также дает кодовое слово.

Пример 1.82. Код из примера 1.79 — циклический.

Для построения линейных циклических кодов потребуется перейти от векторного описания кодов к полиномиальному. Последовательность символов основного алфавита $\{0, 1\}$, составляющих сообщения и кодовые слова, будем интерпретировать как коэффициенты полиномов. Например, считая, что коэффициенты записаны в *порядке возрастания степени*, кодовое слово $(1, 0, 1, 0)$ запишем в виде многочлена $1 + x^2$.

Кодирование сообщения в более «длинное» кодовое слово будет проводиться умножением этого многочлена на другой, что даст в результате многочлен более высокой степени.

Замечательным свойством полиномиального представления кодов является возможность осуществить циклический сдвиг на одну позицию вправо простым умножением кодового многочлена степени $m - 1$ на многочлен x и нахождения остатка от деления на $x^m + 1$.

Пример 1.83. Осуществить единичный правый циклический сдвиг кодового слова $(1, 0, 1, 1)$, используя полиномиальную интерпретацию.

Вектору $(1, 0, 1, 1)$ соответствует полином $1 + x^2 + x^3$. Умножим его на x , что дает $x + x^3 + x^4$. Далее, найдем остаток от деления на $x^4 + 1$, т. е. $x + x^3 + x^4 \pmod{x^4 + 1}$.

Используя алгоритм 1.21 или 1.20, получаем

$$x + x^3 + x^4 = (x^4 + 1) \cdot 1 + (1 + x + x^3).$$

Многочлен остатка $1 + x + x^3$ соответствует вектору $(1, 1, 0, 1)$, который получается из $(1, 0, 1, 1)$ правым циклическим сдвигом на одну позицию.

Процедура кодирования в полиномиальной интерпретации сводится к умножению многочлена-сообщения на подходящий многочлен, называемый *порождающим многочленом* данного кода. Очевидно, что не любой многочлен может служить порождающим.

Пример 1.84. Многочлен $G(x) = x$ не является порождающим. Действительно, все кодовые слова, полученные при умножении на $G(x)$, имеют вид $(0, a_2, a_3, a_4)$ и не удовлетворяют требованию цикличности.

Справедлива следующая теорема¹.

Теорема 1.68. Многочлен $G(x)$ является порождающим многочленом линейного циклического кода длиной n тогда и только тогда, когда $g(x)$ делит $1 + x^n$.

Следствие 1.25. Если $G(x)$ — порождающий многочлен, то его свободный член $g_0 \neq 0$.

¹ Доказательство можно найти в [4].

Доказательство. Если $g_0 = 0$, то многочлен x — делитель $G(x)$ и, следовательно, делитель $1 + x^n$. Противоречие. ■

Из теоремы 1.68 следует, что для получения порождающего многочлена $G(x)$ необходимо разложить на множители $x^n + 1$ и выделить многочлен такой степени, которая соответствует длине кодового слова. Длина кодового слова n и длина исходного сообщения m связаны соотношением

$$n = m + \deg(G(x)).$$

Пусть $C(x)$ обозначает переданное кодовое слово. Это значит, что символами переданного слова были коэффициенты многочлена $C(x)$. Пусть многочлен $V(x)$ обозначает принятое кодовое слово и

$$E(x) = V(x) - C(x). \quad (1.87)$$

Многочлен $E(x)$ в (1.87) называют *многочленом ошибок*. Ненулевые коэффициенты этого многочлена стоят на тех позициях, где в канале произошли ошибки. Определим также понятие *синдромного многочлена* $S(x)$ как остаток от деления $V(x)$ на $G(x)$:

$$S(x) = \langle V(x) \rangle_{G(x)} = \langle E(x) \rangle_{G(x)}. \quad (1.88)$$

Очевидно, что если кодовое слово было передано без помех, то синдром равен нулю. При некоторых условиях на код можно установить взаимно однозначное соответствие между неизвестным многочленом ошибки и вычисляемым на приеме синдромным многочленом. Имеет место теорема.

Теорема 1.69. Пусть d^* — минимальное кодовое расстояние линейного циклического кода. Тогда каждому многочлену ошибок с весом меньше $d^*/2$ соответствует единственный синдромный многочлен.

Доказательство. Предположим, что двум многочленам ошибок $E_1(x)$ и $E_2(x)$ с весами, меньшими $d^*/2$, соответствует один и тот же синдромный многочлен $S(x)$. Тогда согласно (1.87) и (1.88) и учитывая, что $C(x)$ по определению делится на $G(x)$, получаем представления:

$$E_1(x) = Q_1(x)G(x) + S(x), \quad E_2(x) = Q_2(x)G(x) + S(x).$$

Тогда

$$E_1(x) - E_2(x) = (Q_1(x) - Q_2(x))G(x). \quad (1.89)$$

По предположению, вес каждого из многочленов E_1 и E_2 меньше $d^*/2$, поэтому вес их разности меньше d^* . С другой стороны, в правой части

(1.89) стоит по определению некоторое кодовое слово, следовательно, если $Q_1 \neq Q_2$, то вес правой части не менее d^* . Полученное противоречие показывает, что Q_1 и Q_2 равны и, следовательно, равны E_1 и E_2 . ■

Таким образом, задача исправления ошибок сводится к вычислению многочлена ошибки $E(x)$ с наименьшим весом, удовлетворяющего условию

$$S(x) = \langle E(x) \rangle_{G(x)}.$$

При небольших длинах кода согласно теореме 1.69 для каждого многочлена ошибки вычисляется и табулируется соответствующий синдромный многочлен. Такая таблица носит название *таблицы синдромов*.

Вычисляя синдром $S(x)$ по принятому кодовому слову $V(x)$, декодер находит в таблице соответствующий многочлен $E(x)$. При больших длинах кодов такой подход неприменим. Далее в примере 1.86 будет рассмотрен более эффективный алгоритм декодирования.

Найдем все порождающие многочлены для циклического кода длины n . Согласно теореме 1.68 нужно разложить многочлен $1 + x^n$ на простые множители:

$$1 + x^n = F_1(x) F_2(x) \cdots F_l(x), \quad (1.90)$$

где l — число простых сомножителей.

Произведение произвольного подмножества этих множителей даст порождающий многочлен $G(x)$. Если все простые множители в разложении (1.90) различны, то всего имеется $2^l - 2$ различных нетривиальных циклических кодов длиной n (исключаются тривиальные случаи $G(x) = 1 + x^n$ и $G(x) = 1$).

Вопрос о том, какие из них дают коды с большим минимальным расстоянием d^* , решается достаточно сложно и остается вне рамок данной книги¹.

Пример 1.85. Найти порождающий многочлен линейного циклического кода длиной $n = 15$, который осуществляет кодирование сообщений длиной $m = 7$. Затем кодировать сообщение $(0, 1, 1, 0, 1, 1, 0)$.

Для построения требуемого порождающего многочлена нужно найти делитель $x^{15} + 1$ степени $15 - 7 = 8$. Имеем

$$x^{15} + 1 = (1 + x)(1 + x + x^2)(1 + x + x^2 + x^3 + x^4)(1 + x + x^4)(1 + x^3 + x^4),$$

поэтому можно взять

$$G(x) = (1 + x + x^2 + x^3 + x^4)(1 + x + x^4) = 1 + x^4 + x^6 + x^7 + x^8.$$

¹ Подобный анализ проведен в монографии [4].

Так как многочлен $G(x)$ сам является кодовым словом ($G(x) = G(x) \cdot 1$) и $w(G(x)) = 5$, то по теореме 1.67 $d^* \leq 5$. На самом деле в данном случае имеет место равенство. Таким образом, согласно теореме 1.69 данный код исправляет две ошибки.

В полиномиальной интерпретации вектору $(0, 1, 1, 0, 1, 1, 0)$ соответствует многочлен $x + x^2 + x^4 + x^5$. Умножим его на порождающий многочлен $G(x)$. Имеем

$$(x + x^2 + x^4 + x^5)(1 + x^4 + x^6 + x^7 + x^8) = x + x^2 + x^4 + x^6 + x^7 + x^8 + x^9 + x^{13}.$$

Соответствующее кодовое слово: $(0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0)$.

Рассмотрим более эффективный алгоритм декодирования. В нем выполняется циклический сдвиг принятого кодового слова, пока вес остатка от деления не будет равен известному синдрому. Подробное обоснование можно найти также в [4].

Входные данные: многочлен $V(x)$ — принятое слово, $G(x)$ — порождающий многочлен кода, максимальное число исправляемых кодом ошибок равно t .

На выходе получаем исправленное слово $V(x)$ или отказ в декодировании.

Алгоритм 1.26. Полиномиальное декодирование

```

 $S(x) \leftarrow \langle V(x) \rangle_{G(x)}$  // находим синдромный многочлен
for  $i \leftarrow 0, n - 1$  do
   $S_i(x) \leftarrow \langle x^i S(x) \rangle_{G(x)}$ 
  if  $w(S_i) \leq t$  then
     $E(x) \leftarrow \langle x^{n-i} S_i(x) \rangle_{1+x^n}$  // Вычисляем полином ошибки
    Производим исправления в  $V(x)$  с учетом  $E(x)$ 
    Декодирование успешно завершено
  end if
end for
В принятом слове более  $t$  ошибок, декодирование невозможно

```

Пример 1.86. Декодируем слово $(0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0)$, которое было отправлено после кодирования кодом из примера 1.85. Соответствующий многочлен есть $x + x^2 + x^4 + x^6 + x^7 + x^8 + x^{10} + x^{13}$.

Согласно алгоритму 1.26 находим многочлен-синдром:

$$S(x) = \langle x + x^2 + x^4 + x^6 + x^7 + x^8 + x^{10} + x^{13} \rangle_{1+x^4+x^6+x^7+x^8} = 1+x^2+x^4+x^7.$$

Для кодового слова синдром, как известно, равен 0. В данном случае это не так, посланное слово было искажено помехой. В соответствии с описанной

процедурой декодирования будем вычислять

$$S_i(x) = \langle x^i(1 + x^2 + x^4 + x^7) \rangle_{G(x)}$$

для последовательных возрастающих значений $i = 0, 1, \dots, n - 1$, пока не найдем многочлен степени, меньшей или равной двум (число ошибок $t = 2$).

Имеем:

$$\begin{aligned} S_1 &= \langle xs(x) \rangle_{G(x)} = \langle x + x^3 + x^5 + x^8 \rangle_{G(x)} = 1 + x + x^3 + x^4 + x^5 + x^6 + x^7, \\ S_2 &= \langle x^2s(x) \rangle_{G(x)} = \langle x^2 + x^4 + x^6 + x^9 \rangle_{G(x)} = 1 + x + x^2 + x^5, \\ S_3 &= \langle x^3s(x) \rangle_{G(x)} = \langle x^3 + x^5 + x^7 + x^{10} \rangle_{G(x)} = x + x^2 + x^3 + x^6, \\ S_4 &= \langle x^4s(x) \rangle_{G(x)} = \langle x^4 + x^6 + x^8 + x^{11} \rangle_{G(x)} = x^2 + x^3 + x^4 + x^7, \\ S_5 &= \langle x^5s(x) \rangle_{G(x)} = \langle x^5 + x^7 + x^9 + x^{12} \rangle_{G(x)} = 1 + x^3 + x^5 + x^6 + x^7, \\ S_6 &= \langle x^6s(x) \rangle_{G(x)} = \langle x^6 + x^8 + x^{10} + x^{13} \rangle_{G(x)} = 1 + x. \end{aligned}$$

Многочлен $x + 1$ имеет вес 2, поэтому для нахождения многочлена ошибок вычисляем

$$E(x) = \langle x^{15-6}s_6(x) \rangle_{1+x^n} = \langle x^9(x + 1) \rangle_{1+x^{15}} = x^{10} + x^9.$$

Итак, если отправленное кодовое слово имеет не более двух ошибок, то оно было таким

$$(x + x^2 + x^4 + x^6 + x^7 + x^8 + x^{10} + x^{13}) + (x^{10} + x^9) = x + x^2 + x^4 + x^6 + x^7 + x^8 + x^9 + x^{13}.$$

Этот многочлен соответствует слову $(0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0)$. Чтобы восстановить само сообщение, надо разделить кодовое слово на порождающий многочлен $G(x)$ и получить $x + x^2 + x^4 + x^5$, значит, отправленное сообщение было $(0, 1, 1, 0, 1, 1, 0)$.

Справедлива теорема.

Теорема 1.70. Если порождающий многочлен $G(x)$ делится на $1 + x$, то любое кодовое слово имеет четное число ненулевых коэффициентов.

Доказательство. Любое кодовое слово $C(x)$ получается умножением исходного слова $A(x)$ на $G(x)$. Тогда

$$C(x) = c_0 + c_1x + \dots + c_{n-1}x^{n-1} = A(x)G(x) = (1 + x)H(x). \quad (1.91)$$

Подставив в (1.91) $x = 1$, получим $c_0 + c_1 + \dots + c_{n-1} = (1 + 1)H(1) = 0$. ■

Это и есть *код с проверкой на четность*, обнаруживающий любое нечетное число ошибок передачи. Все кодовые слова — «четные».

1.6.6. Код Шеннона — Фано и алгоритм Хаффмана

В предыдущих подразделах на каждый кодовый символ отводилось одинаковое число двоичных разрядов. Клод Шеннон и Роберт Фано (Fano) предложили конструкцию *кода переменной длины*. При таком коде у каждого символа своя длина кодовой последовательности, поэтому необходимо побеспокоиться о том, как определять конец кода отдельного символа. Предлагается такое ограничение на код: *никакая кодовая последовательность не является началом другой кодовой последовательности*. Это свойство называют *свойством префикса*, а код, обладающий таким свойством, — *беспрефиксным кодом*.

Пусть $\{p_1, p_2, \dots, p_n\}$ — частоты появления (использования) кодируемых символов, а $\{s_1, s_2, \dots, s_n\}$ — длины их кодовых последовательностей. Будем стремиться уменьшить среднее число двоичных разрядов, приходящихся на один кодовый символ, т. е. величину $\sum_{i=1}^n p_i s_i$.

Шеннон и Фано предложили метод построения кода, близкого к оптимальному: разбить все символы на две группы с приблизительно одинаковой суммарной частотой появления, коды первой группы начать с нуля, а второй — с единицы. Внутри группы делать то же самое, пока в каждой группе не останется только по одному символу.

Элегантный алгоритм для точного решения этой задачи, основанный на нескольких очевидных свойствах оптимальной кодовой последовательности, предложил Дэвид Хаффман.

Очевидно следующее утверждение.

Лемма 1.13. Если $p_1 \geq p_2 \geq \dots \geq p_n$, то $s_1 \leq s_2 \leq \dots \leq s_n$.

Какими свойствами должен обладать оптимальный код? Посмотрим на самое длинное кодовое слово. Оно должно быть парным, то есть слово, которое отличается в последнем бите, тоже должно быть кодовым словом. Например, если мы какую-то букву кодируем как $b_0 \dots b_n 0$, а как $b_0 \dots b_n 1$ ничего не кодируем, то почему же мы не используем просто битовую последовательность $b_0 \dots b_n$? Ведь получится тоже беспрефиксный код, и будет короче.

Таким образом справедлива теорема.

Теорема 1.71. В предположениях леммы 1.13 справедливо $s_{n-1} = s_n$.

Опишем теперь собственно алгоритм: два символа кодируем 0 и 1, а если символов больше, то соединяем два самых редких символа в один новый символ, решаем получившуюся задачу, а затем вновь разделяем этот новый символ на два, приписав соответственно 0 и 1 к его кодовой последовательности.

Пример 1.87. Пусть имеется семь кодовых символов a, b, c, d, e, f, g с частотами появления соответственно

$$0.22(a), 0.20(b), 0.15(c), 0.13(d), 0.12(e), 0.10(f), 0.08(g).$$

С помощью алгоритма Хаффмана построить код Шеннона — Фано и зашифровать сообщение: $[beda]$.

Воспользуемся алгоритмом Хаффмана. Проводя объединение, получаем:

- 1) $0.22(a), 0.20(b), 0.18(fg), 0.15(c), 0.13(d), 0.12(e)$;
- 2) $0.25(de), 0.22(a), 0.20(b), 0.18(fg), 0.15(c)$;
- 3) $0.33(cfg), 0.25(de), 0.22(a), 0.20(b)$;
- 4) $0.42(ab), 0.33(cfg), 0.25(de)$;
- 5) $0.58(cfgde), 0.42(ab)$.

Последовательность объединения схематично проиллюстрирована на рисунке 1.10. Алгоритм работает «снизу вверх» (движение по стрелкам). Объединяемые наборы символов отмечены серым цветом и пунктирными линиями. При объединении символы с большей частотностью отмечены светло серым, а с меньшей — темно-серым цветом.

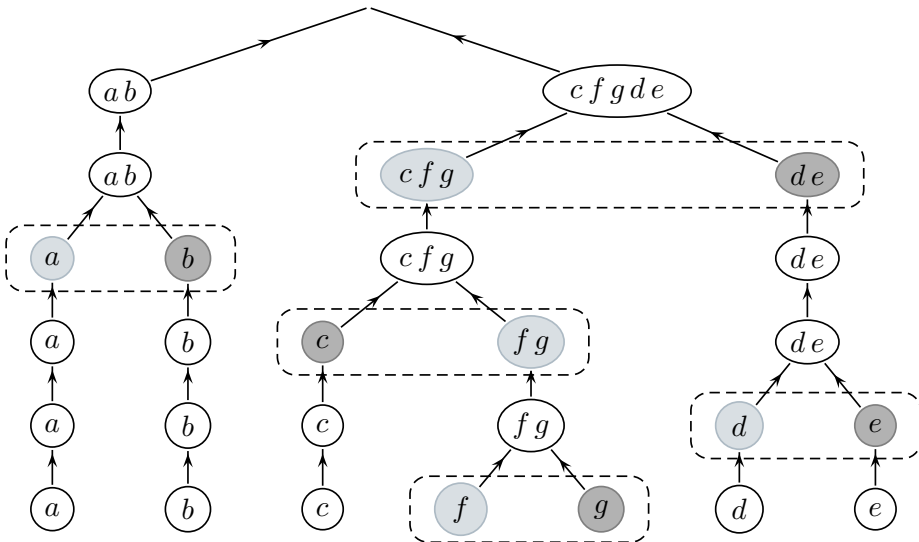


Рис. 1.10.

Расщепляем символы и приписываем 0 и 1 справа к кодовым последовательностям. Для определенности символу с большей частотностью приписываем 1, с меньшей — 0. Очевидно, что при обратном правиле получится инверсный код.

- 1) $0(ab), 1(cfgde)$;
- 2) $0(ab), 11(cfg), 10(de)$;
- 3) $01(a), 00(b), 11(cfg), 10(de)$;
- 4) $01(a), 00(b), 110(c), 111(fg), 10(de)$;
- 5) $01(a), 00(b), 110(c), 111(fg), 101(d), 100(e)$;
- 6) $01(a), 00(b), 110(c), 1111(f), 1110(g), 101(d), 100(e)$.

Таким образом кодовая таблица имеет вид (таблица 1.38):

Таблица 1.38.

a	b	c	d	e	f	g
01	00	110	101	100	1111	1110

Шифруем исходное сообщение: 0010010101. Согласно кодовой таблице 1.38 легко произвести дешифровку.

Замечание 1.51

Рассмотренный алгоритм является примером жадного алгоритма. Жадный алгоритм (англ. *Greedy algorithm*) — это алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

1.6.7. Лексикографический порядок. Генерация подмножеств

Рассмотрим задачу генерации всех k -элементных подмножеств данного n -элементного множества $X = \{x_1, \dots, x_n\}$. Очевидно, что всего их C_n^k . Не умаляя общности, будем считать, что $X = \{1, \dots, n\}$. Определим упорядоченность k -элементных подмножеств на основе упорядоченности элементов.

Определение 1.49. Рассмотрим два k -элементных подмножества:

$$A = \{x_1, \dots, x_k\}, B = \{y_1, \dots, y_k\}.$$

Говорят, что:

- 1) подмножество A лексикографически предшествует подмножеству B , если найдется индекс $m \leq k$ такой, что

$$x_i = y_i, x_m < y_m, \quad 1 \leq i < m; \quad (1.92)$$

2) подмножество A *антилексикографически* предшествует подмножеству B , если существует индекс $m \leq k$ такой, что

$$x_i = y_i, \quad x_m > y_m, \quad m < i \leq k.$$

Если в (1.92) выполнено $x_m > y_m$, то такой порядок называют *обратным лексикографическим*.

Замечание 1.52

В общем случае будем считать, что для каждого индекса $1 \leq i \leq k$ определено сравнение элементов x_i и y_i . Это позволяет лексикографически сравнивать последовательности, состоящие из элементов разных типов.

Пример 1.88. Если в качестве элементов X взять буквы алфавита, то слова длиной k появляются в словаре в лексикографическом порядке.

В качестве примера построим алгоритм генерации k -элементных подмножеств X в лексикографическом порядке. Проведем некоторые предварительные рассуждения. Очевидно, что первое подмножество — это $\{1, 2, \dots, k\}$, а за подмножеством $\{a_1, a_2, \dots, a_k\}$ следует подмножество

$$\{b_1, b_2, \dots, b_k\} = \{a_1, a_2, \dots, a_{p-1}, a_p + 1, a_p + 2, \dots, a_p + k - p + 1\},$$

где p — максимальный индекс, для которого $b_k = a_p + k - p + 1 \leq n$.

Очередное подмножество получается из предыдущего заменой последних элементов на идущие подряд целые числа так, чтобы последний элемент не превосходил n , а первый изменяемый элемент был на 1 больше, чем соответствующий элемент предыдущего подмножества.

Таким образом, индекс p , начиная с которого следует проводить изменения, определяется по значению a_k . Если $a_k < n$, то следует изменить только a_k , и при этом $p = k$. Если же $a_k = n$, то нужно уменьшать индекс $p = p - 1$, увеличивая тем самым длину изменяемой части подмножества.

Запишем теперь сам алгоритм.

Алгоритм 1.27. Генерация k -элементных подмножеств

```

for  $i \leftarrow 1, k$  do // первое подмножество
   $a_i \leftarrow i$ 
end for
 $p \leftarrow k$ 
while  $p \geq 1$  do
  if  $a_k = n$  then
     $p \leftarrow p - 1$ 
  else

```

```

    p ← k
  end if
  if p ≥ 1 then // p могло стать меньше 1 на предыдущем шаге
    for i ← p, k do
      ai ← ap + i - p + 1
    end for
  end if
end while

```

Пример 1.89. Пусть $X = \{1, 2, 3, 4, 5, 6\}$, $k = 4$. Протокол работы алгоритма 1.27 дан в таблице 1.39.

Таблица 1.39.

Шаг	p	Текущее разбиение	Шаг	p	Текущее разбиение
1	4	{1, 2, 3, 4}	9	3	{1, 3, 5, 6}
2	4	{1, 2, 3, 5}	10	2	{1, 4, 5, 6}
3	4	{1, 2, 3, 6}	11	1	{2, 3, 4, 5}
4	3	{1, 2, 4, 5}	12	4	{2, 3, 4, 6}
5	4	{1, 2, 4, 6}	13	3	{2, 3, 5, 6}
6	3	{1, 2, 5, 6}	14	2	{2, 4, 5, 6}
7	2	{1, 3, 4, 5}	15	1	{3, 4, 5, 6}
8	4	{1, 3, 4, 6}		0	Генерация закончена

1.6.8. Перечислительная комбинаторика

Рассмотрим комбинаторные задачи следующего вида: нужно пронумеровать все комбинации с заданным свойством и найти алгоритм, который по заданному номеру будет строить соответствующую комбинацию, и алгоритм, который будет решать обратную задачу — по заданной комбинации вычислять ее номер. Эту же задачу можно назвать задачей перечисления всех комбинаций.

В предыдущем подразделе был приведен частный случай решения такой задачи (алгоритм 1.27). Разберем общие приемы, позволяющие перечислять подмножества данного множества, элементы декартова произведения, перестановки, сочетания.

1.6.8.1. Перечисление подмножеств

Пусть задано множество, содержащее n элементов: $M = \{m_1, \dots, m_n\}$.

Требуется найти алгоритм, генерирующий все подмножества данного множества.

Для решения задачи перечисления подмножеств используем часто применяемый в комбинаторике прием — конструирование взаимно однозначного соответствия для того, чтобы свести одну комбинаторную задачу к другой.

Рассмотрим двоичные наборы, содержащие n цифр. Каждый такой набор можно интерпретировать как шифр некоторого подмножества. Единица на k -м месте означает, что элемент m_k включается в соответствующее подмножество, а ноль — что элемент не включается.

Пример 1.90. У множества из трех элементов $M = \{a, b, c\}$ имеется восемь подмножеств, два из них — пустое подмножество, само множество M и еще шесть так называемых собственных подмножеств

$$\{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}.$$

Указанные собственные подмножества множества из трех элементов шифруются так: 100, 010, 001, 110, 101, 011. Пустое подмножество имеет шифр 000, а само множество — шифр 111. Если эти наборы интерпретировать как двоичные записи чисел, то получим соответственно 4, 2, 1, 6, 5, 3 и 0, 7, т. е. все целые числа от 0 до 7. Это дает идею решения всех поставленных задач.

Число подмножеств множества из n элементов равно 2^n . Генерация подмножеств делается так: надо, начиная с $i = 0$, переводить i в двоичную запись, которая определяет очередное подмножество, а затем увеличивать i на единицу вплоть до $2^n - 1$.

Алгоритм 1.28. Генерация всех подмножеств n -элементного множества
(1)

for $i \leftarrow 1, 2^n - 1$ **do**

 перевести i в двоичную запись

 дополнить эту запись спереди нулями, чтобы число цифр стало равным n ,

 построить по двоичному набору подмножество

end for

Замечание 1.53

При перечислении можно оперировать двоичной арифметикой, тогда перевод номера подмножества в двоичную запись станет излишним.

Нахождение десятичного номера подмножества осуществляется алгоритмом перевода числа из двоичной системы в десятичную, а нахождение двоичного набора, описывающего данное подмножество, по его десятичному номеру — переводом из десятичной в двоичную.

Задача перечисления подмножеств станет более интересной, если наложить дополнительное условие на их порядок. Например, естественным ограничением будет такое: два последовательных подмножества должны отличаться ровно одним элементом.

Пример 1.91. На столе лежат четыре разноцветных шара. За один ход можно взять или положить ровно один шар. Как за наименьшее число ходов перебрать все возможные подмножества этих шаров.

Понятно, что если найти алгоритм, позволяющий после каждого взятия или добавления шара получать новое подмножество, то такая стратегия даст наименьшее число ходов. Поскольку всего подмножеств (включая пустое) $2^4 = 16$, то достаточно будет 15 ходов (если учесть, что уже до первого хода одно из подмножеств — само множество — уже представлено).

Приведем протокол решения этой задачи (таблица 1.40), сравнивая двоичные наборы, определяющие текущие подмножества, с теми, которые дает предыдущий алгоритм (поскольку мы начали перебор не с пустого подмножества, а, наоборот, с самого множества, будем обозначать нулем оставленный шар, а единицей — взятый).

Можно заметить следующую закономерность, когда при обычном перечислении двоичными числами в k -м разряде 0 сменяет 1, то с k -м шаром совершают операцию переключивания: если он лежал на столе — снимают, если его не было на столе — добавляют.

Таблица 1.40.

Номера шаров, лежащих на столе	Комментарий к очередному действию	Двоичный код подмножества шаров B	Двоичный номер шага A	Комментарий к правилу изменения B по изменению A
1234	Исходная комбинация	0000	0000	
123	Сняли 4-й шар	0001	0001	В 4-м разряде 0 заменили на 1
12	Сняли 3-й шар	0011	0010	В 3-м разряде 0 заменили на 1
124	Добавили 4-й шар	0010	0011	В 4-м разряде 0 заменили на 1
14	Сняли 2-й шар	0110	0100	Во 2-м разряде 0 заменили на 1

Продолжение таблицы 1.40

Номера шаров, лежащих на столе	Комментарий к очередному действию	Двоичный код подмножества шаров B	Двоичный номер шага A	Комментарий к правилу изменения B по изменению A
1	Сняли 4-й шар	0111	0101	В 4-м разряде 0 заменили на 1
13	Добавили 3-й шар	0101	0110	В 3-м разряде 0 заменили на 1
134	Добавили 4-й шар	0100	0111	В 4-м разряде 0 заменили на 1
34	Сняли 1-й шар	1100	1000	В 1-м разряде 0 заменили на 1
3	Сняли 4-й шар	1101	1001	В 4-м разряде 0 заменили на 1
	Сняли 3-й шар	1111	1010	В 3-м разряде 0 заменили на 1
4	Добавили 4-й шар	1110	1011	В 4-м разряде 0 заменили на 1
24	Добавили 2-й шар	1010	1100	Во 2-м разряде 0 заменили на 1
2	Сняли 4-й шар	1011	1101	В 4-м разряде 0 заменили на 1
23	Добавили 3-й шар	1001	1110	В 3-м разряде 0 заменили на 1
234	Добавили 4-й шар	1000	1111	В 4-м разряде 0 заменили на 1

Пусть массив A_i представляет двоичные номера при обычном переборе, а B_i — при переборе с заданными выше ограничениями. Пусть в начальный момент оба массива заполнены нулями. Тогда, двигаясь сначала справа налево по разрядам двоичного набора A_i , надо найти первый нулевой разряд, запомнить его номер k и записать в него 1 : $A_k := 1$, а все разряды правее k -го заполнить нулями.

Затем надо поменять k -й разряд в массиве B_i и вывести очередное подмножество. Все это надо повторять в цикле, пока в массиве A_i не будут стоять одни единицы. Таким образом, получаем новый алгоритм генерации подмножеств.

Алгоритм 1.29. Генерация всех подмножеств n -элементного множества
 (2)

```

for  $i \leftarrow 1, n$  do
   $A_i \leftarrow 0$ 
   $B_i \leftarrow 0$ 
end for
repeat
   $k \leftarrow n$ 
  while  $A_k \neq 0$  do // ищем первый нулевой разряд
     $k \leftarrow k - 1$ 
  end while
  if  $k > 0$  then
     $A_k \leftarrow 1$ 
    for  $i \leftarrow k + 1, n$  do // все разряды правее  $k$ -го заполняем нулями
       $A_i \leftarrow 0$ 
    end for
     $B_k \leftarrow B_k + 1 \pmod{2}$ 
    return  $B$  // вывод очередного подмножества  $B$ 
  end if
until  $k = 0$ 

```

Для того чтобы обосновать корректность алгоритма 1.29, посмотрим на него с «рекурсивной» точки зрения.

Действительно, анализируя пример, нетрудно заметить, что первые восемь подмножеств из шестнадцати содержат 1-й шар, оставшиеся восемь — нет. Так же каждую из этих восьмерок можно разделить на четверки по признаку, содержат ли они 2-й шар или нет (заметим, что, если в первой восьмерке этот шар содержат первые четыре подмножества, то во второй — последние четыре).

Затем в каждой четверке можно сравнить двойки, которые будут отличаться наличием 3-го шара. И наконец, каждая двойка разбивается на отдельные подмножества, которые отличаются наличием 4-го шара.

Получается, что любые два подмножества отличаются друг от друга. То, что два соседних (относительно порядка перебора) подмножества отличаются ровно одним элементом, объясняется так. Если два подмножества принадлежат одной «двойке», то они отличаются только 4-м шаром. Если они принадлежат одной четверке так, что одно является вторым для первой двойки, а второе — первым для второй, то они отличаются только 3-м шаром, так как 4-й шар будет в обоих подмножествах и т. д.

То, что номера разрядов, которые соответствуют перемещаемым шарам, получаются из алгоритма добавления 1 к двоичному числу, следует из того,

что первые восемь двоичных чисел имеют 0 в старшем разряде, а следующие восемь — 1. Аналогичное утверждение можно сделать про четверки и т. д.

Замечание 1.54

Более подробно вопросы, связанные с перечислением подмножеств в примере 1.91 и алгоритме 1.29, будут рассмотрены в разделе 1.6.9.

Задача 1.17. Решает ли следующий алгоритм 1.30 поставленную задачу о переборе всех подмножеств. Ответ обосновать.

Алгоритм 1.30. Генерация всех подмножеств n -элементного множества
(3)

```

while число сгенерированных подмножеств меньше  $2^n$  do
  for  $i \leftarrow n, 1$  do
    if перемещение  $i$ -го шара не приводит к уже встречавшейся комбинации then
      переместить шар (снять или добавить)
    end if
  end for
end while

```

Пример 1.92. Для приведенного выше примера 1.91 последовательность подмножеств, генерируемая алгоритмом 1.30, получается такой:

1234, 123, 12, 1, \emptyset , 4, 34, 234, 23, 2, 24, 124, 14, 134, 13, 3.

1.6.8.2. Перечисление элементов декартова произведения множеств

Пусть нам нужно перечислить элементы множества $M = M_1 \times \dots \times M_n$, где

$$M_1 = \{a_1, a_2, \dots, a_{m_1}\}, M_2 = \{b_1, b_2, \dots, b_{m_2}\}, \dots, M_n = \{x_1, x_2, \dots, x_{m_n}\}.$$

Заметим, что в комбинаторике проще работать с номерами элементов, забывая о том, какую природу имеют эти элементы (буквы, множества, фигуры и пр.). Так и будем делать в дальнейшем. Тогда

$$M = \{(i_1, i_2, \dots, i_n) \mid 0 \leq i_1 < m_1, 0 \leq i_2 < m_2, \dots, 0 \leq i_n < m_n\}. \quad (1.93)$$

Пример 1.93. Элемент декартова произведения, описываемый набором $(2, 0, \dots, 3)$ для обозначений (1.93), будет (a_3, b_1, \dots, x_4) .

В соответствии с определением набора декартова произведения можно генерировать с помощью n вложенных циклов.

Алгоритм 1.31. Перечисление декартова произведения

```

for  $i_1 \leftarrow 0, m_1 - 1$  do
  for  $i_2 \leftarrow 0, m_2 - 1$  do
    .....
    for  $i_n \leftarrow 0, m_n - 1$  do
      return набор  $(i_1, \dots, i_n)$ 
    end for
  .....
end for
end for

```

По этому алгоритму легко определить, каким по счету будет сгенерирован набор (i_1, i_2, \dots, i_n) .

$$N(i_1, i_2, \dots, i_n) = m_2 m_3 \cdots m_n i_1 + m_3 m_4 \cdots m_n i_2 + \cdots + m_n i_{n-1} + i_n.$$

 **Замечание 1.55**

Номера $N(i_1, i_2, \dots, i_n)$ также будут начинаться с 0. Для привычной нумерации с 1 к полученному выражению надо добавить 1. Полученное выражение можно рассматривать как представление числа N в системе с переменным основанием. Перевод числа в такую систему — построение набора по его номеру — можно осуществить последовательными делениями с остатком.

Представим N следующим образом:

$$N(i_1, i_2, \dots, i_n) = (\cdots (i_1 m_2 + i_2) m_3 + i_3) m_4 + \cdots + i_{n-1}) m_n + i_n.$$

Тогда $i_n = N \pmod{m_n}$, а выражение в скобках перед m_n равно $N \div m_n$. Алгоритм вычисления всех элементов набора будет такой.

Алгоритм 1.32. Вычисление элементов набора (i_1, i_2, \dots, i_n)

```

for  $k \leftarrow n, 1$  do
   $i_k \leftarrow N \pmod{m_k}$ 
   $N \leftarrow N \div m_k$ 
end for

```

 **Замечание 1.56**

В алгоритме 1.32 предполагается знание длины набора n . Если перенумеровать элементы набора от последнего к первому, то значение n не понадобится. В следующем алгоритме этот прием использован.

На основе этого алгоритма алгоритм перечисления элементов декартова произведения можно записать, используя всего два вложенных цикла:

Алгоритм 1.33. Перечисление декартова произведения

```

return (0, ..., 0)
for  $N \leftarrow 1, m_1 m_2 \dots m_n - 1$  do
   $k \leftarrow 1$ 
   $R \leftarrow N$ 
  while  $R > 0$  do
     $i_k \leftarrow R \pmod{m_k}$ 
     $R \leftarrow R \div m_k$ 
     $k \leftarrow k + 1$ 
  end while
  return ( $i_n, \dots, i_1$ )
end for

```

1.6.8.3. Перечисление перестановок

Рассмотрим задачу нумерации перестановок и составим алгоритм, который по перестановке определяет ее номер, а также алгоритм, который по номеру определяет соответствующую перестановку.

Расположим наборы в лексикографическом порядке и пронумеруем их. Нумерацию будем начинать с 0.

Пример 1.94. Для $n = 4$ имеем $4! = 24$ перестановки (таблица 1.41).

Таблица 1.41.

0	1234	4	1423	8	2314	12	3124	16	3412	20	4213
1	1243	5	1432	9	2341	13	3142	17	3421	21	4231
2	1324	6	2134	10	2413	14	3214	18	4123	22	4312
3	1342	7	2143	11	2431	15	3241	19	4132	23	4321

Построим сначала алгоритм перехода от одной перестановки к следующей за ней в лексикографическом порядке. Сравнив, например, переход от 5-й к 6-й перестановке в предыдущем примере, можно сформулировать такой алгоритм.

Алгоритм 1.34. Переход от одной перестановки к следующей

```

Находим упорядоченный по убыванию «остаток» перестановки наибольшей длины
if если остаток» не совпадает со всей перестановкой then

```

Элемент, предшествующий «остатку», поменяем с наименьшим элементом «остатка», *большим его*

Элементы «остатка» упорядочиваем в порядке возрастания

end if

Замечание 1.57

На шаге 2 алгоритма 1.34 в «остатке» всегда найдется элемент для обмена (большой, чем элемент, предшествующий «остатку») В противном случае элемент, предшествующий «остатку», сам был членом «остатка».

Пример 1.95. Какая перестановка идет следующей за перестановкой 61287543?

Искомый «остаток» — 87543, его наименьший элемент — 3. Меняем его с элементом, предшествующим «остатку», — 2 и упорядочиваем «остаток» по возрастанию. Получаем 61324578.

Задача 1.18. Предложите алгоритм перехода к следующей перестановке для обратного лексикографического порядка.

Из приведенной таблицы для перечисления перестановок из четырех элементов в лексикографическом порядке (см. таблицу 1.41) можно заметить и другие закономерности, например, что перестановки в разных столбиках (24 перестановки были разбиты на четыре столбика по 6 элементов) начинаются с разных цифр, а в одном — с одной.

Это напоминает систему счисления. Действительно, посчитаем, какой по счету идет перестановка (i_1, i_2, \dots, i_n) в лексикографическом порядке.

До этой перестановки стоят $(i_1 - 1)(n - 1)!$ перестановок, у которых первый элемент меньше (а остальные переставляются всеми $(n - 1)!$ возможными способами). Далее, перед этой перестановкой стоят все те перестановки, у которых первый элемент такой же, а второй меньше. Таких перестановок будет $(i_2 - 1)(n - 2)!$ либо $(i_2 - 2)(n - 2)!$, если первая цифра меньше второй. В общем случае полученный результат можно записать в форме следующего утверждения.

Лемма 1.14. Номер N перестановки (i_1, i_2, \dots, i_n) вычисляется по формуле

$$N = (i_1 - 1)(n - 1)! + (i_2 - 1 - k_2)(n - 2)! + \dots + (i_{n-1} - 1 - k_{n-1})1! + i_n - 1 - k_n, \quad (1.94)$$

где k_j — число элементов, меньших i_j и стоящих левее его.

Пример 1.96. Какой по счету идет перестановка 3412?

$$N = 2 \times 3! + (3 - 1) \times 2! + 0 \times 1! + (1 - 1) = 12 + 4 + 0 + 0 = 16.$$

Формулу (1.94) можно записать изящнее, используя факториальную систему счисления.

Теорема 1.72. При лексикографическом упорядочении перестановок, номер перестановки (i_1, i_2, \dots, i_n) вычисляется по формуле

$$N = a_1(n-1)! + a_2(n-2)! + \dots + a_{n-1}1!, \quad (1.95)$$

где a_j — количество элементов, меньших i_j и стоящих правее его, причем $0 \leq a_j \leq n-j$.

Доказательство. В формуле (1.94) обозначим коэффициент перед факториалом $i_j - 1 - k_j$ через a_j . Его можно интерпретировать как количество элементов, меньших i_j и стоящих правее него.

Действительно, всего существует $i_j - 1$ натуральное число, меньшее i_j , все они входят в перестановку, а k_j из них стоят левее. Значит, оставшиеся $i_j - 1 - k_j$ элементов стоят правее.

Неотрицательность a_j следует из определения. Так же объясняется и неравенство $a_j \leq n-j$: количество элементов, меньших i_j и стоящих правее него, не может быть больше $n-j$, так как равенство достигается тогда, когда все элементы, стоящие правее i_j , меньше него. Это и доказывает теорему. ■

Замечание 1.58

В более привычных для позиционных систем обозначениях можно записать:

$$N = b_{n-1}(n-1)! + b_{n-2}(n-2)! + \dots + b_1! = (b_{n-1}b_{n-2} \dots b_1)!,$$

где $b_j = a_{n-j}$.

Следствие 1.26. Если перестановки пронумерованы в лексикографическом порядке, то для отыскания перестановки по ее номеру надо представить номер в факториальной системе счисления, а затем восстановить перестановку, учитывая, что j -я цифра показывает число элементов, меньших j -го элемента и стоящих правее его.

Пример 1.97. Найти перестановку, определяемую номером в факториальной системе счисления $(230211)_!$.

Первый элемент набора 2, значит, в перестановке правее него стоят два элемента, меньших его. Тогда искомый элемент — 3.

На втором месте стоит 3, значит, в перестановке правее него стоят три элемента, меньших его. Тогда искомый элемент — 5 (а не 4, так как 3 стоит на первом месте).

Весь ход решения отражен в таблице 1.42.

Ответ: 3516472.

Таблица 1.42.

Факториальное представление номера перестановки	Выбор элемента перестановки	Элементы перестановки
230211	7654321	3
230211	765421	5
230211	76421	1
230211	7642	6
230211	742	4
230211	72	7
230211 \emptyset	2	2

Рассмотрим решение обратной задачи для примера 1.97 — по перестановке определить ее номер.

Пример 1.98. Дана перестановка 3516472. Определить ее номер N в факториальной системе счисления.

Согласно теореме 1.72 строим факториальную запись перестановки:

$$(230211)_1.$$

Применяя формулу (1.95) и схему Горнера для факториальной записи, получаем

$$\begin{aligned} N &= 1 \cdot 1! + 1 \cdot 2! + 2 \cdot 3! + 0 \cdot 4! + 3 \cdot 5! + 2 \cdot 6! = \\ &= (((((6 \cdot 2 + 3)5 + 0)4 + 2)3 + 1)2 + 1 = 1815. \end{aligned}$$

Рассмотрим еще один пример.

Пример 1.99. Найдем перестановку по ее номеру 3973. Получим из номера факториальную запись перестановки. Производя последовательно деления на $2, 3, \dots, n$, имеем:

$$\begin{aligned} 3973 &= 1986 \cdot 2 + 1, \\ 1986 &= 662 \cdot 3 + 0, \\ 662 &= 165 \cdot 4 + 2, \\ 165 &= 33 \cdot 5 + 0, \\ 33 &= 5 \cdot 6 + 3, \\ 5 &= 0 \cdot 7 + 5. \end{aligned}$$

Тем самым, факториальная запись искомой перестановки — 530201. Отсюда получаем перестановку 6415273.

Задача 1.19. Перестановки n элементов упорядочены в лексикографическом порядке. Постройте алгоритмы:

- 1) нахождения номера перестановки,
- 2) нахождения перестановки по ее номеру.

Приведем решение для п. 2. Для реализации алгоритма потребуются две вспомогательные процедуры.

Алгоритм 1.35. Генерация перестановок

// построение факториальной записи числа: $i \rightarrow (a_1, \dots, a_{n-1})$

procedure RECORD(i, n, a)

Входные данные:

i — заданное число, n — число элементов перестановки

Выход:

$a = (a_1, \dots, a_{n-1})$ — факториальная запись, начиная с младших разрядов

$t \leftarrow i$

for $k \leftarrow 2, n$ **do**

$a_{k-1} \leftarrow t \bmod k$

$t \leftarrow t \div k$

end for

end procedure

// построение перестановки по факториальной записи: $(a_1, \dots, a_{n-1}) \rightarrow (c_1, \dots, c_{n-1})$

procedure PERMUTATION(n, a, c)

Входные данные:

n — число элементов перестановки, a — факториальная запись

Выход:

$c = (c_1, \dots, c_{n-1})$ — перестановка

// Инициализация: заполняем рабочий массив b

for $k \leftarrow 1, n$ **do**

$b_k \leftarrow k$

end for

for $k \leftarrow 1, n-1$ **do**

// построение очередного элемента искомой перестановки c_k ; учитывается, что нумерация элементов факториальной записи велась с конца

$c_k \leftarrow b_{a_{n-k}+1}$

for $j \leftarrow a_{n-k} + 1, n - k$ **do**

// удаление из состава элементов перестановки того, который был использован на текущем шаге, — сдвиг элементов влево для заполнения освобожденного места

```

     $b_j \leftarrow b_{j+1}$ 
  end for
end for
end procedure

```

// Основной алгоритм

```

for  $i \leftarrow 1, n! - 1$  do
  строим факториальную запись числа  $i$  — procedure RECORD( $i, n, a$ )
  строим перестановку по факториальной записи —
    procedure PERMUTATION( $n, a, c$ )
end for

```

Задача 1.20. Решите задачу 1.19 для обратного лексикографического порядка.

Для перечисления перестановок также можно поставить задачу на перечисление с ограничениями: две последующие перестановки должны получаться друг из друга перестановкой двух соседних элементов.

Рассмотрим идею такого алгоритма на примере перечисления перестановок четырех элементов: [1 2 3 4]:

1) возьмем последний элемент перестановки и будем попарными обмена-ми «двигать» его влево: [1 2 4 3], [1 4 2 3], [4 1 2 3];

2) после того как элемент 4 достиг левой границы набора, сдвинем элемент 3, стоящий на последнем месте, влево, обменяв с предыдущим: [4 1 3 2];

3) теперь снова начнем «движение» элемента 4, но в обратную сторону: [1 4 3 2], [1 3 4 2], [1 3 2 4];

4) теперь сдвинем элемент 3 еще на 1 влево и, получив перестановку [3 1 2 4], вновь повторим «движение» элемента 4: [3 1 4 2], [3 4 1 2], [4 3 1 2];

5) теперь (так как элемент 3 достиг своего левого положения) начнем «движение» элемента 2 влево, поменяв с предыдущим [4 3 2 1];

6) далее начнем «движение» элемента 4: [3 4 2 1], [3 2 4 1], [3 2 1 4];

7) затем опять продвинем элемент 3 вправо, получив [2 3 1 4], а затем снова начнем «движение» элемента 4: [2 3 4 1], [2 4 3 1], [4 2 3 1];

8) затем опять продвинем элемент 3 вправо, получив [4 2 1 3], а затем снова начнем «движение» элемента 4: [2 4 1 3], [2 1 4 3], [2 1 3 4].

Получены все перестановки четырех элементов.

1.6.9. Бинарный код Грея

Рассмотрим в общем виде идею кодирования, частный случай которой был рассмотрен при перечислении перестановок в примере 1.91.

Определение 1.50. Кодом Грея называется такая система нумерования неотрицательных чисел, когда коды двух соседних чисел отличаются ровно в одном бите. Код числа n обозначается $G(n)$. Этот код был изобретен Фрэнком Греем (Frank Gray) в 1953 г.

Замечание 1.59

В современной технике коды Грея активно используются для минимизации ошибок при преобразовании аналоговых сигналов в цифровые (например, в датчиках). Отметим, что коды Грея и были открыты в связи с этим применением.

Пример 1.100. Для чисел длиной 3 бита имеем такую последовательность кодов Грея:

$$000, 001, 011, 010, 110, 111, 101, 100.$$

Например, $G(4) = 6$.

Рассмотрим теперь обобщение алгоритма 1.29.

✓ **Нахождение кода Грея числа N .** Рассмотрим биты числа N и биты числа $G(N)$. Заметим, что i -й бит $G(N)$ равен единице только в том случае, когда i -й бит N равен единице, а $i+1$ -й бит равен нулю, или наоборот (i -й бит равен нулю, а $i+1$ -й равен единице). Таким образом, имеем

$$G(N) = N \oplus \frac{N}{2}$$

Рассмотрим обратную задачу: по $G(N)$ восстановить число N .

✓ **Нахождение обратного кода Грея.** Идем от старших битов к младшим (нумерация от 1 до k). Получаем такие соотношения между битами N_i числа N и битами G_i числа $G(N)$:

$$\left\{ \begin{array}{l} N_k = G_k, \\ N_{k-1} = G_{k-1} \oplus N_k = G_k \oplus G_{k-1}, \\ N_{k-2} = G_{k-2} \oplus N_{k-1} = G_k \oplus G_{k-1} \oplus G_{k-2}, \\ N_{k-3} = G_{k-3} \oplus N_{k-2} = G_k \oplus G_{k-1} \oplus G_{k-2} \oplus G_{k-3}, \\ \dots = \dots \dots \dots \end{array} \right.$$

1.6.9.1. Код Грея и задача о Ханойских башнях

Коды Грея применяются в решении задачи о Ханойских башнях.

Историческая справка

Первоначально эта игра была предложена французским математиком Эдуардом Люка в 1883 г. для трех колец.

✓ **Постановка задачи.** Даны три стержня, на один из которых наизаны n колец, причем кольца отличаются размером и лежат меньшее на большем. Задача состоит в том, чтобы перенести пирамиду из n колец за наименьшее число ходов на другой стержень. За один раз разрешается переносить только одно кольцо, причем нельзя класть большее кольцо на меньшее.

✓ **Решение.** Для представления перемещений n колец будем использовать n -разрядный код Грея.

Будем двигаться по кодам Грея по возрастанию:

$$G(0) = (\underbrace{0, \dots, 0}_n), \dots, G(2^n - 1) = (\underbrace{1, \dots, 1}_n).$$

Каждому биту i текущего кода Грея поставим в соответствие диск i (младшему биту соответствует наименьший по размеру диск, а старшему биту — наибольший).

Применим тот же подход, что и в задаче 1.91 о перечислении подмножеств шаров на столе. Так как на каждом шаге изменяется ровно один бит, будем считать, что изменение бита i соответствует перемещению диска i . Заметим, что для всех дисков, кроме наименьшего, на каждом шаге имеется ровно один вариант хода (за исключением стартовой и финальной позиций). Для наименьшего диска всегда имеются два варианта хода, однако имеется стратегия выбора хода, всегда приводящая к ответу.

Обозначим f — стартовый стержень, t — финальный стержень, r — оставшийся стержень. Тогда последовательность перемещений наименьшего диска имеет вид

$$\begin{cases} f \rightarrow t \rightarrow r \rightarrow f \rightarrow t \rightarrow r \rightarrow \dots & n - \text{нечетно,} \\ f \rightarrow r \rightarrow t \rightarrow f \rightarrow r \rightarrow t \rightarrow \dots & n - \text{четно.} \end{cases}$$

Очевидно, что для решения задачи потребуется $2^n - 1$ перемещений колец.

Историческая справка

Одно из любопытных применений кода Грея связано с театром. Это код Беккета — Грея, он получил свое название в честь ирландского драматурга Сэмюэля Беккета. Одна из его пьес была написана для четырех актеров и состояла из 16 актов. В конце каждого акта один из четырех актеров выходил на сцену или же уходил с нее. Пьеса начиналась на пустой сцене, и Беккет хотел, чтобы каждое подмножество актеров появлялось ровно один раз. Очевидно, что множество актеров на сцене может быть представлено в виде двоичного кода Грея для $n = 4$.

Драматург, однако, добавил дополнительное условие в сценарий: чтобы со сцены уходил всегда тот из актеров, кто находился на ней дольше остальных. Беккет не смог найти решение для своей пьесы. Позднее было доказано, что такие коды существуют для $n = 2, 5, 6, 7, 8$ и не существуют для $n = 3, 4$.

1.6.10. Числа Стирлинга первого и второго родов

Определение 1.51. Числом Стирлинга второго рода (обозначают $S_{n,k}$) называют число разбиений n -элементного множества на k непересекающихся непустых подмножеств.

Историческая справка

Джеймс Стирлинг (1692–1770) — шотландский математик, член Лондонского королевского общества. Наиболее важный его труд — «Разностный метод» (1730), где Стирлинг впервые дал асимптотическое разложение логарифма гамма-функции (так называемый ряд Стирлинга), рассмотрел бесконечные произведения. Некоторые из открытий Стирлинга были позднее сделаны Л. Эйлером в его более общих исследованиях.

Очевидно, что $S_{n,k} = 0$ для $k > n$. Положим $S_{0,0} = 1$.

Лемма 1.15. Для $S_{n,k}$ справедливы следующие свойства:

- 1) $S_{n,0} = 0, n > 0$;
- 2) $S_{n,n} = 1, n \geq 0$;
- 3) $S_{n,k} = S_{n-1,k-1} + kS_{n-1,k}, 0 < k < n$.

Доказательство. Свойства 1 и 2 очевидны. Установим свойство 3. Рассмотрим множество всех разбиений n -элементного множества $\{1, 2, \dots, n\}$ на k непересекающихся подмножеств. Эти разбиения распадаются на две группы:

- разбиения, которые содержат $\{n\}$ как одноэлементное множество;
- разбиения, где n содержится в составе по крайней мере двухэлементных множеств.

Для первой группы существует $S_{n-1,k-1}$ вариантов разбиений, т. е. разбиений множества $\{1, \dots, n-1\}$ на $k-1$ непересекающихся подмножеств. Для второй группы каждому разбиению множества $\{1, \dots, n-1\}$ на k непересекающихся подмножеств ($S_{n-1,k}$ вариантов) соответствует k разбиений, получающихся поочередным добавлением элемента n к каждому подмножеству. Следовательно, в этой группе $kS_{n-1,k}$ вариантов разбиений. ■

Эти свойства позволяют легко вычислять числа Стирлинга для произвольных n и k . В таблице 1.43 даны числа Стирлинга второго рода.

Таблица 1.43.

$k \backslash n$	1	2	3	4	5	6
1	1	0	0	0	0	0
2	1	1	0	0	0	0
3	1	3	1	0	0	0
4	1	7	6	1	0	0
5	1	15	25	10	1	0
6	1	31	90	65	15	1

Существует другая рекуррентная формула для вычисления чисел Стирлинга второго рода.

Теорема 1.73. Для $k \geq 2$ справедливо соотношение

$$S_{n,k} = \sum_{i=k-1}^{n-1} C_{n-1}^i S_{i,k-1}. \quad (1.96)$$

Доказательство. Рассмотрим n -элементное множество $\{1, \dots, n\}$ и зафиксируем в нем последний элемент n . Из оставшихся элементов для каждого $j \in 0 : n-k$ выделим j (это можно сделать C_{n-1}^j способами) и добавим к ним зафиксированный последний элемент n . Рассматриваем выбранные $j+1$ элементы как одно подмножество. Для оставшегося $(n-j-1)$ -элементного множества существует $S_{n-j-1,k-1}$ разбиений. Таким образом,

$$S_{n,k} = \sum_{j=0}^{n-1} C_{n-1}^j S_{n-j-1,k-1}. \quad (1.97)$$

Сделав в (1.97) замену переменной $i = n-j-1$ и учитывая свойство биномиальных коэффициентов ($C_{n-1}^j = C_{n-1}^{n-1-j}$), установленное в теореме 1.63, получаем требуемое. ■

Определение 1.52. Числом Белла (обозначают B_n) называют число всех разбиений n -элементного множества на $1, \dots, n$ непересекающихся подмножеств, т. е. $B_n = \sum_{k=0}^n S_{n,k}$.

Положим $B_0 = 1$. Докажем следующее рекуррентное соотношение.

Теорема 1.74. Справедливо равенство

$$B_{n+1} = \sum_{i=0}^n C_n^i B_i.$$

Доказательство. Для доказательства немного изменим рассуждения, используемые при доказательстве теоремы 1.73.

Рассмотрим $(n+1)$ -элементное множество и зафиксируем в нем элемент $n+1$. Будем выбирать произвольные i элементов из множества $\{1, \dots, n\}$. Оставшиеся $n+1-i$ элементов рассматриваем как *одно* подмножество. Так как $0 \leq i \leq n$, то в последнем подмножестве может быть от 1 до $n+1$ элементов. Для каждой выборки из i элементов существует B_i разбиений, а сами элементы можно выбрать C_n^i способами. ■

Введем в рассмотрение понятие *факториальных многочленов*.

Определение 1.53. Факториальный многочлен степени k

$$[x]_k = x(x-1) \cdots (x-k+1),$$

т. е. $[x]_1 = x$, $[x]_2 = x(x-1)$, $[x]_3 = x(x-1)(x-2)$ и т. д.

Очевидно, что $\{1, [x]_1, [x]_2, \dots, [x]_n, \dots\}$ — базис в пространстве многочленов.

Теорема 1.75. Справедливо равенство

$$x^n = \sum_{k=0}^n S_{n,k} [x]_k,$$

т. е. числа Стирлинга второго рода — это матрица перехода от канонического базиса $\{1, x, x^2, \dots, x^n, \dots\}$ к базису факториальных многочленов $\{1, [x]_1, [x]_2, \dots, [x]_n, \dots\}$.

Упражнение 1.12. Докажите теорему 1.75.

Определим обратную связь этих базисов.

Определение 1.54. Числа Стирлинга первого рода — это матрица перехода от базиса факториальных многочленов $\{1, [x]_1, [x]_2, \dots, [x]_n, \dots\}$ к каноническому базису $\{1, x, x^2, \dots, x^n, \dots\}$, т. е. $[x]_n = \sum_{k=0}^n s_{n,k} x^k$.

Лемма 1.16. Для введенных чисел Стирлинга первого рода $s(n, k)$ справедливы следующие свойства:

- 1) $s_{n,0} = 0, n > 0$;
- 2) $s_{n,n} = 1, n \geq 0$;
- 3) $s_{n,k} = s_{n-1,k-1} - (n-1)s_{n-1,k}, 0 < k < n$.

Доказательство. Свойства 1 и 2 очевидны. Докажем свойство 3. Имеем $[x]_n = [x]_{n-1}(x-n+1)$, тогда согласно определению 1.54:

$$\begin{aligned} \sum_{k=0}^n s_{n,k}x^k &= (x-n+1) \sum_{k=0}^{n-1} s_{n-1,k}x^k = \\ &= \sum_{k=0}^{n-1} s_{n-1,k}x^{k+1} - (n-1) \sum_{k=0}^{n-1} s_{n-1,k}x^k = \\ &= \sum_{k=1}^{n-1} s_{n-1,k-1}x^k + s_{n-1,k-1}x^n - (n-1) \sum_{k=1}^{n-1} s_{n-1,k}x^k - s_{n-1,0}(n-1) = \\ &= \sum_{k=1}^{n-1} (s_{n-1,k-1} - (n-1)s_{n-1,k})x^k + s_{n,n}x^n - (n-1)s_{n-1,0}. \end{aligned}$$

Заметим, что $s_{n,n} = 1, s_{n-1,0} = 0$. Приравняв коэффициенты при одинаковых степенях x , получаем требуемое. ■

Для введенных факториальных многочленов имеет место теорема, аналогичная теореме 1.62, для обычных многочленов.

Теорема 1.76 (факториальная теорема Вандермонда). Справедливо равенство

$$[x+y]_n = \sum_{k=0}^n C_n^k [x]_k [y]_{n-k}.$$

Упражнение 1.13. Докажите теорему 1.76.

1.6.11. Числа Каталана

Рассмотрим несколько задач, связанных с известной комбинаторной структурой — числами Каталана, и на их примере покажем приемы, позволяющие одни комбинаторные задачи сводить к другим установлением подходящего взаимно однозначного соответствия.

Историческая справка

Эжен Шарль Каталан (1814–1894) — французский и бельгийский математик, иностранный член-корреспондент Российской академии наук. Эти числа были известны еще Эйлеру за 100 лет до Каталана.

Задача 1.21. Рассмотрим правильные скобочные структуры, состоящие из n открывающих и n закрывающих скобок. Термин «правильные» означает, что для каждой открывающей скобки найдется единственная соответствующая ей закрывающая, стоящая правее ее.

Например, $(() ())$ и $() (())$ — правильные скобочные структуры для $n = 3$, а $() (())$ — неправильная структура для того же значения n .

Сколько существует правильных $2n$ -скобочных структур?

Задача 1.22. Рассмотрим произведение $n + 1$ матриц общего вида: $A_0 A_1 \cdots A_n$, в котором не расставлены скобки, так как произведение матриц ассоциативно.

Сколькими способами можно вычислить это произведение, сводя его к попарному умножению матриц? Или, что то же самое, сколькими способами можно расставить $2n$ скобок в указанном произведении?

На первый взгляд кажется, что задачи 1.21 и 1.22 одинаковы, но если сравнить два разных выражения $((AB)C)$ и $(A(BC))$, просто опустив буквы, то получатся одинаковые скобочные структуры $(())$.

Обратно, для правильной скобочной структуры $() ()$ нет произведения трех матриц с такой расстановкой скобок.

Задача 1.23. Сколькими способами можно разрезать неправильный выпуклый $n + 2$ угольник на n треугольников отрезками, соединяющими различные вершины (рисунок 1.11)?

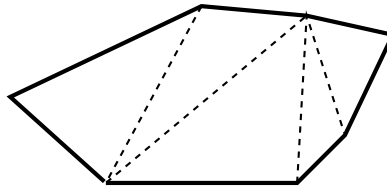


Рис. 1.11.

Задача 1.24. Сколько существует корневых бинарных деревьев с $n + 1$ листьями, у которых каждая вершина, не являющаяся листом, имеет ровно двух потомков (степень каждой вершины либо 1, либо 3)?

Задача 1.25. Человек с завязанными глазами стоит в одном шаге от края обрыва. Он может сделать либо шаг по направлению к обрыву (не

зная этого), либо шаг в обратную сторону. Каково число вариантов, сделав $2(n-1)$ шагов, не упасть с обрыва?

Задача 1.26. Сколько существует различных корневых деревьев на плоскости, имеющих n вершин?

Задача 1.27. Сколько существует последовательностей из $2n-1$ неотрицательных целых чисел, начинающихся с 1 и заканчивающихся на 1, у которых соседние числа отличаются на 1? Например, для $n=3$ к таким относятся последовательности 1212321 и 1232121.

Покажем, что все эти задачи имеют одинаковую комбинаторную структуру.

✓ *Сведение задачи 1.22 к задаче 1.21.*

Оставим открытые скобки на месте, буквы удалим, а каждый знак операции умножения заменим закрывающей скобкой. Получим взаимно однозначное соответствие. Например, выражениям $((AB)C)$ и $(A(BC))$ будут соответствовать скобочные структуры $(())$ и $() ()$. То, что это соответствие взаимно однозначно, будет следовать из теоремы о префиксной форме арифметического выражения (см. главу 3.1.1).

Сейчас заметим только, что префиксные записи, однозначно определяющие арифметические выражения, для наших примеров выглядят так: $\times \times ABC$ и $\times A \times BC$. Если отбросить последнюю букву, заменить знаки умножения на открывающие скобки, а буквы — на закрывающие, то получатся упомянутые скобочные структуры.

Упражнение 1.14.

1. Придумайте другое взаимно однозначное соответствие, сводящее задачи 1.21 и 1.22 друг к другу.

2. Как по скобочной структуре восстановить произведение матриц?

✓ *Сведение задачи 1.23 к задаче 1.22.*

Выделим одну из сторон многоугольника, а по периметру, начиная с соседней стороны, сопоставим имена переменных (матриц) остальным сторонам многоугольника.

Будем рассматривать различные разбиения многоугольника на треугольники, сопоставляя каждому вновь проведенному отрезку произведение выражений, сопоставленных ранее другим его сторонам, начиная со сторон многоугольника и учитывая естественный порядок множителей (рисунки 1.12a).

Полученное сопоставление является искомым взаимно однозначным соответствием.

Упражнение 1.15. По заданному порядку выполнения в умножении матриц постройте разбиение многоугольника на треугольники. Например, для многоугольника на рисунке 1.12а постройте разбиение по выражению $((AB)(CD))$.

✓ *Связь задач 1.24 и 1.23.*

На рисунке 1.12b показана связь между разбиением многоугольника на треугольники и корневыми бинарными деревьями.

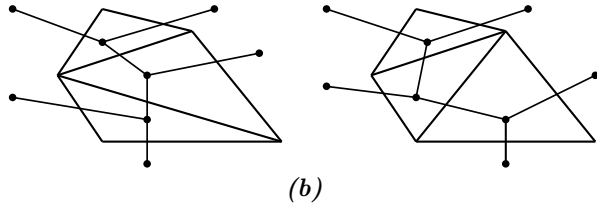
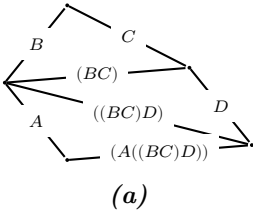


Рис. 1.12.

✓ *Связь задач 1.24 и 1.22.*

Обратим внимание, что для каждого арифметического выражения можно построить соответствующее синтаксическое дерево, которое и будет требуемым бинарным деревом (рисунок 1.13).

✓ *Сведение задачи 1.25 к задаче 1.21.*

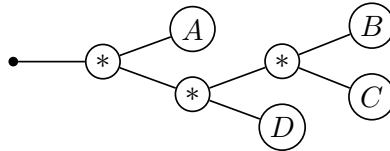


Рис. 1.13.

Закодируем каждый шаг человека от обрыва открывающей скобкой, к обрыву — закрывающей. Тогда для того, чтобы человек не дошел до обрыва, любой начальный отрезок скобочной структуры должен содержать больше открывающих скобок, чем закрывающих. Возвращение человека на исходную позицию означает совпадение числа открывающих и закрывающих скобок.

Эти два условия являются необходимыми и достаточными для правильности скобочной структуры. Взаимно однозначное соответствие доказано.

✓ *Сведение задачи 1.26 к задаче 1.21.*

Рассмотрим искомое взаимно однозначное соответствие на примере обхода дерева на рисунке 1.14. Будем пометать движение вверх открывающей скобкой, вниз — закрывающей. Например, обходу

$$A \rightarrow B \rightarrow D \rightarrow G \rightarrow D \rightarrow F \rightarrow H \rightarrow F \rightarrow D \rightarrow E \rightarrow D \rightarrow B \rightarrow C \rightarrow B \rightarrow A$$

соответствует правильная скобочная структура $((()((()())())))$.

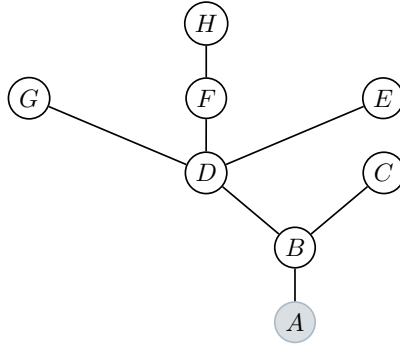


Рис. 1.14.

Упражнение 1.16. Сведите задачу 1.27 к одной из предыдущих задач.

В заключение решим одну из задач, что автоматически даст решение для остальных. Наибольший интерес представляет формулировка задачи 1.25, так как в ее решении идея взаимно однозначного соответствия используется еще раз.

Нарисуем график положений человека в дискретные моменты времени, соответствующие состояниям после очередного шага. Для наглядности точки соединим отрезками. Например, на рисунке 1.15 изображен путь, который может быть также описан скобочной структурой, соответствующей рисунку 1.14.

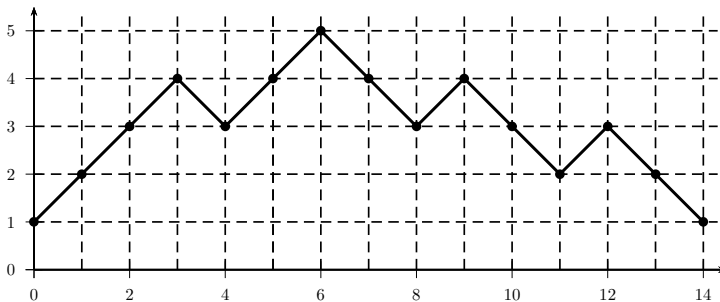


Рис. 1.15.

Любой путь из $2(n-1)$ шагов, не приводящий к падению человека с обрыва, изображается графиком, соединяющим точки $(0; 1)$ и $(2(n-1); 1)$ и не пересекающим ось абсцисс.

Чтобы найти число таких графиков, найдем число графиков, соединяющих эти точки, но не пересекающих ось абсцисс.

Каждому графику, соединяющему точки $(0; 1)$ и $(2(n-1); 1)$ и пересекающему ось абсцисс, можно взаимно однозначно сопоставить график, соединяющий точки $(0; -1)$ и $(2(n-1); 1)$. Для этого надо часть графика до первого пересечения отразить относительно оси абсцисс (рисунок 1.16).

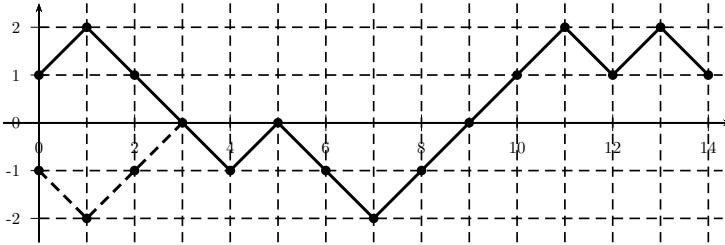


Рис. 1.16.

График состоит из $2(n-1)$ звеньев, из которых возрастающих звеньев на два больше, чем убывающих (так как разница ординат начала и конца графика равна 2): n — возрастающих и $n-2$ — убывающих. Значит, общее число графиков будет равно числу способов выбрать из $2(n-1)$ звеньев n возрастающих: C_{2n-2}^n .

Аналогично общее число графиков, соединяющих две точки $(0; 1)$ и $(2(n-1); 1)$, будет равно: C_{2n-2}^{n-1} . Таким образом, искомое число путей равно

$$\begin{aligned} C_{2n-2}^{n-1} - C_{2n-2}^n &= \frac{(2n-2)!}{(n-1)!(n-1)!} - \frac{(2n-2)!}{n!(n-2)!} = \\ &= \frac{(2n-2)!}{(n-1)!(n-2)!} \left(\frac{1}{n-1} - \frac{1}{n} \right) = \\ &= \frac{(2n-2)!}{(n-1)!(n-2)!} \frac{1}{n(n-1)} = \frac{1}{n} \frac{(2n-2)!}{(n-1)!(n-1)!} = \frac{1}{n} C_{2n-2}^{n-1}. \end{aligned}$$

Определение 1.55. Числа $\frac{1}{n} C_{2n-2}^{n-1}$ называют *числами Каталана*.

1.6.12. Разбиения чисел

Рассмотрим обобщение задачи 1.7. Пусть n — произвольное натуральное число.

Определение 1.56. *Разбиением* числа n называют набор чисел

$$\{a_1, \dots, a_k\}, \quad a_i \in \mathbb{Z}, \quad a_1 \geq \dots \geq a_k > 0 \quad \text{и} \quad n = a_1 + a_2 + \dots + a_k.$$

Будем обозначать разбиение числа n как $n = (a_1, \dots, a_k)$.

Пример 1.101. Пусть $n = 7$. Запишем все разбиения в обратном лексикографическом порядке (см. определение 1.49):

(7), (6, 1), (5, 2), (5, 1, 1), (4, 3), (4, 2, 1), (4, 1, 1, 1), (3, 3, 1), (3, 2, 1, 1),
(3, 1, 1, 1, 1), (2, 2, 2, 1), (2, 2, 1, 1, 1), (2, 1, 1, 1, 1, 1), (1, 1, 1, 1, 1, 1, 1).

Определение 1.57. Для каждого разбиения (a_1, \dots, a_k) можно построить *диаграмму Ферре* (*граф Ферре*). Она представляет собой таблицу из k строк, i -я строка которой содержит последовательность из a_i точек. Если заменить точки квадратами, то получим *диаграмму Юнга*.

Любому разбиению соответствует *сопряженное разбиение*, получаемое транспозицией диаграммы Ферре.

Наибольшую квадратную поддиаграмму диаграммы Ферре называют *квадратом Дюрфи* соответствующего разбиения. Если квадрат Дюрфи совпадает со всей диаграммой, такое разбиение называют *самосопряженным*.

Очевидно, что транспозиция диаграммы Ферре определяет взаимно однозначное соответствие между разбиением числа n на k слагаемых и его же разбиением с наибольшим слагаемым, равным k . Справедливо утверждение.

Лемма 1.17. Пусть $n = (a_1, \dots, a_k)$. Тогда число элементов, равных m , в сопряженном разбиении равно $a_m - a_{m+1}$ (считаем, что $a_{k+1} = 0$).

Лемма 1.17 дает удобный способ вычисления сопряженного разбиения, не рисуя диаграмму Ферре.

Пример 1.102. Для разбиения $12 = (4, 3, 3, 2)$ диаграммы Ферре и Юнга приведены на рисунках 1.17а и 1.17б.

Для соответствующего сопряженного разбиения $12 = (4, 4, 3, 1)$ диаграмма Ферре показана на рисунке 1.17с.

Квадраты Дюрфи выделены на рисунке 1.17а и 1.17с утолщенными линиями, на диаграмме Юнга квадрат Дюрфи темно-серого цвета.

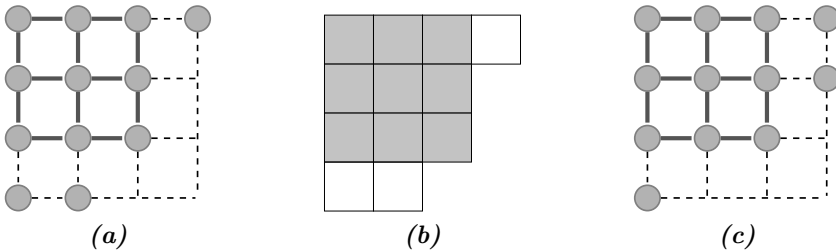


Рис. 1.17.

Теорема 1.77. Число разбиений n на попарно различные слагаемые равно числу разбиений n на нечетные слагаемые.

Доказательство. Рассмотрим разбиение числа n на нечетные слагаемые b_1, \dots, b_p , где слагаемое b_i появляется в разбиении r_i раз:

$$n = \underbrace{b_1 + b_1 + \dots + b_1}_{r_1} + \dots + \underbrace{b_p + b_p + \dots + b_p}_{r_p}. \quad (1.98)$$

Запишем двоичное представление r_i :

$$r_i = 2^{q_1} + \dots + 2^{q_s}, \quad q_1 > \dots > q_s \geq 0.$$

Заменим r_i слагаемых b_i в (1.98) на попарно различные слагаемые

$$b_i 2^{q_1}, \dots, b_i 2^{q_s}.$$

Очевидно, что при этом сохраняется сумма разбиения. Повторив эту операцию для всех $i = 1, \dots, p$ и упорядочив слагаемые, получим требуемое представление, так как $b_i 2^{q_i} \neq b_j 2^{q_m}$ в силу нечетности b_i, b_j . ■

Пример 1.103.

$$26 = 7+5+5+3+3+1+1+1 = 7 \cdot 2^0 + 5 \cdot 2^1 + 3 \cdot 2^1 + 1 \cdot (2^1 + 2^0) = 10+7+6+2+1.$$

Упражнение 1.17. Пусть $E(n)$ — число разбиений числа n с четным числом четных слагаемых, $O(n)$ — соответственно число разбиений с нечетным числом четных слагаемых и $S(n)$ — число самосопряженных разбиений. Покажите, что

$$E(n) - O(n) = S(n).$$

Построим алгоритм для генерации разбиений числа n . Генерацию будем производить в обратном лексикографическом порядке, т. е. разбиение $n = c_1 + c_2 + \dots + c_l$ идет за разбиением $n = a_1 + a_2 + \dots + a_k$ тогда и только тогда, когда существует $p \leq \min\{l, k\} : c_i = a_i$ для $i < p ; c_p < a_p$.

Очевидно, что первое разбиение $n = n$, последнее — $n = \underbrace{1 + 1 + \dots + 1}_n$.

Пусть $n = a_1 + a_2 + \dots + a_k$ — некое промежуточное разбиение. Какое разбиение будет следующим? Будем искать разбиение, имеющее самое большое число начальных слагаемых, равных слагаемым предыдущего разбиения, обозначим их a_1, a_2, \dots, a_{t-1} . Очевидно, что $t = \max\{i | a_i > 1\}$, т. е. предыдущее разбиение имеет вид:

$$n = a_1 + a_2 + \dots + a_{t-1} + a_t + \underbrace{1 + 1 + \dots + 1}_{k-t}.$$

Тогда оставшиеся слагаемые текущего легко определяются. Введем обозначение $s = a_t + \underbrace{1 + 1 + \dots + 1}_{k-t}$, т. е. s — сумма «остатка» предыдущего разбиения. Представим эту сумму s в виде: $s = \underbrace{l + l + \dots + l}_{[s/l]} + \langle s \rangle_l$, где $l = a_t - 1$, т. е. набираем s максимально возможными элементами, меньшими a_t . Тогда текущее разбиение имеет вид

$$n = a_1 + a_2 + \dots + a_{t-1} + \underbrace{l + l + \dots + l}_{[s/l]} + \langle s \rangle_l.$$

Запишем этот процесс в виде алгоритма. Введем следующие обозначения:

a_i — элементы текущего разбиения,

r_i — кратность i -го элемента в разбиении,

j — число элементов в текущем разбиении без учета кратности, значение переменной s описано ранее.

Алгоритм 1.36. Генерация разбиений

Инициализация:

$a_1 \leftarrow n$

$r_1 \leftarrow 1$

$j \leftarrow 1$

// первое разбиение

while $a_1 > 1$ **do**

$s \leftarrow 0$

if $a_j = 1$ **then** *// учет единичных слагаемых предыдущего разбиения*

$s \leftarrow s + r_j$

$j \leftarrow j - 1$

end if

$s \leftarrow s + a_j$ *// добавляем в s последний неединичный элемент предыдущего разложения*

$r_j \leftarrow r_j - 1$ *// уменьшаем его кратность в разложении*

$l \leftarrow a_j - 1$ *// запоминаем новый элемент*

if $r_j > 0$ **then**

$j \leftarrow j + 1$ *// в текущем разложении есть еще элементы a_j , увеличиваем индекс для нового элемента*

end if

$a_j \leftarrow l$ *// строим «остаток» нового разбиения*

$r_j \leftarrow s \div l$

$d \leftarrow \langle s \rangle_l$

if $d > 0$ **then** *// добавляем остаток d последним элементом в разбиение*

$j \leftarrow j + 1$

$a_j \leftarrow d$

```

     $r_j \leftarrow 1$ 
  end if
end while

```

Пример 1.104. Применим алгоритм 1.36 для $n = 9$. Протокол работы алгоритма дан в таблице 1.44.

Таблица 1.44.

Шаг	Текущее разбиение	Шаг	Текущее разбиение
1	$9 = 9$	8	$9 = 5 + 4$
2	$9 = 8 + 1$	9	$9 = 5 + 3 + 1$
3	$9 = 7 + 2$	10	$9 = 5 + 2 + 2$
4	$9 = 7 + 1 + 1$	11	$5 + 2 + 1 + 1$
5	$9 = 6 + 3$	12	$9 = 5 + 1 + 1 + 1 + 1$
6	$9 = 6 + 2 + 1$	13	$9 = 4 + 4 + 2$
7	$9 = 6 + 1 + 1 + 1$

Последнее разбиение $9 = 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1$.

1.6.13. Введение в динамическое программирование. Задача о рюкзаке

Идея динамического программирования заключается в следующем: исходная задача представляется в виде набора «подзадач» разного размера, решаемых одна за другой в *правильном* порядке. Правильный порядок означает, что когда нужно решать некоторую подзадачу набора, для этого уже все готово, то есть те задачи, от которых зависит ее решение, уже решены.

Применим динамическое программирование для решения *задачи о рюкзаке*. Рассмотрим классическую постановку задачи.

Имеется рюкзак вместимости W и набор из n предметов, каждый из которых имеет два положительных параметра: вес и стоимость. Задача заключается в том, чтобы собрать рюкзак с максимальной стоимостью предметов внутри, соблюдая при этом ограничение рюкзака на суммарный вес.

Введем обозначения: $\{w_1, \dots, w_n\}$ — веса и $\{v_1, \dots, v_n\}$ — стоимости предметов.

Существует большое количество постановок задачи, в зависимости от условий, наложенных на рюкзак, предметы или их выбор. Подробное описание и разбор можно найти в [78].

Историческая справка

В настоящее время трудно установить, кто первым привел математическую формулировку задачи о ранце. Одно из первых упоминаний о ней можно найти в статье 1897 г.

Однако интенсивное изучение данной проблемы началось в 1970–1990-е гг. Такой интерес к данной задаче вызван достаточно простой ее формулировкой, большим числом ее разновидностей и свойств и в то же время сложностью решения. Различные ее модификации широко применяются на практике в прикладной математике, криптографии, экономике, логистике и т.п.

В 1972 г. данная задача вошла в список NP-полных задач (см. определение 3.116, в разделе 3.5.9).

Рассмотрим сначала одну из наиболее популярных постановок: каждый предмет имеется в неограниченном количестве¹. При использовании метода динамического программирования очень важно правильно выбрать подзадачи. В этом качестве будем рассматривать рюкзаки меньшей вместимости. Обозначим $C(w)$ — максимально возможную стоимость для рюкзака вместимости w .

Попробуем выразить $C(w)$ через ответы для меньших подзадач. Пусть в оптимальное заполнение рюкзака емкости w входит предмет i , тогда, если его убрать, мы получим оптимальное заполнение рюкзака вместимости $w - w_i$. Тогда $C(w)$ — это $C(w - w_i) + v_i$ для некоторого i .

Поскольку индекс предмета i неизвестен, нужно перебрать все возможные варианты. Имеем

$$C(w) = \max_i \{C(w - w_i) + v_i \mid w_i \leq w\}.$$

Запишем алгоритм решения.

Алгоритм 1.37. Задача о неограниченном рюкзаке

```

C(0) ← 0
for w ← 1, W do
    C(w) = max {C(w - w_i) + v_i | w_i ≤ w}
end for
return C(W)

```

Пример 1.105. Пусть имеется 4 предмета. Данные об их весе и стоимости даны в таблице 1.45. Решим задачу алгоритмом 1.37 для $W = 9$.

Протокол работы алгоритма запишем в таблице 1.46.

Таким образом, получаем оптимальное решение:

$$C(9) = v_1 + v_1 + v_1 + v_2 = 22.$$

¹ Неограниченный рюкзак (англ. Unbounded Knapsack Problem).

Таблица 1.45.

i	1	2	3	4
v_i	5	7	6	3
w_i	2	3	4	2

Таблица 1.46.

i	$C(i)$	Решение	i	$C(i)$	Решение
0	0		5	12	$C(5) = C(5 - w_2) + v_2$
1	0		6	15	$C(6) = C(6 - w_1) + v_1$
2	5	$C(2) = C(2 - w_1) + v_1$	7	17	$C(7) = C(7 - w_2) + v_2$
3	7	$C(3) = C(3 - w_2) + v_2$	8	20	$C(8) = C(8 - w_1) + v_1$
4	10	$C(4) = C(4 - w_1) + v_1$	9	22	$C(9) = C(9 - w_1) + v_1$

Рассмотрим другую постановку задачи: каждый предмет имеется в одном экземпляре¹. Предыдущий подход использовать не удастся, так как на каждом шаге нужно учитывать предметы, которые уже взяты ранее.

Обозначим $C(w, i)$ — максимально возможную стоимость для рюкзака вместимости w и при этом разрешено брать предметы $1, \dots, i$. Как и в предыдущем случае, нужно научиться выражать $C(w, i)$ через результаты для предыдущих подзадач.

Возможны два случая: в оптимальном заполнении предмет i либо участвует, либо нет. Тогда

$$C(w, i) = \max \{C(w, i - 1), C(w - w_i, i - 1) + v_i \mid w_i \leq w\},$$

где второй элемент учитывается только при $w_i \leq w$.

Запишем алгоритм решения.

Алгоритм 1.38. Задача о рюкзаке 0–1

```

for  $i \leftarrow 0, n$  do
     $C(0, i) \leftarrow 0$ 
end for
for  $w \leftarrow 0, W$  do
     $C(w, 0) \leftarrow 0$ 
end for
for  $i \leftarrow 1, n$  do
    for  $w \leftarrow 1, W$  do

```

¹ Рюкзак 0–1 (англ. 0–1 Knapsack Problem).

```

if  $w_i > w$  then
     $C(w, i) \leftarrow C(w, i - 1)$ 
else
     $C(w, i) \leftarrow \max \{C(w, i - 1), C(w - w_i, i - 1) + v_i\}$ 
end if
end for
end for
return  $C(W, n)$ 

```

Пример 1.106. Пусть имеется 5 предметов. Данные об их весе и стоимости даны в таблице 1.47. Решим задачу алгоритмом 1.38 для $W = 6$.

Таблица 1.47.

i	1	2	3	4	5
v_i	1	2	3	1	1
w_i	2	3	2	4	1

Протокол работы алгоритма запишем в таблице 1.48.

Таблица 1.48.


i	$C(1, i)$	$C(2, i)$	$C(3, i)$	$C(4, i)$	$C(5, i)$	$C(6, i)$
1	0	$v_1 = 1$	1	1	1	1
2	0	$v_1 = 1$	$v_2 = 2$	2	$v_1 + v_2 = 3$	3
3	0	$v_3 = 3$	3	$v_1 = v_3 = 4$	$v_2 + v_3 = 5$	5
4	0	$v_3 = 3$	3	$v_1 = v_3 = 4$	$v_2 + v_3 = 5$	5
5	0	$v_3 = 3$	$v_3 + v_5 = 4$	4	$v_2 + v_3 = 5$	$v_2 + v_3 + v_5 = 6$

Таким образом, получаем оптимальное решение:

$$C(6, 5) = v_2 + v_3 + v_5 = 6.$$

1.6.14. Принцип включения-исключения

Рассмотрим свойства $p(1), \dots, p(n)$, которыми обладают элементы множества S , состоящего из N элементов. Предположим, что для каждого элемента можно однозначно определить, обладает он свойством $p(i)$ или нет. Число элементов, обладающих свойствами $p(i_1), \dots, p(i_r)$, обозначим через $N_{i_1 \dots i_r}$.

 **Замечание 1.60**

Здесь не рассматриваются другие свойства, т. е. допускается, что некоторые из элементов, быть может, обладают и другими свойствами.

Число элементов, которые не обладают ни одним из рассматриваемых свойств, обозначим через $N(0)$. Число элементов, которые обладают *ровно* r ($1 \leq r \leq n$) свойствами из n возможных, будем обозначать $N(r)$. Через $N[r]$ обозначим число элементов, обладающих *не менее* r свойствами. Тогда

$$N[r] = \sum_{i_1 < \dots < i_r} N_{i_1 \dots i_r} = \sum_{i=r}^n N(i). \quad (1.99)$$

В самом простом случае ($n = 2$) очевидно, что

$$N(0) = N - (N_1 + N_2) + N_{12} = N[0] - N[1] + N[2].$$

Действительно, каждый элемент, обладающий свойствами 1 и 2 одновременно, учитывается и в N_1 , и в N_2 , а поэтому при вычислении разности $N - (N_1 + N_2)$ он вычитается из N дважды. Следовательно, если к этой разности добавить N_{12} , то получится $N(0)$ — число элементов, не обладающих ни одним из двух рассматриваемых свойств. Обобщением этого правила является следующий принцип включения-исключения.

Теорема 1.78 (принцип включения-исключения).

$$N(0) = N[0] - N[1] + N[2] - \dots + (-1)^s N[s] + \dots + (-1)^n N[n]. \quad (1.100)$$

Доказательство. Элементы, не обладающие ни одним из свойств, входят в правую часть равенства (1.100) ровно по одному разу, а именно в первое слагаемое — N . Рассмотрим элементы, которые обладают ровно r свойствами $p(j_1), \dots, p(j_r)$, и слагаемые из (1.100) вида

$$(-1)^s N[s], \quad s \leq r.$$

Так как совокупность индексов i_1, \dots, i_s из множества j_1, \dots, j_r можно выбрать C_r^s способами, вклад каждого из рассматриваемых элементов в правую часть (1.100) будет

$$1 - C_r^1 + C_r^2 - \dots + (-1)^s C_r^s + \dots + (-1)^r C_r^r = 0.$$

Воспользовались свойством биномиальных коэффициентов (следствие 1.22 к теореме 1.62). ■

Историческая справка

Впервые формулу (1.100) опубликовал португальский математик Даниэль да Сильва в 1854 г. Но еще в 1713 г. Николай Бернулли использовал этот метод для решения задачи Монмора (пример 1.108). Другие названия: символический метод, принцип перекрестной классификации, метод решета.

В качестве применения принципа включения-исключения докажем теорему Лежандра.

Теорема 1.79 (Лежандр). Пусть a_1, \dots, a_k набор попарно взаимно простых натуральных чисел. Тогда количество натуральных чисел, не превышающих некоторого n и взаимно простых с любым из a_i равно

$$n - \sum_{1 \leq i \leq k} \left\lfloor \frac{n}{a_i} \right\rfloor + \sum_{1 \leq i < j \leq k} \left\lfloor \frac{n}{a_i a_j} \right\rfloor - \sum_{1 \leq i < j < s \leq k} \left\lfloor \frac{n}{a_i a_j a_s} \right\rfloor + \dots + (-1)^k \left\lfloor \frac{n}{a_1 \dots a_k} \right\rfloor \quad (1.101)$$

Доказательство. Пусть S — множество натуральных чисел $\{1, \dots, n\}$. Обозначим через $p(i)$ свойство элемента множества S делиться на a_i . Тогда $N_{i_1 \dots i_r}$ — это количество чисел, не превышающих n и делящихся нацело на каждое из чисел a_{i_1}, \dots, a_{i_r} . Так как все a_i попарно взаимно просты, то

$$N_{i_1 \dots i_r} = \left\lfloor \frac{n}{a_{i_1} \dots a_{i_r}} \right\rfloor.$$

Подставляя последнее выражение в (1.100) и учитывая (1.99), получаем требуемое. ■

Следствие 1.27. Из теоремы Лежандра легко получить уже известную нам формулу (1.54) для функции Эйлера.

Доказательство. В теореме Лежандра заменим a_i на p_i — простые делители числа n . Тогда

$$N_{i_1 \dots i_r} = \frac{n}{p_{i_1} \dots p_{i_r}}.$$

Подставим последнее выражение в (1.101):

$$\begin{aligned} \varphi(n) &= n - \sum_{1 \leq i \leq k} \frac{n}{p_i} + \sum_{1 \leq i < j \leq k} \frac{n}{p_i p_j} - \sum_{1 \leq i < j < s \leq k} \frac{n}{p_i p_j p_s} + \dots + (-1)^k \frac{n}{p_1 \dots p_k} = \\ &= n \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \dots \left(1 - \frac{1}{p_k}\right), \end{aligned}$$

получаем (1.54). ■

Теорему 1.78 можно обобщить для числа элементов, обладающих ровно r свойствами.

Теорема 1.80.

$$\begin{aligned} N(r) &= C_r^r N[r] + \dots + (-1)^i C_{r+i}^r N[r+i] + \dots + (-1)^{n-r} C_n^r N[n] = \\ &= \sum_{i=0}^{n-r} (-1)^i C_{r+i}^r N[r+i]. \end{aligned} \quad (1.102)$$

Доказательство. Покажем, что каждый элемент, обладающий ровно r свойствами, учитывается в правой части (1.102) ровно один раз. Действительно, элементы, обладающие числом свойств меньше r , очевидно не учитываются. Элемент, обладающий фиксированными $s = r + j$ свойствами ($j \geq 0$), учитывается во внутренней сумме (1.102) ровно C_{r+j}^{r+i} раз.

По свойствам биномиальных коэффициентов (теорема 1.65 и следствие 1.22 к теореме 1.62) имеем

$$\sum_{i=0}^{n-r} (-1)^i C_{r+i}^r C_{r+j}^{r+i} = \sum_{i=0}^{n-r} (-1)^i C_{r+j}^r C_j^i = C_{r+j}^r \sum_{i=0}^j (-1)^i C_j^i = \begin{cases} 1 & \text{при } j = 0, \\ 0 & \text{при } j > 0. \end{cases}$$

Таким образом, в (1.102) элементы, обладающие ровно r свойствами, учитываются ровно по одному разу, а прочие элементы не учитываются. ■

Пример 1.107. Из 50 студентов 30 изучают английский язык, 25 — немецкий, 25 — французский, 15 — английский и немецкий, 20 — английский и французский, 10 — немецкий и французский, 5 — английский, французский и немецкий.

1. Сколько студентов не изучает ни одного из перечисленных языков?
2. Сколько студентов изучает ровно два языка?
3. Сколько студентов изучает не менее двух языков?

Обозначим признаки изучения английского, немецкого и французского языков соответственно $p(1), p(2), p(3)$. Согласно принципу включения-исключения (теорема 1.78) число студентов, не изучающих ни одного языка, равно

$$N(0) = N[0] - N[1] + N[2] - N[3] = 50 - (30 + 25 + 25) + (15 + 20 + 10) - 5 = 10.$$

Для ответа на второй вопрос воспользуемся теоремой 1.80. Тогда


$$N(2) = C_2^2 N[2] - C_3^2 N[3] = (15 + 20 + 10) - 3 \cdot 5 = 30.$$

Ответ на третий вопрос следует из формулы (1.99):

$$N[2] = N(2) + N(3) = 30 + 5 = 35.$$

Определение 1.58. Перестановку из n целых чисел $1, 2, \dots, n$, в которой ни один элемент не занимает своего естественного места, называют *беспорядком*. Например, при $n = 3$ перестановка $(3, 1, 2)$ является беспорядком, а $(3, 2, 1)$ не является, так как элемент 2 занимает в ней свое естественное место.

Пример 1.108 (задача о числе беспорядков). Найдем с помощью принципа включения-исключения общее число беспорядков. Обозначим его d_n .

 **Замечание 1.61**

Задачу о числе беспорядков называют также задачей Монмора или задачей о встрече. Пьер Монмор — французский математик (1678–1719). Настоящая фамилия — Ремон, Монмор — название его поместья.

Обозначим через $p(i)$ свойство перестановки, состоящее в том, что i -я позиция в ней занимает число i . Так как $P_n = n!$, то $N = n!$. Очевидно, что $N_{i_1 \dots i_r}$ — число перестановок, в которых позиции i_1, \dots, i_r должны быть заняты соответствующими числами, а остальные $n - r$ позиций могут быть заполнены оставшимися $n - r$ числами произвольным образом. Тогда $N_{i_1 \dots i_r} = (n - r)!$.

Имеется C_n^r способов выбрать r позиций i_1, \dots, i_r из n возможных, так что

$$N[r] = C_n^r (n - r)! = \frac{n!}{r!}.$$

Согласно принципу включения-исключения (см. теорему 1.78)

$$d_n = n! \left[1 - 1 + \frac{1}{2!} - \frac{1}{3!} + \dots + (-1)^r \frac{1}{r!} + \dots + (-1)^n \frac{1}{n!} \right] = n! \sum_{r=0}^n \frac{(-1)^r}{r!}.$$

Заметим, что $d_n/n!$ равно сумме первых $(n+1)$ членов ряда Тейлора при разложении $e^{-1} = 0.36806 \dots$ При этом скорость сходимости к e^{-1} достаточно велика: $d_6/6! = 0.36788 \dots$

Величину d_n часто называют *субфакториалом*. Субфакториал обладает свойствами, похожими на свойства обычного факториала. Действительно, справедливы рекуррентные соотношения:

$$\begin{aligned} n! &= (n - 1) [(n - 1)! + (n - 2)!], \\ d_n &= (n - 1) [d_{n-1} + d_{n-2}]. \end{aligned} \tag{1.103}$$

Рекуррентную формулу (1.103) для d_n легко преобразовать к виду

$$d_n = nd_{n-1} + (-1)^n. \quad (1.104)$$

Используя формулу (1.104), вычислим значения первых субфакториалов (таблица 1.49).

Таблица 1.49.

n	1	2	3	4	5	6	7	8	9	10	11
d_n	0	1	2	9	44	255	1784	14273	128456	1284561	14130170

Упражнение 1.18. Докажите соотношения (1.103).

1.6.15. Обращение Мебиуса

Найденная в предыдущем подразделе формула включений-исключений (1.78) относится к так называемым формулам обращения. Еще одна знаменитая формула обращения связана с суммированием по множеству делителей числа n . Эта формула использует введенную ранее функцию Мебиуса $\mu(n)$ (определение 1.25) и имеет обобщение на случай произвольных частично упорядоченных множеств, о которых речь пойдет во второй части.

Историческая справка

Август Фердинанд Мебиус (August Ferdinand Möbius) (1790–1868) — немецкий математик, работал ассистентом Гаусса. Им был сделан значительный вклад в геометрию, в частности топологию (лента Мебиуса названа его именем). В 1840 г. им впервые была сформулирована «проблема четырех красок».

Рассмотрим функцию, заданную на множестве делителей натурального числа n следующим образом:

$$g(n) = \sum_{d|n} f(d).$$

Пример 1.109. Рассмотрим натуральное число n , представленное в каноническом виде (1.6):

$$n = p_1^{k_1} p_2^{k_2} \dots p_k^{k_l}, \quad k_i \geq 1, \quad i \in 1 : l.$$

1. Пусть $N(n)$ — число делителей числа n . Тогда $N(n) = \sum_{d|n} 1$, т. е. $f(d) = 1$. Значения этой функции для небольших значений аргумента:

$$N(2) = 2, \quad N(6) = 4, \quad N(12) = 6, \quad N(36) = 9, \quad N(60) = 12.$$

Заметим, что в общем случае число делителей находится по формуле

$$N(n) = (k_1 + 1)(k_2 + 1) \cdots (k_l + 1).$$

2. Пусть $S(n)$ — сумма всех делителей числа n . Тогда $S(n) = \sum_{d|n} d$, т. е. $f(d) = d$. Значения этой функции для небольших значений аргумента:

$$\begin{aligned} S(2) &= 3, S(4) = 7, S(6) = 12, S(10) = 18, S(12) = 28, \\ S(20) &= 42, S(30) = 72, S(36) = 91, S(60) = 168. \end{aligned}$$

Можно ли решить обратную задачу и найти функцию $f(n)$ через значения функции $g(d)$ на множестве делителей числа n ?

Попробуем найти эту формулу, применив принцип включений-исключений. Не умаляя общности в качестве функции $g(n)$, рассмотрим количество всех делителей числа $N(n)$. Тогда задача будет состоять в том, чтобы через количество делителей всех делителей данного числа выразить само число.

Обозначим $N'(i_1, \dots, i_k)$ — число всех делителей числа $n/(p_{i_1} \cdots p_{i_k})$, где p_{i_1}, \dots, p_{i_k} — различные простые делители числа n .

По формуле включений-исключений (1.100):

$$N'(0) = N(n) - \sum_{p_i|n} N'(i) + \sum_{i_1 < i_2} N'(i_1, i_2) - \cdots + (-1)^k \sum_{i_1 < \cdots < i_k} N'(i_1, \dots, i_k), \quad (1.105)$$

где $N'(0)$ — число делителей числа n , которые не являются делителями других делителей числа n , таким числом является только само n : $N'(0) = 1$.

Учитывая определение 1.25 функции Мебиуса $\mu(n)$, данное выражение можно переписать так:

$$1 = \sum_{d|n} \mu(d) N\left(\frac{n}{d}\right). \quad (1.106)$$

Действительно, если $d = 1$, то $\mu(d) = 1$ и получается первый член суммы; если d имеет кратные простые делители, то $\mu(d) = 0$ и соответствующие члены в сумме будут отсутствовать. Оставшиеся члены в точности совпадают с членами суммы (1.105).

Пример 1.110. Подставим в формулу (1.106) данные из примера 1.109:

$$\begin{aligned} 1 &= N(12) - (N(6) + N(4)) + N(2), \\ 1 &= N(60) - (N(30) + N(20) + N(12)) + (N(10) + N(6) + N(4)) - N(2). \end{aligned}$$

Заметим, что в правой части встречаются суммы не всех делителей, например в первую сумму не входит $N(3)$, во вторую — $N(15)$.

Упражнение 1.19. Проверьте полученную формулу на функции $S(n)$ из примера 1.109.


В общем случае получаем такой результат.

Теорема 1.81 (формула обращения Мебиуса).

$$\text{Если } g(n) = \sum_{d|n} f(d), \text{ то } f(n) = \sum_{d|n} \mu(d) g\left(\frac{n}{d}\right).$$

Рассмотрим применение этой формулы для решения следующей комбинаторной задачи.

Задача 1.28. Сколько существует циклических последовательностей длины n , составленных из алфавита, в котором r различных символов?

 **Замечание 1.62**

Описанную в задаче 1.28 последовательность можно интерпретировать как бесконечную, полученную повторением начального куска длины n в обе стороны, тогда конечные последовательности

$$[a_1, a_2, \dots, a_n], [a_2, a_3, \dots, a_n, a_1], \dots, [a_n, a_1, \dots, a_{n-1}]$$

следует считать одинаковыми, так как они соответствуют одной бесконечной в обе стороны последовательности. Трудность подсчета состоит в том, что в самой конечной последовательности могут быть циклы различной длины.

Пример 1.111. Рассмотрим циклические последовательности длиной 6 над алфавитом из двух символов $\{A, B\}$. Их четырнадцать:

$$\begin{aligned} & [AAAAAA] \quad [AAAAAB] \quad [AAAABB] \quad [AAABAB] \quad [AABAAB] \\ & [AAABBB] \quad [AABABB] \quad [ABAABB] \quad [ABABAB] \quad [BBABBA] \\ & [BBBABA] \quad [BBBBA] \quad [BBBBBA] \quad [BBBBBB]. \end{aligned}$$

Эти последовательности можно разбить на группы по длине периода:

Период длиной 1: $[AAAAAA]$, $[BBBBBB]$.

Период длиной 2: $[ABABAB]$.

Период длиной 3: $[AABAAB]$, $[BBABBA]$.

Период длиной 6: $[AAAAAB]$, $[AAAABB]$, $[AAABAB]$, $[AAABBB]$,


$[AABABB]$, $[ABAABB]$, $[BBBABA]$, $[BBBBA]$,

$[BBBBBA]$.

Из конечной последовательности с периодом длиной k циклическим сдвигом периода можно получить $k - 1$ других конечных последовательностей, представляющих ту же самую циклическую последовательность.

Пример 1.112. Из последовательности $[АВААВ]$ циклическим сдвигом периода можно получить еще две последовательности $[ВААВАА]$ и $[АВААВА]$.

Обозначим число циклических последовательностей длиной n над алфавитом из r символов с периодом k как $M(k)$.

 **Замечание 1.63**

Более аккуратно это число надо обозначать как $M(n, r, k)$, так как оно зависит от количества символов в алфавите и длины конечной последовательности.

Добавив к каждой конечной последовательности с периодом длины k остальные $k - 1$ конечных последовательностей, полученных циклическим сдвигом периода, получим $kM(k)$ последовательностей. Если их суммировать по всем k , являющихся делителями n , то получим множество всех слов из n букв над алфавитом из r символов. Число таких слов равно r^n .

Тогда

$$r^n = \sum_{k|n} kM(k). \quad (1.107)$$

Пример 1.113. Подставив в формулу (1.107) данные из примера 1.110, получим

$$2^6 = 1 \cdot 2 + 2 \cdot 1 + 3 \cdot 2 + 6 \cdot 9,$$

что истинно.

Применив к формуле (1.107) формулу обращения Мебиуса (см. теорему 1.81), имеем:

$$nM(n) = \sum_{k|n} \mu(k)r^{\frac{n}{k}}, \quad \text{откуда} \quad M(n) = \frac{1}{n} \sum_{k|n} \mu(k)r^{\frac{n}{k}}. \quad (1.108)$$

Пример 1.114. Подставив в формулу (1.108) $n = 6, r = 2$, получим

$$\begin{aligned} M(6) &= \frac{1}{6} \sum_{k|6} \mu(k)2^{\frac{6}{k}} = \frac{1}{6} (1 \cdot 2^6 - 1 \cdot 2^3 - 1 \cdot 2^2 + 1 \cdot 2^1) = \\ &= \frac{1}{6} (64 - 8 - 4 + 2) = \frac{54}{6} = 9 \end{aligned}$$

— число циклических последовательностей с периодом длиной 6.

Таким образом, общее число циклических последовательностей длиной n над алфавитом из r букв вычисляется по формуле

$$M = \sum_{d|n} M(d) = \sum_{d|n} \frac{1}{d} \sum_{k|d} \mu(k) r^{\frac{d}{k}}.$$

Задача 1.29. На основе формулы Мебиуса выведите формулу для подсчета числа циклических последовательностей длиной n , которые можно получить из набора символов, в котором ровно b_i элементов i -го рода и $b_1 + b_2 + \dots + b_r = n$.

Пример 1.115. Рассмотрим циклические последовательности длиной 6, построенные из четырех букв A ($b_1 = 4$) и двух букв B ($b_2 = 2$). Таких последовательностей три: $[AAAABB]$, $[AAABAB]$, $[AABAAB]$, из них первые две имеют период длиной 6, а последняя — период длиной 3.

1.6.16. Понятие о группе: теорема Бернсайда

1.6.16.1. Самосовмещения правильного тетраэдра

Рассмотрим правильный тетраэдр — треугольную пирамиду, грани которой являются правильными треугольниками. Поставим тетраэдр на лист бумаги и обведем основание. Теперь поднимем тетраэдр, повернем и поставим обратно какой-либо гранью на начерченный треугольник. При этом тетраэдр займет то же место в пространстве, но его положение изменится (говорят, что мы сделали самосовмещение тетраэдра движением первого рода, т. е. движением, состоящим из поворотов и параллельных переносов).

Сколько существует различных движений такого сорта?

Начнем перечисление, упорядочив перебор вариантов. Переберем сначала преобразования, сохраняющие положение какой-нибудь вершины (повороты тетраэдра вокруг его высот: рисунок 1.18*b*, d , f и h).

Далее будем действовать более формально. Обозначим начальные положения вершин тетраэдра цифрами 1, 2, 3, 4 (рисунок 1.18*a*), тогда преобразование тетраэдра запишем в виде так называемой подстановки.

Например,

$$a_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix}, \quad a_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \end{pmatrix}$$

описывают повороты тетраэдра относительно высоты, проходящей через вершину 4. Читать такую запись, например, для a_1 , можно так: вершина 1 переходит на место вершины 3, вершина 2 — на место вершины 1 и т. д.

Подстановочная запись очень удобна, если необходимо узнать результат нескольких последовательных преобразований.

Например, найдем, как будет выглядеть положение тетраэдра после последовательного выполнения поворотов:

$$a_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix}, \quad a_4 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 3 & 2 \end{pmatrix}.$$

Проследим за перемещением вершин тетраэдра:

$$\begin{array}{ll} 1 \mapsto 3 \text{ (после } a_1) & 3 \mapsto 3 \text{ (после } a_4) \\ 2 \mapsto 1 \text{ (после } a_1) & 1 \mapsto 4 \text{ (после } a_4) \\ 3 \mapsto 2 \text{ (после } a_1) & 2 \mapsto 1 \text{ (после } a_4) \\ 4 \mapsto 4 \text{ (после } a_1) & 4 \mapsto 2 \text{ (после } a_4) \end{array}$$

Эти перемещения тоже описываются подстановкой, которую называют произведением подстановок a_1, a_4 и обозначают $a_1 \circ a_4$. Этой подстановке соответствует самосовмещение тетраэдра, обозначенное на рисунке 1.18*j* как a_9 :

$$a_9 = a_1 \circ a_4 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}.$$

Заметим, что это преобразование попарно меняет местами положения вершин 1, 3 и 2, 4 (рисунок 1.18*j*). Можно наглядно представить это преобразование как поворот на 180° относительно середин противоположных ребер. Обратим внимание, что если выполнить преобразования a_1, a_4 в другом порядке, то получится другой результат:

$$a_4 \circ a_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix} = a_{10}.$$

Для подстановок есть другой способ их записи, который называют записью в виде произведения независимых циклов.

$$a_9 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix} = (13)(24), \quad a_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix} = (132)(4).$$

Последняя запись означает, что вершины 1, 2, 3 циклически переходят друг в друга (см. рисунок 1.18*b*), а вершина 4 остается на месте (переходит в себя). Кроме a_9 есть еще два преобразования, не сохраняющие положения ни одной из вершин (см. рисунок 1.18*k* и *l*).

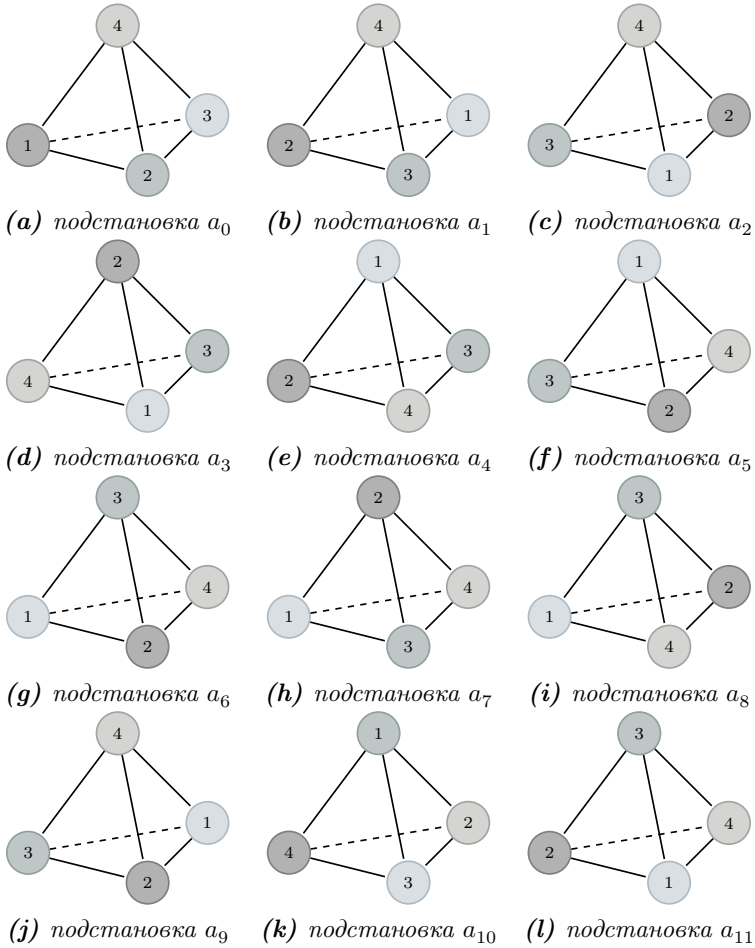


Рис. 1.18.

Обратим внимание на различные закономерности, которым удовлетворяют поворотные самосовмещения тетраэдра, например:

$$a_1 \circ a_3 = a_6, \quad a_1^2 = a_1 \circ a_1 = a_2, \quad a_9^2 = a_0,$$

а среди них — на произведения, которые в результате дают преобразование a_0 , т. е. возвращают тетраэдр в исходное состояние, например:

$$a_1 \circ a_2 = a_0, \quad a_3 \circ a_4 = a_0, \quad \dots \quad a_0^2 = a_0, \quad a_9^2 = a_0.$$

Геометрически это означает, что a_2 является обратным преобразованием к a_1 . Аналогично, a_4 является обратным к a_3 , a_0 — к a_0 и a_9 — к a_9 . Это обозначают следующим образом:

$$a_2 = a_1^{-1}, \quad a_4 = a_3^{-1}, \quad a_0 = a_0^{-1}, \quad a_9 = a_9^{-1}.$$

Сформулируем основные свойства описанных преобразований тетраэдра (подстановок):

1) существует нейтральный элемент

$$a_0 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix},$$

не меняющий положения тетраэдра;

2) у каждого элемента (некоторого преобразования тетраэдра) из полученного множества есть обратный (возвращающий тетраэдр в исходное состояние);

3) произведением двух подстановок множества является подстановка, соответствующая двум последовательным преобразованиям тетраэдра;

4) произведение подстановок (преобразований тетраэдра) обладает свойством ассоциативности, но не обладает, вообще говоря, свойством коммутативности.

Эти свойства определяют на множестве подстановок математическую структуру, которую называют *некоммутативной группой умножения подстановок* (см. определение 1.17).

1.6.16.2. Подгруппы и классы смежности

Определение 1.59. Если множество G конечно, то группу называют *конечной*, а число ее элементов — *порядком* группы. Обозначают порядок группы как $|G|$.

Пример 1.116. Порядок описанной группы поворотных самосовмещений тетраэдра равен 12.

Поставим теперь такую задачу: можно ли получить все положения тетраэдра, используя не все, а одно или несколько различных преобразований, если при этом их можно осуществлять любое количество раз в любых комбинациях, перемножая и обращая.

Пример 1.117. Возьмем преобразование

$$a_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix}.$$

Преобразование a_2 можно получить повторным выполнением преобразования a_1 :

$$a_2 = a_1 \circ a_1 = a_1^2,$$

или обращением преобразования a_1 :

$$a_2 = a_1^{-1}.$$

Далее, если умножить a_1 на себя трижды или, что то же самое, умножить a_1 на уже полученное a_2 , то результатом будет a_0 :

$$a_1^3 = a_1 \circ a_2 = a_0.$$

Как бы мы ни старались комбинировать эти подстановки, новых преобразований не получим. Таким образом, выразить все элементы группы G через a_1 нельзя.

В примере 1.117 мы построили группу G_1 меньшего размера, являющуюся частью группы всех поворотов: $G_1 = \{a_0, a_1, a_2\}$. Аналогично можно построить группы $G_2 = \{a_0, a_3, a_4\}$, $G_3 = \{a_0, a_5, a_6\}$ и т. д., которые называют *подгруппами* исходной группы G . Порядок каждой из подгрупп G_1, G_2, G_3 равен 3.

Определение 1.60. Если $G_1 \subset G$ и G_1, G являются группами, то G_1 называют *подгруппой* G .

Определение 1.61. Множества вида $aG_1 = \{a \circ g \mid g \in G_1\}$, где a — фиксированный элемент группы G , называют *классами смежности* G по подгруппе G_1 .

Очевидно, если элемент a принадлежит подгруппе G_1 , то $aG_1 = G_1$.

Пример 1.118. Рассмотрим классы смежности G по G_1 , не совпадающие с G_1 :

$$\begin{aligned} a_3G &= \{a_3 \circ a_0, a_3 \circ a_1, a_3 \circ a_2\} = \{a_3, a_7, a_9\}; \\ a_4G &= \{a_4 \circ a_0, a_4 \circ a_1, a_4 \circ a_2\} = \{a_4, a_{10}, a_5\}; \\ a_5G &= \{a_5 \circ a_0, a_5 \circ a_1, a_5 \circ a_2\} = \{a_5, a_4, a_{10}\}; \\ a_6G &= \{a_6 \circ a_0, a_6 \circ a_1, a_6 \circ a_2\} = \{a_6, a_{11}, a_8\}; \\ a_7G &= \{a_7 \circ a_0, a_7 \circ a_1, a_7 \circ a_2\} = \{a_7, a_9, a_3\}; \\ a_8G &= \{a_8 \circ a_0, a_8 \circ a_1, a_8 \circ a_2\} = \{a_8, a_6, a_{11}\}; \\ a_9G &= \{a_9 \circ a_0, a_9 \circ a_1, a_9 \circ a_2\} = \{a_9, a_3, a_7\}; \\ a_{10}G &= \{a_{10} \circ a_0, a_{10} \circ a_1, a_{10} \circ a_2\} = \{a_{10}, a_5, a_4\}; \\ a_{11}G &= \{a_{11} \circ a_0, a_{11} \circ a_1, a_{11} \circ a_2\} = \{a_{11}, a_8, a_6\}. \end{aligned}$$

В примере 1.118 можно проследить закономерность — классы смежности либо совпадают, либо не пересекаются. Число классов смежности равно частному от деления порядка группы на порядок подгруппы.

Теорема 1.82. Если G_1 подгруппа группы G и $p = |G|$, $p_1 = |G_1|$ — порядки групп G и G_1 , то:

- 1) p делится на p_1 ;
- 2) множество классов смежности $\{gG_1 \mid g \in G\}$ состоит из p/p_1 непересекающихся множеств, каждое из которых имеет p_1 элементов.

Доказательство.

1) Докажем, что классы смежности не пересекаются. От противного: пусть найдутся два совпадающие элемента из разных классов смежности: $c \in aG_1$ и $c \in bG_1$.

Это означает существование элементов g_1 и g_2 подгруппы G_1 таких, что $a \circ g_1 = b \circ g_2$, откуда следует

$$a = b \circ g_2 \circ g_1^{-1} = b \circ f, \text{ где } f = g_2 \circ g_1^{-1} \in G_1.$$

Таким образом, элемент a принадлежит классу bG_1 , но тогда и все элементы класса aG_1 попадут в класс bG_1 .

Действительно, любой элемент класса aG_1 записывается как

$$a \circ g = (b \circ f) \circ g = b \circ (f \circ g) = b \circ h \in bG_1, \text{ где } f, g, h \in G_1.$$

Следовательно, предположение о непустом пересечении привело к совпадению классов.

2) Докажем, что количество элементов одного класса смежности равно числу элементов подгруппы. Также от противного: пусть $a \circ g_1 = a \circ g_2$ для разных элементов подгруппы, тогда, домножив обе части на элемент, обратный a , получим:

$$a^{-1} \circ (a \circ g_1) = a^{-1} \circ (a \circ g_2) \text{ или } (a^{-1} \circ a) \circ g_1 = (a^{-1} \circ a) \circ g_2,$$

откуда $g_1 = g_2$, что и дает противоречие с условием.

Таким образом, мы разбили все p элементов исходной группы G на k непересекающихся классов, в каждом из которых число элементов равно числу элементов p_1 подгруппы G_1 . Значит, $p = kp_1$. ■

Вернемся к задаче о порождающих элементах группы.

Определение 1.62. Набор элементов группы называют *системой образующих*, если любой элемент группы можно выразить как некоторое произведение образующих или им обратных.

Если система образующих состоит из одного элемента, то такую группу называют циклической.

Пример 1.119. Определенная в предыдущих примерах подгруппа G_1 вращений относительно высоты, проходящей через вершину 4, задается одним образующим элементом a_1 :

$$a_2 = a_1^2, \quad a_0 = a_1^3.$$

Если в качестве образующих взять элементы a_1 и a_3 , то получим все элементы группы поворотов тетраэдра.

Задача 1.30. Выразите все элементы группы G через a_1 и a_3 , используя операции умножения и, если надо, обращения. Можно ли обойтись только операциями умножения?

Определение 1.63. *Порядком элемента* группы g называют наименьший показатель степени p , в которую надо возвести элемент (p раз перемножить элемент g), чтобы получить нейтральный элемент e :

$$\underbrace{g \circ g \circ \dots \circ g}_p = g^p = e.$$

Замечание 1.64

Обратите внимание, что когда образующий элемент один, т. е. когда подгруппа циклическая, порядок подгруппы G_1 равен порядку образующего ее элемента.

Пример 1.120. Порядок преобразования a_1 , описывающего поворот тетраэдра на 120° относительно высоты, проходящей через вершину 4, равен 3.

Задача 1.31. Попробуйте в качестве образующих использовать другие элементы, например:

$$a_9 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix} \quad \text{и} \quad a_{10} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix}.$$

Все ли повороты тетраэдра будут порождаться этими элементами?

Задача 1.32.

1. Найдите другие подгруппы G .

2. Существуют ли у группы поворотов тетраэдра подгруппы порядка 1, 4, 6? Если да, то постройте их.

✓ Изоморфные группы

Группу можно описать и по-другому — таблицей умножения, которую в теории групп называют *таблицей Кэли*.

Пример 1.121. Для описанных выше подгрупп G_1 и G_2 таблицы Кэли имеют вид: таблица 1.50а и б.

Таблица 1.50.

G_1	a_0	a_1	a_2
a_0	a_0	a_1	a_2
a_1	a_1	a_2	a_0
a_2	a_2	a_0	a_1

(a)

G_2	a_0	a_3	a_4
a_0	a_0	a_3	a_4
a_3	a_3	a_4	a_0
a_4	a_4	a_0	a_3

(b)

Подгруппы G_1, G_2 — это группы самосовмещений тетраэдра при поворотах вокруг его высот.

Группы G_1, G_2 изоморфны. Это означает, что можно переименовать элементы одной группы так, что таблица умножения одной группы перейдет в таблицу умножения другой. В нашем примере это достигается таким переименованием:

$$a_1 \mapsto a_3, \quad a_2 \mapsto a_4.$$

Оказывается, подстановками можно описать любую конечную группу. Имеет место следующий результат.

Теорема 1.83. Каждая конечная группа изоморфна некоторой группе подстановок.

✓ Группа симметрий тетраэдра

В заключение рассмотрим группу, определяемую всевозможными подстановками на множестве из четырех элементов.

Подсчитаем число таких подстановок: первый элемент можно переместить в любое из четырех положений, второй — в любое из трех оставшихся, третий — в любое из двух оставшихся, четвертый же поставить на единственно свободное место. Тогда с каждым из четырех способов перемещения первого элемента комбинируется три способа перемещения второго и два способа перемещения третьего, что дает $4 \cdot 3 \cdot 2 = 24$ комбинации.

Из них 12 подстановок соответствуют поворотам тетраэдра. Какой геометрический смысл имеют оставшиеся 12? Возьмем, например, такую под-

становку:

$$a_{12} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 2 & 4 \end{pmatrix}.$$

Она означает такое преобразование тетраэдра, при котором две вершины поменяются местами, а две другие останутся в прежнем положении. Нетрудно увидеть, что такой поворот невозможен.

Однако если зеркально отобразить тетраэдр (рисунок 1.19), то после этого такое совмещение вершин станет возможным.

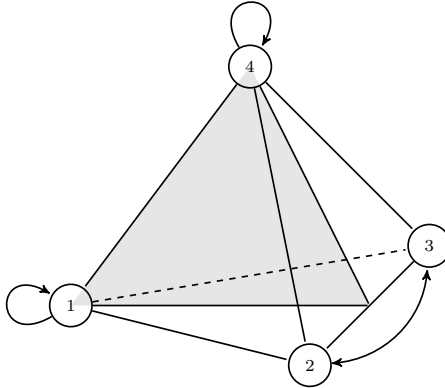


Рис. 1.19.

Таким образом, если к преобразованиям поворота добавить преобразования зеркального отражения относительно сечений тетраэдра, каждое из которых проходит через ребро и соответствующую высоту (см. рисунок 1.19), то все подстановки будут соответствовать некоторым преобразованиям тетраэдра. Порядок этой самой большой группы преобразований будет 24, а группа поворотов станет подгруппой этой группы.

Обратите внимание, что исходная группа поворотных самосовмещений является подгруппой группы симметрий тетраэдра и, соответственно, ее порядок является делителем порядка вновь построенной группы (24 делится на 12).

1.6.16.3. Теорема Бернсайда

Поставим задачу: сколькими способами можно раскрасить грани правильного тетраэдра n красками, если считать разными те раскраски, которые не получаются друг из друга самосовмещениями тетраэдра при поворотах.

Пример 1.122. Рассмотрим раскраску граней тетраэдра двумя красками, например красной и белой.

Перечислим все пять возможных раскрасок:

- все грани белые;
- одна грань красная;
- две грани красные (всего одна возможная раскраска, так как любые две грани имеют общее ребро, значит, любые две такие раскраски перейдут друг в друга при повороте, совмещающем эти ребра);
- три грани красные;
- все грани красные.

Заметим, что если не отождествлять раскраски, совпадающие при поворотных самосовмещениях, то число раскрасок будет $2^4 = 16$.

В общем случае задачу можно поставить так. Имеется множество X (множество раскрасок), на котором определена группа преобразований G (переводящих одни раскраски в другие). Требуется найти количество орбит, т. е. количество подмножеств, на которые разбивается все множество раскрасок по заданной группе G . В одну орбиту попадают все элементы множества, которые переводятся друг в друга некоторым преобразованием группы G .

Пример 1.123. Показанные на рисунке 1.20 раскраски тетраэдра входят в одну орбиту относительно группы поворотных самосовмещений тетраэдра. Преобразования a_7 и a_8 , описанные ранее, переводят их друг в друга соответственно.

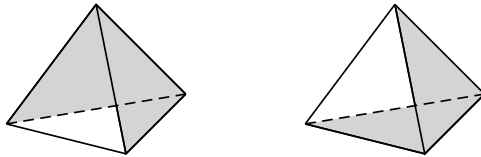


Рис. 1.20.

Теорема Бернсайда позволяет свести нахождение числа орбит к нахождению для каждого преобразования группы числа раскрасок, которые оно не меняет.

Пример 1.124. Преобразование a_1 , описанное ранее, переводит в себя следующие раскраски тетраэдра двумя красками:

- все грани одного цвета (две раскраски);
- основание одного цвета, а все боковые грани — другого (две раскраски).

Теорема 1.84 (Бернсайд). Число орбит получается, если для каждого преобразования найти число раскрасок, которое оно не меняет, результаты сложить и поделить на число элементов группы:

$$|S| = \frac{1}{|G|} \sum_{g \in G} \eta(g). \quad (1.109)$$

 **Замечание 1.65**

| Традиционно это утверждение называют леммой Бернсайда.

Пример 1.125 (к применению теоремы Бернсайда). Для задачи раскраски тетраэдра двумя красками получаем:

$\eta(a_0) = 16$ (тождественное преобразование сохраняет все раскраски);

$\eta(a_1) = \dots = \eta(a_8) = 4$; $\eta(a_9) = \eta(a_{10}) = \eta(a_{11}) = 4$, $|G| = 12$;

$$|S| = \frac{16 + 4 \cdot 11}{12} = \frac{60}{12} = 5.$$

Пример 1.126 (к доказательству теоремы Бернсайда). Основная идея доказательства видна из таблицы 1.51, столбцы которой соответствуют различным преобразованиям, а строки — раскраскам.

Таблица 1.51.

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}
0000	+	+	+	+	+	+	+	+	+	+	+	+
0001								+	+			
0010	+					+	+					
0011	+											+
0100	+			+	+							
0101	+									+		
0110	+										+	
0111	+	+	+									
1000	+	+	+									
1001	+										+	
1010	+									+		
1011	+			+	+							

Продолжение таблицы 1.51

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}
1100	+											+
1101	+					+	+					
1110	+							+	+			
1111	+	+	+	+	+	+	+	+	+	+	+	+

Здесь раскраска граней тетраэдра закодирована следующим образом: 1 — красная, 0 — белая, первый элемент набора — цвет основания, второй — цвет левой грани, третий — цвет задней грани, четвертый — цвет правой грани. В таблице 1.51 знаком «+» отмечены те клетки, для которых соответствующее преобразование (столбец) не меняет соответствующей раскраски (строка).

Объединим в таблице 1.51 клетки тех строк, которые соответствуют одной раскраске (на рисунке 1.20 отмечена одна из таких групп). Заметим, что в каждой группе будет ровно 12 отмеченных клеток, а наша группа имеет 12 элементов. Это мы и должны обосновать в общем случае.

Доказательство теоремы Бернсайда.

Рассмотрим множество пар $(g; x)$ таких, что преобразование g сохраняет раскраску x , т. е. $g(x) = x$. Зафиксируем раскраску x и рассмотрим все преобразования, переводящие ее в себя. Тогда будет верно следующее утверждение.

Лемма 1.18. Множество элементов группы g , обладающих свойством $g(x) = x$, является подгруппой группы G .

Доказательство. Обозначим $G_1 = \{g \mid g(x) = x\}$. Достаточно доказать, что произведение элементов G_1 также является элементом G_1 . Тогда умножение не будет нас выводить за пределы множества G_1 , а выполнимость групповых свойств для элементов G_1 будет иметь место, так как $G_1 \subset G$:

Пусть $g_1, g_2 \in G_1$. Рассмотрим $g_1 \circ g_2 : g_1 \circ g_2(x) = g_1(g_2(x)) = g_1(x) = x$. ■

Лемма 1.19. Если раскраска x может быть переведена преобразованием f группы G в раскраску y , то

$$|G_1| = |G_2|, \text{ где } G_1 = \{g \mid g(x) = x\}, G_2 = \{g \mid g(y) = y\}.$$

Доказательство. Между G_1 и G_2 существует взаимно однозначное соответствие, определяемое через f следующим образом. Если $g \in G_1$, то $f \circ g \circ f^{-1} \in G_2$, и наоборот, если $g \in G_2$, то $f^{-1} \circ g \circ f \in G_1$.

Докажем, к примеру, первое утверждение:

$$\begin{aligned} g \in G_1 \Rightarrow g(x) = x \Rightarrow g(f^{-1}(y)) = f^{-1}(y) \Rightarrow f(g(f^{-1}(y))) = y \Rightarrow \\ \Rightarrow f \circ g \circ f^{-1}(y) = y. \quad \blacksquare \end{aligned}$$

Упражнение 1.20. Докажите, что указанное соответствие взаимно однозначно.

Перейдем теперь к доказательству собственно теоремы Бернсайда. Подсчитаем количество пар $\{(g; x) \mid g(x) = x\}$ двумя способами.

Первый способ состоит в том, чтобы для каждого преобразования найти все раскраски x , которые оно сохраняет (суммирование по столбцам). Обозначим для каждого преобразования g множество раскрасок $\eta(g)$. Тогда количество искомых пар запишется как $\sum_{g \in G} \eta(g)$.

Второй способ состоит в том, чтобы просуммировать преобразования для раскрасок, входящих в одну орбиту (суммирование по группам строк). Докажем, что в каждой такой сумме будет одинаковое число слагаемых, равное порядку группы $|G|$.

Действительно, преобразования, которые сохраняют данную раскраску (строка), образуют подгруппу G_1 группы G (см. лемму 1.18).

Рассмотрим классы смежности группы G по подгруппе $G_1 : fG_1$. По теореме 1.82 о классах смежности классы разбивают все множество преобразований на равные по количеству части, в каждой из которых $|G_1|$ элементов.

С другой стороны, существует взаимно однозначное соответствие между подгруппами преобразований, сохраняющими отдельные элементы орбиты, и классами смежности: по лемме 1.19 преобразование вида $f \circ g \circ f^{-1}$ переводит элементы одной подгруппы в другую.

Преобразование $f \circ g \in fG_1$, а значит, преобразование f^{-1} сопоставляет классу смежности fG_1 подгруппу G_2 . По соображениям, высказанным ранее, это сопоставление взаимно однозначно.

Итак, второй способ подсчета дает: $|S| \times |G|$, где S — множество орбит.

Приравняв полученные в обоих способах вычисления результаты, получим искомую формулу (1.109). \blacksquare

Пример 1.127. Применим теорему Бернсайда для подсчета количества циклических последовательностей длиной $n = 6$ над алфавитом из двух символов $\{A, B\}$, которые были подсчитаны ранее на основе обращения Мебиуса (см. пример 1.111).

Рассмотрим циклическую группу порядка 6. Ее образующий элемент — циклический сдвиг последовательности на одну позицию — задается подстановкой:

$$a_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 1 \end{pmatrix}.$$

«Раскраска» в данном случае означает конкретную строку из шести символов A и B , «орбита» — раскраски переходящие друг в друга при циклических сдвигах.

Подсчитаем для каждого преобразования число сохраняемых им раскрасок.

Тождественное преобразование

$$a_0 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix} = (1)(2)(3)(4)(5)(6)$$

сохраняет все 2^6 последовательностей.

Преобразования

$$a_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 1 \end{pmatrix} = (123456), \quad a_5 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 1 & 2 & 3 & 4 & 5 \end{pmatrix} = (165432)$$

сохраняют по две последовательности каждая (последовательности, состоящие из одинаковых символов).

Преобразование

$$a_3 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 1 & 2 & 3 \end{pmatrix} = (14)(25)(36)$$

сохраняет 2^3 последовательностей (символы на местах, входящих в один цикл подстановки, должны быть одинаковы).

Преобразования

$$a_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 1 & 2 \end{pmatrix} = (135)(246), \quad a_4 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 6 & 1 & 2 & 3 & 4 \end{pmatrix} = (153)(264)$$

сохраняют по 2^2 последовательности каждая.

Подставив результаты в формуле Бернсайда (1.109), получим

$$|S| = \frac{1}{6}(2^6 + 2 \cdot 2 + 2^3 + 2 \cdot 2^2) = \frac{84}{6} = 14.$$

Задачи и вопросы

1. Обладает ли данный код свойством префикса:

$ac, c, bb, ca, abc, bac, cab, abb, bcab, abcb, abbc?$

2. Расставьте в лексикографическом (антилексикографическом) порядке слова:

$(dcda), (badc), (cabd), (abdc), (bbaa).$

3. Как улучшить двоичный код, чтобы не было резких изменений между соседними элементами, например $(0, 1, 1, 1), (1, 0, 0, 0)$.

4. Найдите число целых слагаемых бинома $(\sqrt[3]{3} + \sqrt[5]{5})^{35}$.

5. Найдите коэффициент при x^5 в разложении $(1 + x - x^3)^6$.

6. Имеется n черных и m белых шаров. Сколькими способами можно их выложить в ряд так, чтобы никакие два черных шара не лежали рядом?

7. Сколькими способами из 28 костей домино можно выбрать две кости так, чтобы их можно было приложить друг к другу (т. е. чтобы некоторое одинаковое число очков встретилось на обеих костях)?

8. Будем говорить, что два числа образуют инверсию, если большее из них написано ранее, чем меньшее. Сколько инверсий во всех перестановках чисел $\{1, \dots, n\}$?

9. Определите количество двоичных 15-значных чисел, имеющих в записи 11 единиц.

10. На окружности последовательно отмечены точки A_1, \dots, A_n . Сколько существует хорд с концами в отмеченных точках?

11. На окружности отмечено n точек. Точки соединяются всевозможными хордами; известно, что никакие три из них не пересекаются в одной точке внутри круга. Найти число точек пересечения хорд внутри круга.

12. Из колоды, содержащей 52 карты, вынули 10 карт.

а) В скольких случаях среди этих карт окажется хотя бы один туз?

б) В скольких случаях — ровно один туз?

13. Сколько существует решений уравнения $x_1 + x_2 + \dots + x_{15} = 200$ в целых числах, где $x_i \geq 1$?

14. Сколько существует 5-значных чисел в 14-ричной системе счисления, у которых не все цифры разные?

15. Все перестановки 7 чисел $\{1, 2, 3, 4, 5, 6, 7\}$ упорядочены в лексикографическом порядке.

- а) Какой по счету идет перестановка $\{4, 6, 5, 3, 7, 2, 1\}$?
- б) Найдите перестановку с номером 2546.

16. Перестановки чисел от 1 до n упорядочены в лексикографическом порядке (первая перестановка $(1, 2, 3, \dots, n)$). Оцените утверждения:

- а) перестановка $(2, 1, 3, \dots, n)$ стоит на 121-м месте;
- б) единица стоит на последнем месте в 20 перестановках.

17. Среди 100 целых чисел 40 кратно 3, 3 кратно 11, 18 кратно 9, 2 кратно 33, 1 кратно 99. Определить, сколько из них кратно 3 или 11, но не кратно 9.

18. Каждый студент в группе знает хотя бы один иностранный язык. Шестеро знают английский, шестеро — немецкий, семеро — французский. Четверо знают английский и немецкий, трое — немецкий и французский, двое — французский и английский. Один человек знает все три языка.

- а) Сколько человек в группе?
- б) Сколько из них знают только английский язык?

19. Экзамен по математике содержал три задачи: по алгебре, геометрии и тригонометрии. Из 750 абитуриентов задачу по алгебре решили 400 абитуриентов, по геометрии — 480, по тригонометрии — 420. Задачи по алгебре или геометрии решили 630 абитуриентов; по геометрии или тригонометрии — 600 абитуриентов; по алгебре или тригонометрии — 620 абитуриентов. 100 абитуриентов не решили ни одной задачи.

- а) Сколько абитуриентов решили все задачи?
- б) Сколько абитуриентов решили только одну задачу?

20. Сколькими способами можно окрасить в три цвета:

- а) грани правильного тетраэдра;
- б) ребра правильного тетраэдра;
- в) вершины правильного тетраэдра.

Как и раньше, считаем раскраски одинаковыми, если их можно получить друг из друга самосовмещениями тетраэдра при поворотах.

21. Решите ту же задачу, если кроме указанных одинаковыми считаются раскраски, которые переходят друг в друга при самосовмещениях посредством зеркальной симметрии.

22. Решите предыдущие две задачи для куба, октаэдра, правильной треугольной призмы и раскраски n красками.

23. Сколькими способами можно раскрасить ребра куба в 6 красок так, чтобы одинаково окрашенных ребер было ровно по два (6 пар по 2 ребра). Одинаковыми считаются раскраски, переходящие друг в друга поворотными самосовмещениями.

24. Сколькими способами можно собрать ожерелье из 12 бусинок, из которых 6 бусинок красного цвета, 4 зеленого и 2 синего.

25. Пусть имеется 6 кодовых символов: D, E, H, T, C, Y с частотами появления:

D	E	H	T	C	Y
0.20	0.21	0.15	0.17	0.18	0.09

Постройте код Шеннона — Фано и зашифруйте сообщение: $CTUDENT$.

26. Порождающий многочлен линейного циклического кода длиной $n = 7$, который осуществляет кодирование сообщений длиной $m = 4$, имеет вид $g(x) = 1 + x + x^3$. Декодируйте сообщение $(1, 1, 1, 0, 1, 1, 1)$, если известно, что оно имеет не более одной ошибки.

1.7. Производящие функции и рекуррентные уравнения

1.7.1. Производящие функции

Рассмотрим сначала несколько примеров.

Пример 1.128. Дана полоска шириной 2 и длиной n . Заполняем ее домино — прямоугольниками 1×2 или 2×1 (рисунок 1.21).



Рис. 1.21.

Поставим задачу — перебрать все варианты заполнения полоски костяшками домино для последовательных значений n .

Можно заметить, что горизонтально расположенные домино встречаются только парами. Поэтому, обозначив вертикальное домино буквой x , а пару горизонтальных — y , каждому заполнению полоски можно сопоставить алгебраическое выражение.

Например,

$$xuxu \dots, \quad yxxu \dots$$

Все такие комбинации можно записать в виде суммы (ряда, если рассматривать полоски произвольной длины):

$$x + y + xx + xy + yx \dots$$

Теперь введем новую интерпретацию перемножения алгебраических выражений, соответствующих заполнению полосок. Будем рассматривать умножение двух заполнений полосок одинаковой ширины как простое соединение этих заполненных полосок в одну с большей длиной.

Например,

$$(xxy)(yx) = xxuyx.$$

Теперь перепишем ряд следующим образом:

$$x(1 + x + y + xx + xy + yx + \dots) + y(1 + x + y + xx + xy + yx + \dots).$$

Это корректно, так как если отделить левое вертикальное домино от всех заполнений бесконечной полоски, которые с него начинаются, то опять

получим все возможные заполнения бесконечной полосы домино! То же можно сказать и про отделение двух горизонтально расположенных домино.

Что означает 1 в этой сумме? «Пустое домино» означает, что добавление его к полоске не меняет ее заполнения.

Таким образом, если обозначить все возможные полосы шириной 2, полученные соединением домино (включая пустую) как S , то можно записать уравнение

$$S = 1 + xS + yS.$$

Используя формально привычные алгебраические операции, получим

$$S = \frac{1}{1 - x - y}.$$

Далее, опять же формально, применим формулу для бесконечно убывающей геометрической прогрессии

$$\frac{1}{1 - z} = 1 + z + z^2 + z^3 + \dots$$

и после подстановки $(x + y)$ вместо z будем иметь

$$\frac{1}{1 - x - y} = 1 + (x + y) + (x + y)^2 + (x + y)^3 + \dots$$

Как интерпретировать этот результат?

Например, если нас интересует число полосок, которые можно составить из трех вертикальных и четырех (двух пар) горизонтальных домино, то надо взять слагаемое $(x + y)^{3+2}$ и найти коэффициент при x^3y^2 . Как известно, это биномиальный коэффициент $C_5^3 = 10$.

Таким образом, произведя формальные операции, мы сопоставили бесконечному ряду величину S , являющуюся функцией от x и y :

$$S(x, y) = \frac{1}{1 - x - y} = \sum_{k=0}^{\infty} (x + y)^k.$$

Пример 1.129. Рассмотрим соотношения:

$$1 + 2 + \dots + n = \frac{n(n + 1)}{2}, \quad (1.110)$$

$$1^2 + 2^2 + \dots + n^2 = \frac{n(n + 1)(2n + 1)}{6}. \quad (1.111)$$

Если обозначить сумму в левой части (1.110) через f_n , а сумму в левой части (1.111) через g_n , то будут иметь место соотношения:

$$f_{n+1} - f_n = n + 1; \quad g_{n+1} - g_n = (n + 1)^2. \quad (1.112)$$

Взяв в качестве начальных значений $f_1 = g_1 = 1$, с помощью (1.112) можно определить f_2 и g_2 , затем — f_3 и g_3 и т. д. Таким образом последовательности $\{f_n\}$ и $\{g_n\}$ однозначно определяются соотношениями (1.112) и начальными условиями.

Обобщением (1.112) является *линейное рекуррентное¹ уравнение с постоянными коэффициентами*

$$f_{n+r} + a_1 f_{n+r-1} + a_2 f_{n+r-2} + \dots + a_r f_n = \varphi(n), \quad (1.113)$$

выполненное для всех неотрицательных целых чисел n . Коэффициенты a_1, \dots, a_r — фиксированные числа, причем $a_r \neq 0$, $\varphi(n)$ — заданная функция от n . Число r называют *порядком* уравнения.

Замечание 1.66

Если $a_r = 0$, то уравнение (1.113) можно привести к уравнению меньшего порядка.

Если зафиксировать значения f_0, f_1, \dots, f_{r-1} и рассматривать их как начальные условия, то с помощью рекуррентного уравнения (1.113) можно шаг за шагом однозначно определять значения f_r, f_{r+1}, \dots и таким образом определить всю последовательность $\{f_n\}$.

Поставим в соответствие последовательности $\{f_n\}$ степенной ряд

$$F(z) = f_0 + f_1 z + f_2 z^2 + \dots, \quad (1.114)$$

который называют *производящей функцией* для последовательности $\{f_n\}$.

Замечание 1.67

Последовательность чисел однозначно определяет производящую функцию, но обратное утверждение верно не всегда. Из теории аналитических функций известно, что если указанный степенной ряд сходится в круге радиусом $R > 0$, то коэффициенты f_0, f_1, \dots определяются по $F(z)$ однозначно.

В дальнейшем нам будет неважно, при каких значениях z сходится соответствующий ряд. Мы не будем вычислять значение производящей функции в конкретной точке z , будем только выполнять некоторые формальные операции на таких рядах, а затем сравнивать коэффициенты при отдельных степенях z .

Рассмотрим некоторые свойства производящих функций. Пусть

$$G(z) = g_0 + g_1 z + g_2 z^2 + \dots$$

¹ От лат. *recurrens* — возвращающийся.

— производящая функция для последовательности $\{g_n\}$, а c и d — произвольные фиксированные числа. Поскольку

$$cF(z) + dG(z) = (cf_0 + dg_0) + (cf_1 + dg_1)z + \dots,$$

то последовательности $\{cf_n + dg_n\}$ отвечает производящая функция

$$cF(z) + dG(z).$$

Возьмем произведение производящих функций

$$H(z) = F(z)G(z) = h_0 + h_1z + h_2z^2 + \dots,$$

тогда последовательность чисел $\{h_n\}$ может быть получена из последовательностей $\{f_n\}$ и $\{g_n\}$ с помощью соотношений

$$h_n = f_0g_n + f_1g_{n-1} + \dots + f_{n-1}g_1 + f_ng_0. \quad (1.115)$$

Пример 1.130. Найдем производящие функции для последовательностей чисел $\{1\}$ и $\{n\}$. Для первой последовательности получаем

$$1 + z + z^2 + \dots = \frac{1}{1-z}. \quad (1.116)$$

Последовательности $\{n\}$ соответствует ряд

$$z + 2z^2 + 3z^3 + \dots = z(1 + 2z + 3z^2 + \dots). \quad (1.117)$$

Поскольку выражение в скобках в правой части (1.117) получается дифференцированием ряда (1.116), то

$$z(1 + 2z + 3z^2 + \dots) = z \frac{d}{dz} \left(\frac{1}{1-z} \right) = \frac{z}{(1-z)^2}.$$

Пример 1.131. Обобщая предыдущий пример для последовательности (1.116), легко найти производящую функцию для последовательности $\{a^k\}$.

$$1 + az + a^2z^2 + a^3z^3 + \dots = \frac{1}{1-az}. \quad (1.118)$$

Пример 1.132. Найдем производящую функцию для разбиений числа n на не более чем k слагаемых. Обозначим число таких разбиений $p_k(n)$, а саму производящую функцию — $P_k(z)$. В силу свойств сопряженных разбиений $p_k(n)$ равно числу разбиений числа n с наибольшим слагаемым, не превосходящим k .

Такие разбиения можно рассматривать как решения уравнения

$$\alpha_1 + 2\alpha_2 + \dots + k\alpha_k = n, \quad \alpha_i \in \mathbb{Z}, \quad \alpha_i \geq 0,$$

где α_i — кратность слагаемого i в разбиении. Тогда согласно (1.114) имеем

$$\begin{aligned} P_k(z) &= p_k(0) + p_k(1)z + p_k(2)z^2 + \dots = \sum_{n=0}^{\infty} p_k(n)z^n = \sum_{n=0}^{\infty} \sum_{\alpha_1 + \dots + k\alpha_k = n} z^n = \\ &= \sum_{\alpha_1 \geq 0} \dots \sum_{\alpha_k \geq 0} z^{\alpha_1 + \dots + k\alpha_k} = \left(\sum_{\alpha_1 \geq 0} z^{\alpha_1} \right) \dots \left(\sum_{\alpha_k \geq 0} z^{k\alpha_k} \right). \end{aligned}$$

Применив (1.116), получим

$$P_k(z) = \frac{1}{1-z} \dots \frac{1}{1-z^k} = \frac{1}{(1-z) \dots (1-z^k)} = \prod_{i=1}^k (1-z^i)^{-1}.$$

Используем теперь производящие функции для решения линейного рекуррентного уравнения.

Пример 1.133. Требуется решить уравнение $f_{n+2} - f_{n+1} - f_n = 0$ с начальными условиями $f_0 = f_1 = 1$.

Обозначим через $F(z)$ производящую функцию последовательности чисел $\{f_n\}$. Умножим обе части рекуррентного уравнения на z^{n+2} , получим

$$f_{n+2}z^{n+2} - zf_{n+1}z^{n+1} - z^2f_nz^n = 0, \quad n = 0, 1, 2, \dots$$

Сложив эти равенства для всех n от 0 до ∞ , имеем

$$\sum_{n=0}^{\infty} f_{n+2}z^{n+2} - z \sum_{n=0}^{\infty} f_{n+1}z^{n+1} - z^2 \sum_{n=0}^{\infty} f_nz^n = 0. \quad (1.119)$$

Заметим, что первая сумма в (1.119) равна разности функции $F(z)$ и первых двух членов ее разложения $f_0 + f_1z = 1 + z$, вторая сумма — разности $F(z)$ и первого члена $f_0 = 1$, а третья сумма — в точности $F(z)$. Поэтому можно записать следующее равенство:

$$[F(z) - (1+z)] - z[F(z) - 1] - z^2F(z) = 0.$$

Откуда находим

$$F(z) = \frac{1}{1-z-z^2}. \quad (1.120)$$

Разложим правую часть (1.120) на простейшие дроби. Для этого решим

уравнение $1 - z - z^2 = 0$. Имеем

$$1 - z - z^2 = -(z_1 - z)(z_2 - z) = -z_1 z_2 \left(1 - \frac{z}{z_1}\right) \left(1 - \frac{z}{z_2}\right),$$

$$\text{где } z_1 = -\frac{1}{2} + \frac{\sqrt{5}}{2}, \quad z_2 = -\frac{1}{2} - \frac{\sqrt{5}}{2}.$$

Положим

$$\alpha_1 = \frac{1}{z_1} = \frac{1 + \sqrt{5}}{2}, \quad \alpha_2 = \frac{1}{z_2} = \frac{1 - \sqrt{5}}{2}.$$

Заметим, что $z_1 z_2 = -1$. Тогда $1 - z - z^2 = (1 - \alpha_1 z)(1 - \alpha_2 z)$ и, учитывая (1.118), окончательно получаем

$$F(z) = \frac{1}{\sqrt{5}} \left(\frac{\alpha_1}{1 - \alpha_1 z} - \frac{\alpha_2}{1 - \alpha_2 z} \right) = \sum_{n=0}^{\infty} \frac{1}{\sqrt{5}} (\alpha_1^{n+1} - \alpha_2^{n+1}) z^n.$$

Таким образом,

$$f_n = \frac{1}{\sqrt{5}} (\alpha_1^{n+1} - \alpha_2^{n+1}), \quad n = 0, 1, 2, \dots \quad (1.121)$$

Формула (1.121) носит название *формулы Бине*. Члены последовательности $\{f_n\}$ уже появлялись в разделе 1.2.7, далее были подробно рассмотрены в разделе 1.2.12 и известны как числа Фибоначчи.

При больших n члены приближенно равны $\alpha_1^{n+1} \sqrt{5}$. Как было указано в том же подразделе, коэффициент $\alpha_1 = 1.6180337 \dots$ известен в геометрии как «золотое сечение» или число Фидия.

Историческая справка

Формула (1.121) впервые опубликована Даниэлем Бернулли в 1728 г. и вновь открыта Жаком Бине в 1843 г.

1.7.2. Решение однородного линейного рекуррентного уравнения

Найдем общее решение однородного линейного рекуррентного уравнения, получающегося из (1.113) при $\varphi(n) = 0$. Метод решения является обобщением решения примера 1.133.

Предположим, что последовательность чисел $\{f_n\}$ удовлетворяет следующему однородному линейному рекуррентному уравнению:

$$f_{n+r} + a_1 f_{n+r-1} + a_2 f_{n+r-2} + \dots + a_r f_n = 0, \quad (1.122)$$

где a_1, a_2, \dots, a_r — заданные постоянные коэффициенты и $a_r \neq 0$.

Ограничимся случаем $r = 2$:

$$f_{n+2} + a_1 f_{n+1} + a_2 f_n = 0. \quad (1.123)$$

Для задания начальных условий фиксируем значения f_0, f_1 . Как и в примере 1.133, обозначим через $F(z)$ производящую функцию последовательности f_0, f_1, \dots . По заданным постоянным коэффициентам уравнения (1.123) построим многочлен

$$K(z) = 1 + a_1 z + a_2 z^2.$$

Этот многочлен можно считать производящей функцией последовательности $\{1, a_1, a_2, 0, \dots\}$.

Рассмотрим произведение производящих функций $C(z) = K(z)F(z)$. Согласно (1.115), коэффициент c_{n+2} при z^{n+2} определяется соотношением

$$c_{n+2} = f_0 \cdot 0 + f_1 \cdot 0 + \dots + f_{n-1} \cdot 0 + f_n a_2 + f_{n+1} a_1 + f_{n+2} = f_n a_2 + f_{n+1} a_1 + f_{n+2}.$$

В силу (1.123) $c_{n+2} = 0$ для $n = 0, 1, \dots$. Таким образом, производящая функция $C(z)$ имеет вид $C(z) = c_0 + c_1 z$. Тогда

$$F(z) = \frac{C(z)}{K(z)} = \frac{c_0 + c_1 z}{1 + a_1 z + a_2 z^2}.$$

Определение 1.64. *Характеристическим многочленом* линейного рекуррентного уравнения (1.122) называют многочлен степени r :

$$G(z) = z^r + a_1 z^{r-1} + a_2 z^{r-2} + \dots + a_r.$$

Корни этого многочлена называют *характеристическими*. При $r = 2$ многочлен имеет вид

$$G(z) = (z - \alpha_1)(z - \alpha_2).$$

Характеристический многочлен $G(z)$ и многочлен $K(z)$ связаны между собой соотношением

$$K(z) = z^2 G\left(\frac{1}{z}\right),$$

откуда получаем

$$K(z) = (1 - \alpha_1 z)(1 - \alpha_2 z), \text{ а } F(z) = \frac{c_0 + c_1 z}{(1 - \alpha_1 z)(1 - \alpha_2 z)}.$$

Возможны два случая:

1) корни характеристического многочлена $G(z)$ различны ($\alpha_1 \neq \alpha_2$), тогда

$$F(z) = \frac{c_0 + c_1 z}{(1 - \alpha_1 z)(1 - \alpha_2 z)} = \frac{b_0}{1 - \alpha_1 z} + \frac{b_1}{1 - \alpha_2 z};$$

2) характеристический многочлен $G(z)$ имеет корень кратности 2 ($\alpha_1 = \alpha_2$), тогда

$$F(z) = \frac{c_0 + c_1 z}{(1 - \alpha_1 z)^2} = \frac{b_0}{1 - \alpha_1 z} + \frac{b_1}{(1 - \alpha_1 z)^2}.$$

В примере 1.131 было показано, что

$$1 + \alpha z + \alpha^2 z^2 + \alpha^3 z^3 + \dots = \sum_{n=0}^{\infty} (\alpha z)^n = \frac{1}{1 - \alpha z}.$$

Продифференцировав последнее равенство и разделив обе части на α , получаем

$$1 + 2\alpha z + 3\alpha^2 z^2 + 4\alpha^3 z^3 + \dots = \sum_{n=0}^{\infty} (\alpha z)^n (n+1) = \frac{1}{(1 - \alpha z)^2}.$$

Тогда для случая различных корней получаем

$$F(z) = b_0 \sum_{n=0}^{\infty} (\alpha_1 z)^n + b_1 \sum_{n=0}^{\infty} (\alpha_2 z)^n = \sum_{n=0}^{\infty} (b_0 \alpha_1^n + b_1 \alpha_2^n) z^n \text{ и}$$

$$f_n = b_0 \alpha_1^n + b_1 \alpha_2^n, \quad n = 0, 1, \dots$$

Если же корень один кратности 2, то

$$F(z) = b_0 \sum_{n=0}^{\infty} (\alpha_1 z)^n + b_1 \sum_{n=0}^{\infty} (\alpha_1 z)^n (n+1) = \sum_{n=0}^{\infty} (b_0 \alpha_1^n + b_1 (n+1) \alpha_1^n) z^n \text{ и}$$

$$f_n = (b_0 + b_1 + b_1 n) \alpha_1^n, \quad n = 0, 1, \dots$$

Полученные результаты можно обобщить на случай линейного рекуррентного уравнения порядка r .

1) Если все характеристические корни $\alpha_1, \alpha_2, \dots, \alpha_r$ являются простыми, то общее решение однородного уравнения (1.122) имеет вид

$$f_n = c_1 \alpha_1^n + c_2 \alpha_2^n + \dots + c_r \alpha_r^n. \quad (1.124)$$

2) Если характеристические корни $\alpha_1, \alpha_2, \dots, \alpha_s$ имеют соответственно кратности e_1, e_2, \dots, e_s , т. е.

$$G(z) = (z - \alpha_1)^{e_1} (z - \alpha_2)^{e_2} \dots (z - \alpha_s)^{e_s}, \text{ и } e_1 + e_2 + \dots + e_s = r,$$

то общее решение однородного уравнения (1.122) имеет вид

$$f_n = \sum_{i=1}^s p_i(n) \alpha_i^n, \text{ где } p_i(n) = c_{i,0} + c_{i,1}n + \dots + c_{i,e_i-1}n^{e_i-1}$$

— многочлен степени $e_i - 1$.

Для определения r неопределенных постоянных используются r граничных условий, а именно значения f_0, f_1, \dots, f_{r-1} .

Пример 1.134 (числа Фибоначчи). Описанным методом найдем общий член последовательности чисел Фибоначчи (см. пример 1.133).

Уравнение $f_{n+2} - f_{n+1} - f_n = 0$ имеет характеристический многочлен

$$z^2 - z - 1 = (z - \alpha_1)(z - \alpha_2), \text{ где } \alpha_1 = \frac{1 + \sqrt{5}}{2}, \alpha_2 = \frac{1 - \sqrt{5}}{2}.$$

Так как α_1 и α_2 являются простыми корнями, то $f_n = c_1 \alpha_1^n + c_2 \alpha_2^n$, где c_1, c_2 — неопределенные постоянные. Подставив $f_0 = f_1 = 1$, получим систему уравнений:

$$c_1 + c_2 = 1, \quad c_1 \alpha_1 + c_2 \alpha_2 = 1.$$

Решив эти уравнения, находим:

$$c_1 = \frac{\alpha_2 - 1}{\alpha_2 - \alpha_1} = \frac{1}{\sqrt{5}} \alpha_1; \quad c_2 = \frac{\alpha_1 - 1}{\alpha_1 - \alpha_2} = -\frac{1}{\sqrt{5}} \alpha_2.$$

Отсюда получаем формулу Бине (1.121).

Пример 1.135 (полиномы Чебышева). Рассмотрим функции $T_n(x)$, определяемые однородным рекуррентным уравнением второго порядка:

$$T_{n+2}(x) = 2xT_{n+1}(x) - T_n(x), \quad T_0(x) = 1, \quad T_1(x) = x. \quad (1.125)$$

Характеристический многочлен для уравнения (1.125) имеет два корня:

$$\alpha_1 = x + \sqrt{x^2 + 1}, \quad \alpha_2 = x - \sqrt{x^2 + 1}.$$

Согласно (1.124) общее решение (1.125) можно представить в виде

$$T_n(x) = c_1 \left(x + \sqrt{x^2 + 1} \right)^n + c_2 \left(x - \sqrt{x^2 + 1} \right)^n. \quad (1.126)$$

Используя начальные условия (1.125), находим, что $c_1 = c_2 = \frac{1}{2}$. Тогда (1.126) запишем в следующем виде:

$$T_n(x) = \frac{1}{2} \left[\left(x + \sqrt{x^2 + 1} \right)^n + \left(x - \sqrt{x^2 + 1} \right)^n \right]. \quad (1.127)$$

Определение 1.65. Многочлены $T_n(x)$, определенные в (1.127), называют *ортгоналными полиномами Чебышева первого рода*. Первые полиномы Чебышева имеют вид:

$$T_0(x) = 1, T_1(x) = x, T_2(x) = 2x^2 - 1, T_3(x) = 4x^3 - x, T_4(x) = 8x^4 - 8x^2 + 1.$$

Полиномы Чебышева можно задать в аналитическом виде:

$$T_n(x) = \cos(n \arccos(x)).$$

Если в уравнении (1.125) изменить начальные условия: $T_0(x) = 1$, $T_1(x) = 2x$, то его решения (обозначим их $U_n(x)$) называют *ортгоналными полиномами Чебышева второго рода*.

Упражнение 1.21. Выведите рекуррентные формулы для полиномов Чебышева второго рода $U_n(x)$.

Пример 1.136. Вычислим определитель матрицы размером $n \times n$ (обозначим его через f_n):

$$f_n = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & 1 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 1 \end{vmatrix}$$

Разложив определитель по первому столбцу, получим следующее рекуррентное уравнение для f_n :

$$f_{n+2} = f_{n+1} - f_n, \text{ при этом очевидно, что } f_1 = 1, f_2 = 0.$$

Исходя из этого, положим $f_0 = 1$. Данное уравнение имеет характеристический многочлен

$$G(z) = z^2 - z + 1 = (z - \alpha_1)(z - \alpha_2), \text{ где}$$

$$\alpha_1 = \frac{1}{2} + i\frac{\sqrt{3}}{2} = e^{i(\pi/3)}, \quad \alpha_2 = \frac{1}{2} - i\frac{\sqrt{3}}{2} = e^{-i(\pi/3)}.$$

Общее решение имеет вид:

$$f_n = c_1\alpha_1^n + c_2\alpha_2^n = c_1e^{i(\pi/3)n} + c_2e^{-i(\pi/3)n}.$$

Подставив $f_0 = f_1 = 1$, получим систему уравнений:

$$c_1 + c_2 = 1, \quad c_1\alpha_1 + c_2\alpha_2 = 1.$$

Отсюда, в свою очередь, имеем:

$$c_1 = \frac{1}{2} - i\frac{1}{2\sqrt{3}}; \quad c_2 = \frac{1}{2} + i\frac{1}{2\sqrt{3}}.$$

Следовательно,

$$f_n = \left(\frac{1}{2} - i\frac{1}{2\sqrt{3}}\right) \left(\cos\frac{\pi}{3}n + i\sin\frac{\pi}{3}n\right) + \left(\frac{1}{2} + i\frac{1}{2\sqrt{3}}\right) \times$$

$$\times \left(\cos\frac{\pi}{3}n - i\sin\frac{\pi}{3}n\right) = \cos\frac{\pi}{3}n + \frac{1}{\sqrt{3}}\sin\frac{\pi}{3}n.$$

Пример 1.137. Найдем решение уравнения $f_{n+2} - 4f_{n+1} + 4f_n = 0$ с граничными условиями $f_0 = 1, f_1 = 4$.

Так как характеристический многочлен $z^2 - 4z + 4 = (z - 2)^2$ имеет корень 2 кратности 2, то общее решение $f_n = (c_0 + c_1n)2^n$. С помощью граничных условий находим

$$c_0 = 1, \quad 2(c_0 + c_1) = 4, \text{ т. е. } c_1 = 1.$$

Таким образом, решение рассматриваемого уравнения имеет вид

$$f_n = (1 + n)2^n.$$

Пример 1.138. Найдем решение однородного рекуррентного уравнения

$$2f_{n+2} + 5f_{n+1} - 3f_n = 0$$

с граничными условиями $f_0 = 0, f_1 = 7$.

Характеристический многочлен этого уравнения

$$2z^2 + 5z - 3 = (2z - 1)(z + 3)$$

имеет два корня $\alpha_1 = \frac{1}{2}$, $\alpha_2 = -3$. Тогда общее решение уравнения имеет вид

$$f_n = b_0 \left(\frac{1}{2}\right)^n + b_1 (-3)^n.$$

Используя граничные условия, получаем систему уравнений относительно b_0 и b_1 .

$$b_0 + b_1 = 0, \quad \frac{1}{2}b_0 - 3b_1 = 7.$$

Отсюда $b_0 = 2$, $b_1 = -2$. Таким образом, решение рассматриваемого уравнения имеет вид

$$f_n = \left(\frac{1}{2}\right)^{n-1} - 2(-3)^n.$$

Пример 1.139. Найдем решение однородного рекуррентного уравнения

$$9f_{n+2} - 6f_{n+1} + f_n = 0$$

с граничными условиями $f_0 = 1$, $f_1 = 1$.

Характеристический многочлен этого уравнения

$$9z^2 - 6z + 1 = (3z - 1)^2$$

имеет один корень $\alpha = \frac{1}{3}$ кратности 2. Тогда общее решение уравнения имеет вид

$$f_n = (b_0 + b_1 n) \left(\frac{1}{3}\right)^n.$$

Используя граничные условия, получаем систему уравнений относительно b_0 и b_1 .

$$b_0 = 1, \quad \frac{1}{3}(b_0 + b_1) = 1.$$

Отсюда $b_0 = 1$, $b_1 = 2$. Таким образом, решение рассматриваемого уравнения имеет вид

$$f_n = (1 + 2n) \left(\frac{1}{3}\right)^n.$$

1.7.3. Решение неоднородного линейного рекуррентного уравнения

Справедлива следующая теорема.

Теорема 1.85. Пусть f_n — общее решение однородного уравнения, соответствующего уравнению (1.113). Тогда общее решение неоднородного уравнения g_n имеет вид

$$g_n = f_n + h_n,$$

где h_n — любое частное решение уравнения (1.113).

Для произвольного уравнения (1.113) найти его частное решение h_n достаточно трудно, однако в частном случае, когда правая часть $\varphi(n)$ имеет специальный вид:

$$\varphi(n) = c_0 + c_1n + c_2n^2 + \dots + c_m n^m.$$

Метод нахождения дает следующая теорема.

Теорема 1.86. Пусть 1 — корень кратности s ($s \geq 0$) характеристического уравнения для (1.113). Тогда h_n имеет вид

$$h_n = b_0n^s + b_1n^{s+1} + \dots + b_m n^{s+m}.$$

где b_i можно найти методом неопределенных коэффициентов.

Пример 1.140. Рассмотрим линейное неоднородное уравнение первого порядка (1.112):

$$f_{n+1} - f_n = (n+1)^2.$$

Его характеристическое уравнение имеет один корень 1. Согласно теореме 1.86 частное решение ищем в виде

$$h_n = b_0n + b_1n^2 + b_2n^3.$$

Подставим h_n в уравнение, тогда

$$b_0(n+1) + b_1(n+1)^2 + b_2(n+1)^3 - b_0n - b_1n^2 - b_2n^3 = (n+1)^2.$$

Приведем подобные члены, тогда

$$b_0 + b_1 + b_2 + (2b_1 + 3b_2)n + 3b_2n^2 = (n+1)^2.$$

Приравняв в последнем уравнении коэффициенты при одинаковых степенях n , находим

$$b_0 = \frac{1}{6}, b_1 = \frac{1}{2}, b_2 = \frac{1}{3}, h_n = \frac{n}{6}(1 + 3n + 2n^2) = \frac{n(n+1)(2n+1)}{6}.$$

Пример 1.141. Рассмотрим уравнение

$$f_{n+2} - 5f_{n+1} + 6f_n = n^2 + 2n - 1.$$

Его характеристическое уравнение имеет два корня 2 и 3. Согласно теореме 1.86 частное решение ищем в виде

$$h_n = b_0 + b_1n + b_2n^2.$$

Подставим h_n в уравнение, тогда

$$b_0 + b_1(n+2) + b_2(n+2)^2 - 5(b_0 + b_1(n+1) + b_2(n+1)^2) + 6(b_0 + b_1n + b_2n^2) = 2b_0 - 3b_1 - b_2 + (2b_1 - 6b_2)n + 2b_2n^2 = n^2 + 2n - 1.$$

Приравняв в этом уравнении коэффициенты при одинаковых степенях n , находим

$$b_0 = \frac{7}{2}, b_1 = \frac{5}{2}, b_2 = \frac{1}{2}, h_n = \frac{7 + 5n + n^2}{2}.$$

Пример 1.142. Найдем общее решение неоднородного рекуррентного уравнения

$$f_{n+2} - 3f_{n+1} + 2f_n = n^2 + 1.$$

Характеристический многочлен этого уравнения

$$z^2 - 3z + 2 = (z-1)(z-2)$$

имеет два корня $\alpha_1 = 1, \alpha_2 = 2$. Тогда общее решение однородного уравнения имеет вид

$$f_n = b_0 + b_12^n.$$

Частное решение неоднородного уравнения будем искать в виде

$$h_n = a_0n + a_1n^2 + a_2n^3.$$

Подставляя в исходное уравнение, имеем

$$a_0(n+2) + a_1(n+2)^2 + a_2(n+2)^3 - 3(a_0(n+1) + a_1(n+1)^2 + a_2(n+1)^3) + 2(a_0n + a_1n^2 + a_2n^3) = n^2 + 1.$$

Приводя подобные члены, получаем

$$-a_0 + a_1 + 5a_2 + (-2a_1 + 3a_2)n + (3a_1 - 3a_2)n^2 = n^2 + 1.$$

Приравнявая коэффициенты при одинаковых степенях многочленов, имеем систему уравнений относительно a_0, a_1, a_2 :

$$-a_0 + a_1 + 5a_2 = 1, \quad -2a_1 + 3a_2 = 0, \quad 3a_1 - 3a_2 = 1.$$

Отсюда:

$$a_0 = -\frac{8}{15}, \quad a_1 = -\frac{1}{5}, \quad a_2 = -\frac{2}{15}.$$

Тогда общее решение неоднородного уравнения имеет вид

$$g_n = f_n + h_n = b_0 + b_1 2^n - \frac{8n}{15} - \frac{n^2}{5} - \frac{2n^3}{15}.$$

1.7.4. Производящая функция для чисел Каталана

Определим бесконечный степенной ряд, взяв в качестве его коэффициентов числа Каталана (см. определение 1.55).

Для нахождения производящей функции для чисел Каталана используем рекуррентные соотношения. Найдем рекуррентную формулу для чисел Каталана.

Рассмотрим для этого модель с расстановкой скобок в произведении матриц: $A_0 A_1 \cdots A_n$ (см. задачу 1.22). При любой расстановке скобок найдем последнюю операцию, определяемую это парой, и разобьем произведение на две части:

$$(A_0 A_1 \cdots A_k)(A_{k+1} A_{k+2} \cdots A_n)$$

— соответственно первый и второй множители.

Тогда задача сведется к аналогичной задаче меньшей «размерности» — расставить скобки в каждом из множителей, а затем поставить завершающую — внешнюю — пару скобок. Поскольку в первом множителе скобки можно расставить k способами, а во втором — $(n - k)$ способами, то, комбинируя различные расстановки в них, получим по правилу умножения $k(n - k)$ способов.

Далее, так как граница разбиения на два множителя может оказаться в любом месте между первым и $(n + 1)$ -м множителем, получаем n различных вариантов, которые надо просуммировать. Итак, получаем формулу

$$u_n = \sum_{k=0}^n u_k u_{n-k}, \quad \text{где } u_0 = 0, \quad u_1 = 1.$$

Определим производящую функцию для чисел Каталана согласно определению 1.114, бесконечный степенной ряд с коэффициентами — числами Каталана:

$$u(z) = \sum_{k=0}^{\infty} u_k z^k = u_0 + u_1 z + \dots + u_n z^n + \dots = z + z^2 + 2z^3 + \dots + u_n z^n + \dots$$

Найдем квадрат функции $u(x)$:

$$\begin{aligned} u(z)^2 &= (u_0 + u_1 z + \dots + u_n z^n + \dots) (u_0 + u_1 z + \dots + u_n z^n + \dots) = \\ &= u_0^2 + (u_0 u_1 + u_1 u_0) z + (u_0 u_2 + u_1 u_1 + u_2 u_0) z^2 + \dots + \\ &+ \sum_{k=0}^n u_k u_{n-k} z^n + \dots = u(z) - z. \end{aligned}$$

Решив полученное уравнение как квадратное относительно $u(z)$, получим

$$u(z)^2 - u(z) + z = 0, \text{ откуда } u(z)_{1,2} = \frac{1 \pm \sqrt{1-4z}}{2}. \quad (1.128)$$

Так как $u(0) = 0$, в полученной формуле (1.128) надо выбрать знак «минус».

Разложим в степенной ряд $\sqrt{1-4z}$:

$$\begin{aligned} \sqrt{1-4z} &= 1 + \sum_{n=1}^{\infty} (-4)^n \frac{\frac{1}{2} \left(\frac{1}{2} - 1\right) \left(\frac{1}{2} - 2\right) \dots \left(\frac{1}{2} - n + 1\right)}{n!} z^n = \\ &= 1 + \sum_{n=1}^{\infty} (-1)^n 4^n \frac{\frac{1}{2^n} \cdot 1 \cdot (1-2)(1-4) \dots (1-2n+2)}{n!} z^n = \\ &= 1 + \sum_{n=1}^{\infty} (-1)^n 2^n \frac{(-1)^{n-1} \cdot 1 \cdot 1 \cdot 3 \dots (2n-3)}{n!} z^n = \\ &= 1 - \sum_{n=1}^{\infty} 2^n \frac{1 \cdot 2 \cdot 3 \cdot 4 \dots (2n-3)(2n-2)}{n! \cdot 2 \cdot 4 \dots (2n-2)} z^n = \\ &= 1 - \sum_{n=1}^{\infty} 2^n \frac{(2n-2)!}{2^{n-1} (n-1)! n!} z^n = 1 - 2 \sum_{n=1}^{\infty} \frac{(2n-2)!}{(n-1)! n!} z^n. \end{aligned}$$

Подставив полученный ряд в выражение (1.128) для $u(z)$, получим

$$u_n = \frac{(2n-2)!}{(n-1)! n!} = \frac{1}{n} \frac{(2n-2)!}{(n-1)!(n-1)!} = \frac{1}{n} C_{2n-2}^{n-1}.$$

1.7.5. Число многочленов заданной степени, неприводимых над полем вычетов

Использование производящих функций позволяет решать и более сложные комбинаторные задачи. Одной из них является нахождение числа неприводимых многочленов над полем вычетов по модулю p .

В дальнейшем будем рассматривать только нормализованные многочлены, первый коэффициент которых равен 1, что не умаляет общности рассмотрения.

Всего многочленов степени n над полем \mathbb{Z}_p имеется p^n , так как многочлен n -й степени имеет $n + 1$ коэффициент, первый из которых равен 1, а остальные n могут принимать по p значений каждый. Тогда производящая функция для чисел коэффициентов задается следующим рядом:

$$1 + px + p^2x^2 + \dots + \sum_{n=0}^{\infty} p^n x^n = \frac{1}{1 - px} \quad (1.129)$$

по формуле суммы бесконечно убывающей прогрессии.

Вычислим теперь эту функцию другим способом. Обозначим N_d — число различных неприводимых многочленов степени d . Любой многочлен над полем \mathbb{Z}_p может быть единственным образом записан в виде произведения степеней неприводимых многочленов:

$$f(x) = (f_1(x))^{i_1} (f_2(x))^{i_2} \dots$$

Если обозначить через d_k степени многочленов $f_k(x)$, а степень многочлена $f(x)$ через n , то получим равенство $n = i_1 d_1 + i_2 d_2 + \dots$

Заметим, что такой коэффициент у x_n можно получить при перемножении бесконечного числа следующих рядов:

$$(1 + x^{d_1} + x^{2d_1}x^{3d_1} + \dots) \cdot (1 + x^{d_2} + x^{2d_2}x^{3d_2} + \dots) \dots = \frac{1}{1 - x^{d_1}} \cdot \frac{1}{1 - x^{d_2}} \dots$$

Соберем в этом бесконечном произведении все одинаковые множители: для d_i число таких множителей обозначено ранее как N_{d_i} . Получим другое выражение для производящей функции чисел всех многочленов над полем вычетов по модулю p .

$$\prod_{d=1}^{\infty} \left(\frac{1}{1 - x^d} \right)^{N_d} \quad (1.130)$$

Приравняем правую часть (1.129) и (1.130) и прологарифмируем полу-

ченное равенство:

$$\ln \left(\frac{1}{1 - px} \right) = \sum_{d=1}^{\infty} N_d \ln \left(\frac{1}{1 - x^d} \right).$$

Используя формулу для разложения натурального логарифма в степенной ряд

$$\ln(1 + x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \dots \quad \text{или} \quad \ln \left(\frac{1}{1 - x} \right) = x + \frac{1}{2}x^2 + \frac{1}{3}x^3 + \dots,$$

получаем равенство

$$\sum_{n=1}^{\infty} \frac{(px)^n}{n} = \sum_{d=1}^{\infty} N_d \sum_{j=1}^{\infty} \frac{x^{jd}}{j}.$$

Приравняв в последнем равенстве коэффициенты при x_n , имеем

$$\frac{p^n}{n} = \sum_{d|n} N_d \frac{d}{n}, \quad (1.131)$$

так как в (1.131) справа степени x_n получаются, если j удовлетворяет условию $jd = n$, или $j = n/d$, это означает, что для всех делителей d числа n в сумме справа найдутся слагаемые n -й степени.

Преобразовав выражение (1.131), получим

$$p^n = \sum_{d|n} dN_d. \quad (1.132)$$

К равенству (1.132) можно применить формулу обращения Мебиуса (см. теорему 1.81), если заметить, что его можно представить в виде

$$g(n) = \sum_{d|n} f(d), \quad \text{где} \quad g(n) = p^n, \quad f(d) = dN_d.$$

Тогда

$$f(n) = nN_n = \sum_{d|n} \mu(d)g \left(\frac{n}{d} \right) = \sum_{d|n} \mu(d)p^{\frac{n}{d}}, \quad \text{или} \quad N_n = \frac{1}{n} \sum_{d|n} \mu(d)p^{\frac{n}{d}}. \quad (1.133)$$

Пример 1.143. Найдем число неприводимых многочленов шестой степени над полем вычетов по модулю 2. По формуле (1.133) имеем

$$\begin{aligned} N_6 &= \frac{1}{6} \sum_{d|6} \mu(d) p^{\frac{6}{d}} = \frac{1}{6} (\mu(1)2^6 + \mu(2)2^3 + \mu(3)2^2 + \mu(6)2^1) = \\ &= \frac{1}{6} (2^6 - 2^3 - 2^2 + 2^1) = \frac{54}{6} = 9. \end{aligned}$$

Пример 1.144. Найдем все многочлены из примера 1.143.

Заметим, что неприводимый многочлен обязательно имеет ненулевой свободный член (иначе бы выносился множитель x) и нечетное число слагаемых (иначе значение многочлена при $x = 1$ было бы равно 0 и по теореме Безу выносился множитель $x + 1$). Таким образом, из всех $2^6 = 64$ многочленов остается перебрать лишь

$$C_5^1 + C_5^3 + C_5^5 = 5 + 10 + 1 = 16.$$

Ниже приведены результаты вычислений с помощью среды Maple. Многочлены, стоящие на 1, 3, 5, 7, 8, 9, 11, 13, 14-м местах, неприводимые.

```
> Factors(x^6+x^5+1) mod 2;
```

```
[1, [[x^6 + x^5 + 1, 1]]]
```

```
> Factors(x^6+x^4+1) mod 2
```

```
[1, [[x^3 + x^2 + 1, 2]]]
```

```
> Factors(x^6+x^3+1) mod 2
```

```
[1, [[x^6 + x^3 + 1, 1]]]
```

```
> Factors(x^6+x^2+1) mod 2
```

```
[1, [[x^3 + x + 1, 2]]]
```

```
> Factors(x^6+x+1) mod 2
```

```
[1, [[x^6 + x + 1, 1]]]
```

```
> Factors(x^6+x^5+x^4+x^3+1) mod 2
```

```
[1, [[x^4 + x + 1, 1], [x^2 + x + 1, 1]]]
```

```
> Factors(x^6+x^5+x^4+x^2+1) mod 2
```

```
[1, [[x^6 + x^5 + x^4 + x^2 + 1, 1]]]
```

```
> Factors(x^6+x^5+x^4+x+1) mod 2
```

```
[1, [[x^6 + x^5 + x^4 + x + 1, 1]]]
```

```
> Factors(x^6+x^5+x^3+x^2+1) mod 2
```

$$[1, [[x^6 + x^5 + x^3 + x^2 + 1, 1]]]$$

$$> \text{Factors}(x^6+x^5+x^3+x+1) \text{ mod } 2$$

$$[1, [[x^2 + x + 1, 3]]]$$

$$> \text{Factors}(x^6+x^5+x^2+x+1) \text{ mod } 2$$

$$[1, [[x^6 + x^5 + x^2 + x + 1, 1]]]$$

$$> \text{Factors}(x^6+x^4+x^3+x^2+1) \text{ mod } 2$$

$$[1, [[x^4 + x^3 + x^2 + x + 1, 1], [x^2 + x + 1, 1]]]$$

$$> \text{Factors}(x^6+x^4+x^3+x+1) \text{ mod } 2$$

$$[1, [[x^6 + x^4 + x^3 + x + 1, 1]]]$$

$$> \text{Factors}(x^6+x^4+x^2+x+1) \text{ mod } 2$$

$$[1, [[x^6 + x^4 + x^2 + x + 1, 1]]]$$

$$> \text{Factors}(x^6+x^3+x^2+x+1) \text{ mod } 2$$

$$[1, [[x^4 + x^3 + x + 1, 1], [x^2 + x + 1, 1]]]$$

$$> \text{Factors}(x^6+x^5+x^4+x^3+x^2+x+1) \text{ mod } 2$$

$$[1, [[x^3 + x^2 + 1, 1], [x^3 + x + 1, 1]]]$$

Задачи и вопросы

1. Найти решение однородного рекуррентного уравнения с граничными условиями:

$$\text{а) } 3f_{n+2} - 8f_{n+1} + 4f_n = 0, \quad f_0 = 2, f_1 = 2,$$

$$\text{б) } 4f_{n+2} - 8f_{n+1} + 3f_n = 0, \quad f_0 = 0, f_1 = -1,$$

$$\text{в) } 9f_{n+2} - 12f_{n+1} + 4f_n = 0, \quad f_0 = 1, f_1 = -2,$$

$$\text{г) } 9f_{n+2} + 6f_{n+1} + f_n = 0, \quad f_0 = -1, f_1 = -1.$$

2. Найти общее решение неоднородного рекуррентного уравнения:

$$\text{а) } f_{n+2} - 4f_{n+1} + 4f_n = 2n^2 - 1,$$

$$\text{б) } f_{n+2} - f_{n+1} - 2f_n = n^2 + 3n - 3,$$

$$\text{в) } f_{n+2} - 3f_{n+1} + 2f_n = 2n + 3,$$

$$\text{г) } f_{n+2} - f_n = 3n - 2.$$

1.8. Элементы дискретной теории вероятностей

Данный раздел является естественным продолжением раздела по комбинаторике и частично написан по материалам [50].

1.8.1. Основные определения

Будем рассматривать конечное множество *элементарных событий* S , все его элементы считаются равноправными, *равновероятными*. Любое событие A считается подмножеством S и состоит из элементарных событий.

Введем некоторые естественные требования, которым должно будет удовлетворять определение вероятности.

1. При сопоставлении событий вероятность события A , содержащегося в другом событии B , должна иметь вероятность, не превосходящую вероятности B (события — это множества, и они могут содержаться одно в другом).

2. Если событие A составляется из событий, которые не могут выполняться одновременно (такие события называются *несовместными*), то вероятность осуществления события A должна равняться сумме вероятностей составляющих событий. Иначе говоря, если задано разбиение множества исходов A на подмножества A_i , то вероятность события A равна сумме вероятностей событий A_i .

Определение 1.66. Отношение $|A|/|S|$ будем называть *вероятностью* события A и обозначать через $P(A)$.

Событие, вероятность которого равна 1, называется *достоверным* — оно наступает всегда.

Событие, вероятность которого равна 0, называется *невозможным* — оно не наступает никогда.

Замечание 1.68

Понятие вероятности, введенное нами как отношение числа входящих в событие (благоприятных) равновозможных исходов к числу всех исходов, удовлетворяет всем ранее сформулированным условиям.

Пример 1.145. При бросании двух игральных костей вероятность получить не менее 11 очков равна $\frac{3}{36}$, так как всего имеется 36 равновозможных исходов и 3 из них благоприятны — $(5,6), (6,5), (6,6)$.

Было бы неверным считать, что так как может выпасть от 2 до 12 очков и благоприятными являются 2 исхода из этих 11, то искомая вероятность равна $\frac{2}{11}$. Исходы типа «числа выпавших очков» не являются равновозможными. Например выпадению 12 очков соответствует всего одна комбинация, а выпадению 7 очков — целых 6 комбинаций.

Упражнение 1.22. Найдите вероятность того, что на первой кости выпадет очков больше, чем на второй. Ответ: $\frac{15}{36}$.

Определение 1.67. Событие, которое составлено из элементарных событий, входящих одновременно в события A и B , называется *совмещением этих событий*. Таким образом, множество элементарных событий, входящих в совмещение, — это пересечение соответствующих множеств. Очевидно, что вероятность совмещения событий не превосходит вероятности каждого из этих событий.

Оба этих определения легко переносятся на несколько событий.

Определение 1.68. Разбиение множества элементарных событий S на несовместные события S_1, S_2, \dots, S_m называется *полной системой событий*. Каждое событие A можно представить, как объединение несовместных событий — совмещение A с элементами разбиения S_i .

Очевидно, что

$$P(A) = \sum_{i=1}^m P(A \cap S_i). \quad (1.134)$$

Формула (1.134) называется *формулой полной вероятности*.

Введем понятие независимых событий.

Определение 1.69. События A и B называются *независимыми*, если вероятность их совмещения равна произведению их вероятностей:

$$P(A \cap B) = P(A) \times P(B).$$

В случае нескольких событий A_1, A_2, \dots, A_k следует различать попарную независимость этих событий и их *независимость в совокупности*. В первом случае

$$P(A_i \cap A_j) = P(A_i) \times P(A_j) \text{ при } i \neq j,$$

во втором случае (более жесткое требование)

$$P(A_1 \cap A_2 \cap \dots \cap A_k) = P(A_1) \times P(A_2) \times \dots \times P(A_k).$$

Легко показать, что независимость событий в совокупности не следует из их попарной независимости.

Пример 1.146. Рассмотрим игральную кость в виде правильного тетраэдра¹. Одна грань этого тетраэдра выкрашена в белый цвет W , вторая — в черный B , третья — в красный R , а окраска четвертой грани — смешанная M , в ней есть все три цвета.

При каждом бросании кость равновероятно ложится какой-то стороной вниз. Вероятность того, что на этой грани окажется белый, красный или черный цвет, равна, очевидно, $1/2$, так как для этого события благоприятны два исхода из четырех возможных. Вероятность выпадения двух цветов сразу, например красного и черного, равна $1/4$ и не зависит от набора цветов. Таким образом для событий выпал красный цвет и выпал черный цвет имеем:

$$P(R \cap B) = P(R) \times P(B),$$

и по определению эти два события независимы. С другой стороны, вероятность выпадения трех цветов

$$P(R \cap B \cap W) = \frac{1}{4} \neq P(R) \times P(B) \times P(W) = \frac{1}{8},$$

и независимости в совокупности нет.

1.8.2. Условные вероятности и формула Байеса

При работе с зависимыми случайными событиями необходимо учитывать их взаимное влияние на вероятностное поведение. Это влияние проявляется в том, что осуществление одного события, может изменить вероятности других событий и приходится говорить о вероятностях событий в тех или иных условиях. Введем следующее определение.

Определение 1.70. Пусть A — некоторое событие, имеющее положительную вероятность (то есть не невозможное событие). Определим для события B *условную вероятность B при условии A* как отношение

$$P(B|A) = \frac{P(B \cap A)}{P(A)}. \quad (1.135)$$

¹ Пример предложен С. Н. Бернштейном.

 **Замечание 1.69**

Очевидно, что таким образом определенные условные вероятности обладают всеми свойствами обычных вероятностей: они меняются от 0 до 1, возрастают с расширением события, условная вероятность объединения несовместных событий равна сумме их условных вероятностей.

Используя понятие условной вероятности, можно по-другому определить понятие независимости событий (определение 1.69): события A и B независимы, если

$$P(B|A) = P(B). \quad (1.136)$$

Пример 1.147. Необходимо определить, повлиял ли просмотр рекламы на сбыт продукции. Для этого был проведен опрос, результаты которого приведены в таблице 1.52. Событие A — респондент сказал, что купит продукт, событие C — что не купит, событие B — респондент смотрел рекламу, D — не смотрел.

Таблица 1.52.

	A	C
B	459	441
D	51	49

Критерием независимости события является выполнение условия (1.136). Нужно проверить, будет ли вероятность покупки $p(A)$ совпадать с вероятностью покупки при условии, что респондент смотрел рекламу $p(A|B)$. Сосчитаем обе вероятности исходя из определения вероятности:

$$p(A) = \frac{459 + 51}{459 + 51 + 441 + 49} = 0.51; \quad p(A|B) = \frac{459}{459 + 441} = 0.51.$$

Таким образом, определили, что события A — покупка товара и B — просмотр рекламы независимы.

Пример 1.148. Из 9 вопросов экзамена студент выучил 6. Какова вероятность сдать экзамен, если для этого необходимо ответить на два случайно выбранных вопроса.

Определим события A_1 — студент ответил на первый вопрос и A_2 — студент ответил на второй вопрос, тогда $A = A_1 \cap A_2$ и

$$p(A) = p(A_1 \cap A_2) = p(A_1)p(A_2|A_1) = \frac{6}{9} \cdot \frac{5}{8} = \frac{5}{12}.$$

В терминах условной вероятности можно по-другому записать формулу полной вероятности (1.134):

$$P(A) = \sum_{i=1}^m P(A|S_i)P(S_i). \quad (1.137)$$

Пример 1.149. В урне было 10 белых и 10 черных шаров. С вероятностью $1/4$ из урны удалили 5 черных шаров. Какова вероятность того, что вынутый наугад из этой урны шар окажется белым.

Мы не знаем, удалены ли шары из урны или нет, и должны делать гипотезы и рассматривать условные вероятности интересующего нас события.

При условии A_0 (шары не были удалены) вероятность вынуть белый шар равна $P(B|A_0) = 10/20 = 1/2$. При условии A_1 (шары были удалены) вероятность равна $P(B|A_1) = 10/15 = 2/3$. Так как по условию задачи $P(A_1) = 1/4$ и, следовательно, $P(A_0) = 3/4$, то согласно (1.137):

$$P(B) = P(B|A_0)P(A_0) + P(B|A_1)P(A_1) = \frac{1}{2} \cdot \frac{3}{4} + \frac{2}{3} \cdot \frac{1}{4} = \frac{3}{8} + \frac{1}{6} = \frac{13}{24}.$$

Пример 1.150. В магазин привезли 50 смартфонов, изготовленных тремя фирмами в количестве 15, 25 и 10 штук. Качество сборки первой фирмы — 0.8, второй и третьей соответственно 0.65 и 0.9. Найти вероятность того, что купленный смартфон окажется отличного качества.

Введем гипотезы: H_1 — смартфон изготовлен первой фирмой, H_2 — второй и H_3 — третьей. Согласно определению 1.66 вероятности гипотез:

$$P(H_1) = \frac{15}{50} = 0.3, \quad P(H_2) = \frac{25}{50} = 0.5, \quad P(H_3) = \frac{10}{50} = 0.2.$$

Введем событие A — купленный смартфон отличного качества. Согласно условиям:

$$P(A|H_1) = 0.8, \quad P(A|H_2) = 0.65, \quad P(A|H_3) = 0.9.$$

По формуле полной вероятности имеем:

$$\begin{aligned} P(A) &= P(A|H_1)P(H_1) + P(A|H_2)P(H_2) + P(A|H_3)P(H_3) = \\ &= 0.8 \times 0.3 + 0.65 \times 0.5 + 0.9 \times 0.2 = 0.745. \end{aligned}$$

Из формулы полной вероятности легко получить одну очень важную формулу, известную как *формула Байеса*. Выразим совмещение событий $P(B \cap A_i)$ с помощью условных вероятностей. Согласно (1.135) имеем

$$P(A_i|B)P(B) = P(B|A_i)P(A_i).$$

Разделив обе части формулы на $P(B)$ и подставив для $P(B)$ ее выражение из формулы полной вероятности, получим формулу Байеса

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^m P(B|A_j)P(A_j)}. \quad (1.138)$$

Историческая справка

Томас Байес (1702–1761) — английский математик, священник. При жизни опубликовал всего две работы, одна из них богословская, другая математическая. Формула Байеса стала известной уже после его смерти.

Эта формула имеет следующее истолкование и применение.

Предположим, что случайное событие B зависит от некоторого случайного параметра, который определяется заданием полной системой событий $\{A_i\}$. Известны некоторые (*априорные* — заданные до опыта) вероятности событий A_i . Нас интересует, какое из событий осуществилось.

Мы проводим эксперимент и узнаем, что осуществилось событие B . Информация об этом дает возможность пересчитать вероятности A_i — получить *апостериорные* вероятности.

Пример 1.151. Зная вероятности появления различных болезней и вероятности появления отдельных симптомов при той или иной болезни, врач может делать различные анализы, которые постепенно изменяют вероятности наличия этих болезней у пациента. Пусть при заболевании гриппом с вероятностью 0.6 можно заболеть гриппом $A(GA)$, а с вероятностью 0.4 — гриппом $B(GB)$. При этом реакция иммунофлуоресценции (RI) положительна с вероятностью 0.85 в случае гриппа A и с вероятностью 0.35 в случае гриппа B . Требуется оценить вероятность того, что больной гриппом болен гриппом A , если у него реакция положительная.

Подставляя числа в формулу Байеса, получаем

$$P(A|RI^+) = \frac{0.85 \times 0.6}{0.85 \times 0.6 + 0.35 \times 0.4} = \frac{0.51}{0.51 + 0.14} \approx 0.7846,$$

и диагноз склоняется в сторону гриппа A .

Пример 1.152. Имеются две колоды карт по 52 (обозначим ее K_{52}) и по 36 (обозначим ее K_{36}) листов. Выбирается одна из колод (с равной вероятностью) и из нее вытаскивается случайно одна карта. Меняет ли информация об этой карте наши представления о выбранной колоде?

Да! Если выбрана карта, которой нет в K_{36} , то выбрана K_{52} . Это очевидно.

А если выбрана карта, которая есть в K_{36} ? Вероятности по-прежнему равны, или что-то изменилось? Давайте обсудим это.

Представьте себе, что мы много раз проводили один и тот же эксперимент — вытаскивали карту из колоды (а потом возвращали). Мы сделали это 1000 раз, и каждый раз получали карту из K_{36} . Повлиял ли этот длительный эксперимент на наши предположения? Скорее всего, мы ответим да. Не может быть (нет, маловероятно), чтобы при тысяче экспериментов ни разу не появилась карта из продолжения колоды.

А если мы сделаем только 100 попыток? Или 50? Неужели есть такой порог, при котором наши представления изменяются? Конечно же, нет. Просто при каждой попытке вероятность изменяется очень мало. Давайте посчитаем. Формула Байеса и дает нам такую возможность.

Введем следующие события:

- B_{36} — выбрана колода K_{36} ;
- B_{52} — выбрана колода K_{52} ;
- A — из колоды вынута карта $8\clubsuit$.

Согласно (1.138) имеем

$$P(B_{36}|A) = \frac{P(A|B_{36}) \times P(B_{36})}{P(A|B_{36}) \times P(B_{36}) + P(A|B_{52}) \times P(B_{52})} \quad (1.139)$$

Легко показать, что

$$P(B_{36}) = 1/2, \quad P(B_{52}) = 1/2, \quad P(A|B_{36}) = 1/36, \quad P(A|B_{52}) = 1/52.$$

Подставляя в (1.139), получаем

$$P(B_{36}|A) = \frac{1/2 \times 1/36}{1/2 \times 1/36 + 1/2 \times 1/52} = \frac{52}{88} = \frac{13}{22}.$$

Пример 1.153. Экзамен по дискретной математике принимают 3 злых и 9 добрых гномов. Злые гномы поставили 40% двоек и 60% пятерок, добрые — 10% двоек и 90% пятерок. Какова вероятность того, что студент:

- 1) попал на экзамен к доброму гному;
- 2) сдал экзамен;
- 3) сдавший экзамен, сдавал его доброму гному?

Введем следующие гипотезы: H_1 — студент сдавал экзамен злому гному, H_2 — студент сдавал экзамен доброму. Согласно классическому опре-

делению вероятности:

$$p(H_1) = \frac{3}{12} = \frac{1}{4}; \quad p(H_2) = \frac{9}{12} = \frac{3}{4}.$$

Вероятность события A — студент сдал экзамен — зависит от того, кому студент сдавал, т. е. от гипотез H_1 и H_2 . По формуле полной вероятности:

$$p(A) = p(H_1)p(A|H_1) + p(H_2)p(A|H_2) = \frac{1}{4} \cdot 0.6 + \frac{3}{4} \cdot 0.9 = \frac{33}{40} = 0.825.$$

Вероятность того, что студент сдавал экзамен доброму гному, если известно, что экзамен сдан (вероятность гипотезы после испытания), вычисляется по формуле Байеса:

$$p(H_2|A) = \frac{p(A|H_2)p(H_2)}{p(A)} = \frac{0.9 \cdot 0.75}{0.825} = 0.8182.$$

1.8.3. Дискретные случайные величины.

Математическое ожидание и дисперсия

Определение 1.71. Пусть $\Omega = \{\omega\}$ — множество элементарных событий. Заданная на Ω функция ξ

$$\xi : \Omega \rightarrow X = (x_1, \dots, x_n \mid x_i \in R, 1 \leq i \leq n)$$

называется *дискретной случайной величиной*. Обозначается как $\xi = \xi(\omega)$.

Пусть вероятности значений x_1, \dots, x_n случайной величины ξ равны p_1, p_2, \dots, p_n , то есть $P(\xi = x_i) = p_i$. Заметим, что события

$$P(\xi = x_1) = p_1, \dots, P(\xi = x_n) = p_n$$

образуют полную систему событий, поэтому

$$p_1 + \dots + p_n = 1.$$

Такое соответствие, связывающее значения случайной величины с соответствующими им вероятностями, называют *законом распределения дискретной случайной величины*. Часто ее задают табличным способом (таблица 1.53).

Определение 1.72. Случайные величины ξ и η называются *независимыми*, если для любых $a, b \in R$ независимы события $\xi = a$ и $\eta = b$, то есть если

$$P(\xi = a, \eta = b) = P(\xi = a) \times P(\eta = b).$$

Таблица 1.53.

ξ	x_1	\dots	x_n
p	p_1	\dots	p_n

Определение 1.73. Математическим ожиданием случайной величины ξ называется величина:

$$E\xi = x_1p_1 + x_2p_2 + \dots + x_np_n = \sum_{k=1}^n x_kp_k. \quad (1.140)$$

 **Замечание 1.70**

Математическое ожидание похоже на центр тяжести системы материальных точек — если считать вероятности значений случайной величины ξ массами точек, то получается в точности формула (1.140).

Теорема 1.87. Справедливы следующие свойства математического ожидания:

- 1) если $P(\xi = a) = 1$, то $E\xi = a$;
- 2) если ξ и η — две случайные величины, то $E(\xi + \eta) = E\xi + E\eta$;
- 3) если $\eta = b\xi$, где $b \in R$, то $E\eta = bE\xi$.

Доказательство. Все три свойства очевидным образом следуют из определения. ■

Пример 1.154. Найти математическое ожидание дискретной случайной величины ξ , зная закон ее распределения (таблица 1.54).

Таблица 1.54.

ξ	5	4	3
p	0.2	0.5	0.3

Согласно (1.140) имеем

$$E\xi = 5 \times 0.2 + 4 \times 0.5 + 3 \times 0.3 = 3.3.$$

На практике часто приходится оценить рассеяние возможных значений случайной величины вокруг ее среднего значения.

Определение 1.74. Математическое ожидание квадрата отклонения случайной величины от ее математического ожидания называется *дисперсией* этой случайной величины:

$$D\xi = E(\xi - E\xi)^2. \quad (1.141)$$

 **Замечание 1.71**

Дисперсия характеризует разброс случайной величины вокруг ее математического ожидания. Продолжая аналогии с механикой, можно заключить, что дисперсия — это момент инерции системы материальных точек.

Если в (1.141) раскрыть скобки, то получится еще одно выражение для дисперсии:

$$D\xi = E\xi^2 - (E\xi)^2.$$

Теорема 1.88. Справедливы следующие свойства дисперсии:

- 1) если $P(\xi = a) = 1$, то $D\xi = 0$;
- 2) если $\eta = \xi + a$, где $a \in R$, то $D\xi = D\eta$;
- 3) если $\eta = b\xi$, где $b \in R$, то $D\eta = b^2D\xi$;
- 4) если η и ξ — две независимые случайные величины, то

$$D(\xi + \eta) = D\xi + D\eta.$$

Доказательство. Первые три свойства очевидны. Докажем четвертое. По определению дисперсии

$$\begin{aligned} D(\xi + \eta) &= E(\xi + \eta - E(\xi + \eta))^2 = E(\xi - E\xi + \eta - E\eta)^2 = \\ &= E(\xi - E\xi)^2 + 2E((\xi - E\xi)(\eta - E\eta)) + E(\eta - E\eta)^2 = \\ &= D\xi + D\eta + 2(E\xi\eta - E\xi E\eta). \end{aligned}$$

Осталось показать, что $E\xi\eta - E\xi E\eta = 0$. Действительно

$$\begin{aligned} E\xi\eta &= \sum_{a,b} abP(\xi = a, \eta = b) = \sum_{a,b} abP(\xi = a)P(\eta = b) = \\ &= \sum_a aP(\xi = a) \sum_b bP(\eta = b) = E\xi E\eta. \end{aligned}$$

■

Пример 1.155. Найти дисперсию случайной величины ξ , которая задана следующим законом распределения (таблица 1.55).

Используя (1.140), имеем

$$E\xi = 1 \times 0.3 + 2 \times 0.5 + 5 \times 0.2 = 2.3.$$

Таблица 1.55.

ξ	1	2	5
p	0.3	0.5	0.2

Далее

$$\begin{cases} (\xi_1 - E\xi)^2 = (1 - 2.3)^2 = 1.69, \\ (\xi_2 - E\xi)^2 = (2 - 2.3)^2 = 0.09, \\ (\xi_3 - E\xi)^2 = (5 - 2.3)^2 = 7.29. \end{cases}$$

Тогда согласно (1.141):

$$D\xi = 1.69 \times 0.3 + 0.09 \times 0.5 + 7.29 \times 0.2 = 2.01.$$

Пример 1.156. В урне 4 черных и 5 белых шаров. Наудачу извлекают два шара. Случайная величина ξ — количество черных шаров среди извлеченных. Найти математическое ожидание и дисперсию случайной величины.

Возможны три элементарных события, соответствующие извлечению двух шаров из урны: ω_0 — все извлеченные шары белые, ω_1 — извлечены белый и черный шары, ω_2 — извлечены два черных шара. Таким образом, 0, 1, 2 — возможные значения дискретной случайной величины ξ .

Найдем вероятности, с которыми случайная величина ξ принимает эти значения. Общее число исходов $C_9^2 = 36$. Тогда:

$$\begin{aligned} P(\xi = 0) &= P(\omega_0) = p_1 = \frac{C_5^2}{36} = \frac{10}{36}; \\ P(\xi = 1) &= P(\omega_1) = p_2 = \frac{C_4^1 C_5^1}{36} = \frac{20}{36}; \\ P(\xi = 2) &= P(\omega_2) = p_3 = \frac{C_4^2}{36} = \frac{6}{36}. \end{aligned}$$

Согласно (1.140) и (1.141) имеем:

$$\begin{aligned} E\xi &= 0 \times \frac{10}{36} + 1 \times \frac{20}{36} + 2 \times \frac{6}{36} = \frac{8}{9} \approx 0.89, \\ D\xi &= \left(0 - \frac{8}{9}\right)^2 \times \frac{10}{36} + \left(1 - \frac{8}{9}\right)^2 \times \frac{20}{36} + \left(2 - \frac{8}{9}\right)^2 \times \frac{6}{36} \approx 0.4321. \end{aligned}$$

Задачи и вопросы

1. Даны числа $\{1, \dots, 20\}$. Из них случайным образом выбирается одно. Какова вероятность того, что оно делится на 3. Из того же множества случайным образом выбрано 5 различных чисел. Какова вероятность, что 2 из них делятся на 3?

2. Имеется 100 билетов, помеченных одной из четырех меток: (0), (1), (2), (1, 2). Билетов, содержащих метку (1), 38 штук. Билетов, содержащих метку (2), 20 штук. Билетов, содержащих метку (1, 2), 5 штук.

а) Вычислите вероятности событий A — «метка билета содержит цифру 1» и B — «метка билета содержит цифру 2».

б) Являются ли события A и B независимыми?

3. Из урны, в которой имеется 6 зеленых шаров и 7 синих, наудачу выбирают два. Какова вероятность того, что среди них хоть один синий?

4. Бросаются независимо два игральных кубика. Какова вероятность того, что на одном выпадет число меньше 3 на другом — больше 4?

5. Два стрелка независимо друг от друга стреляют по одной мишени, делая по одному выстрелу. Вероятность попадания в мишень для первого стрелка составляет 0.75, а для второго — 0.50. Найти вероятность, что оба стрелка попадут в мишень.

6. Одинаковые шурупы выпускаются двумя заводами. Первый дает 5% брака, второй — 3% брака. Смешали 70 шурупов первого завода и 30 шурупов второго. Случайно выбранный шуруп оказался дефектным. Какова вероятность того, что он сделан на первом заводе?

7. Два стрелка независимо друг от друга стреляют по одной мишени, делая по одному выстрелу. Вероятность попадания в мишень для первого стрелка составляет 0.65, а для второго — 0.75. После стрельбы обнаружено одно попадание. Найти вероятность, что в мишень попал первый стрелок.

8. В первой урне имеется 10 шаров, из которых 4 красных, во второй — 20, из которых 3 красных. Из каждой наудачу выбирают по шару, затем из этих двух шаров наудачу выбирают один. Какова вероятность того, что он красный?

9. Найдите распределение случайной величины ξ , если она может принимать только одно из двух значений $x_1 < x_2$. Известно, что

$$P(\xi = x_1) = 0.2, E\xi = 1.5, D\xi = 0.1.$$

ГРАФЫ И БИНАРНЫЕ ОТНОШЕНИЯ

2.1. Основные понятия теории графов

2.1.1. Простейшие определения и свойства

Определение 2.1. *Неориентированным графом*, или просто *графом* $G = G(V, E)$, называют пару множеств: $V = \{v_1, \dots, v_p\}$ — непустое, конечное множество *вершин* и $E = \{(v_i, v_j) \mid v_i, v_j \in V\}$ — множество неупорядоченных пар вершин, называемых *ребрами*.

Вершины и ребра графа называют его *элементами*. Число вершин графа G называют его *порядком* и обозначают $|G|$. Если $|G| = n$, а $|E| = m$, то граф называют (n, m) -*графом*.

Ребро (v, v) называют *петлей*.

Замечание 2.1

Иногда рассматривают более общее понятие графа, полагая что множество ребер E есть семейство подмножеств множества $V \times V$, т. е. допускается существование кратных ребер в графе. Такой граф с кратными ребрами и петлями называют *псевдографом*. Псевдограф без петель называют *графом с кратными ребрами* или *мультиграфом*. Примерами мультиграфов могут служить структурные формулы в органической химии. Ребра, соединяющие одну и ту же пару вершин, называют *параллельными*. Мультиграф, у которого максимальное число параллельных дуг равно s , называют s -*графом*.

Граф по определению 2.1, в котором любые две его вершины соединены не более чем

одним ребром (т. е. 1-граф), иногда называют униграфом.

Далее в качестве графов будем рассматривать униграфы без петель, хотя некоторые результаты этой главы справедливы и для мультиграфов.

Определение 2.2. Рассмотрим ребро $e = (v, u) \in \mathbf{E}$. Тогда вершины v , u называют концами или концевыми точками ребра e . При этом говорят, что ребро e инцидентно вершинам v , u . Вершины v , u называют смежными.

Множество вершин, смежных с вершиной v , называют множеством смежности вершины v и обозначают $\gamma^+(v)$.

Граф \mathbf{G} называют полным, если любые две его вершины смежны. Полный граф с n вершинами обозначается K_n .

Если в графе $|\mathbf{E}| = 0$, такой граф называют нуль-графом. Далее в учебнике будем обозначать нуль-граф с n вершинами Z_n .

Звездой вершины u называют подмножество ребер

$$\{(v_i, v_j) \in \mathbf{E} \mid v_i = u \text{ или } v_j = u\} = \delta(u).$$

Число $|\delta(u)| = \deg(u)$ называют степенью (или валентностью) вершины u . При $\deg(u) = 0$ вершину называют изолированной, при $\deg(u) = 1$ — концевой, или висячей.

Если степени всех вершин графа равны k , то граф называется регулярным, или однородным, степени k . Регулярные графы третьей степени часто называют кубическими.

Максимальная степень вершины графа \mathbf{G} обозначается $\Delta(\mathbf{G})$.

Теорема 2.1 (Эйлер). Для произвольного графа $\mathbf{G}(\mathbf{V}, \mathbf{E})$ справедливо равенство

$$|\mathbf{E}| = \frac{1}{2} \sum_{v \in \mathbf{V}} \deg(v).$$

Доказательство. В графе \mathbf{G} нет петель и каждое ребро учитывается дважды. ■

Следствие 2.1 (о рукопожатиях). У любого графа \mathbf{G} число вершин нечетной степени четно.

Определение 2.3. Граф $\mathbf{G}' = (\mathbf{V}', \mathbf{E}')$ называют подграфом графа \mathbf{G} , если $\mathbf{V}' \subseteq \mathbf{V}$, $\mathbf{E}' \subseteq \mathbf{E}$. В частности, подграфами \mathbf{G} являются пустой граф и сам граф. Все остальные подграфы называют собственными подграфами. Если $\mathbf{V}' = \mathbf{V}$, то \mathbf{G}' называют остовым подграфом \mathbf{G} .

Определение 2.4. *Путь* длины l называют последовательность вершин и соединяющих ребер:

$$v_0, e_1, v_1, e_2, \dots, v_{l-1}, e_l, v_l, \quad e_i = (v_{i-1}, v_i) \in \mathbf{E}, v_i \in \mathbf{V}.$$

При $v_0 \neq v_l$ путь называют *открытым* (*незамкнутым*), при $v_0 = v_l$ — *циклическим* (*замкнутым*); если $e_i \neq e_j$ ($i \neq j$) — *простым* (иногда *реберно-простым*).

Открытый путь, где все вершины различны, называют *вершинно-простым*, *простой цепью* или просто $(v_0 - v_l)$ -цепью.

Замкнутый путь, где все вершины (кроме конечных) различны, называют *простым циклом*, или *циклом*.

Замечание 2.2

Очевидно, что вершинно-простой путь является реберно-простым.

Степени всех вершин простого цикла четны. Далее, под циклом будем понимать простой цикл.

Пример 2.1. Рисунок 2.1 иллюстрирует определение 2.4.

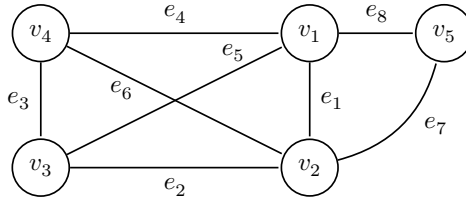


Рис. 2.1.

- 1) $v_1, e_1, v_2, e_6, v_4, e_4, v_1, e_1, v_2$ — открытый путь;
- 2) $v_1, e_1, v_2, e_2, v_3, e_5, v_1, e_4, v_4$ — открытый простой путь, но не цепь;
- 3) $v_1, e_1, v_2, e_6, v_4, e_3, v_3$ — простая $(v_1 - v_3)$ -цепь;
- 4) $v_1, e_5, v_3, e_2, v_2, e_6, v_4, e_3, v_3, e_5, v_1$ — замкнутый путь;
- 5) $v_1, e_1, v_2, e_6, v_4, e_3, v_3, e_2, v_2, e_7, v_5, e_8, v_1$ — замкнутый простой путь, но не цикл;
- 6) $v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_1$ — простой цикл.

Теорема 2.2. Если существует путь из вершины v в вершину u ($v \neq u$), то из ребер этого пути можно построить $(v - u)$ -цепь.

Доказательство. Пусть существует путь: $v = v_0, e_1, \dots, e_l, v_l = u$. Рассмотрим все пути из v в u , состоящие из ребер, входящих в этот путь. Среди

них выберем самый короткий $v = v_0, e'_1, v'_1, \dots, v'_{k-1}, e'_k, v'_k = u$. Покажем, что $v'_i \neq v'_j$ ($i \neq j$). Пусть $v'_i = v'_j$, тогда путь

$$v = v_0, e'_1, \dots, v'_i, e'_{j+1}, v'_{j+1}, \dots, v'_k = u$$

имеет длину $k - (j - i) < k$. Противоречие с предположением о минимальности пути. ■

Теорема 2.3 (аналог теоремы 2.2 для замкнутых путей). Если существует замкнутый простой путь, то из его ребер можно составить цикл.

Замечание 2.3

Требование простоты пути в теореме 2.3 существенно: рассмотрим путь v_0, e, v_1, e, v_0 . Из ребра e цикл построить нельзя.

Упражнение 2.1. Докажите теорему 2.3.

Теорема 2.4. Пусть $G = (V, E)$ — граф, все вершины которого имеют четную степень. Тогда для любого ребра существует замкнутый простой путь, содержащий это ребро.

Доказательство. Рассмотрим произвольное ребро $e_1 = (v_0, v_1)$. По предположению теоремы $\deg(v_1)$ — четное число. Тогда существует ребро $e_2 = (v_1, v_2)$, отличное от e_1 . В силу четности $\deg(v_2)$ найдется ребро $e_3 = (v_2, v_3)$ и т. д. Так как множество вершин V конечно и каждый раз берется новое неиспользованное ребро, инцидентное v_i , то в конце концов снова придем в вершину v_0 . ■

Теорема 2.5. В предположении теоремы 2.4, если $E \neq \emptyset$, то E есть объединение множества ребер некоторых циклов, никакие два из которых не имеют общих ребер.

Доказательство. Возьмем произвольное ребро e_1 . Строим для него замкнутый простой путь, как в теореме 2.4: $v_0, e_1, \dots, e_l, v_l$. Вершины в этом пути могут повторяться.

Допустим, что вершины v_0, v_1, \dots, v_{k-1} различны, а $v_k = v_i$, $i \in 0 : k - 1$. Тогда ребра $\{e_{i+1}, \dots, e_k\}$ образуют цикл.

Рассмотрим множество ребер $E_1 = E \setminus \{e_{i+1}, \dots, e_k\}$. Очевидно, что степень всех вершин графа $G_1 = (V, E_1)$ тоже останется четной. Продолжив этот процесс, пока очередное множество $E_k \neq \emptyset$, получим требуемое представление для E . ■

Следствие 2.2. Если в графе G существуют две различные цепи P_1 и P_2 , содержащие одни и те же различные вершины v и u , то из некоторого подмножества ребер P_1 и P_2 можно построить цикл.

Упражнение 2.2. Докажите, что в предположении теоремы 2.5 для любого ребра $e_i \in E$ существует цикл, его содержащий.

Определение 2.5. Если элементами множества ребер E являются упорядоченные пары вершин $[v, u]$, то граф $G = G(V, E)$ называют *ориентированным*, или *орграфом*. В этом случае элементы множества V называют *узлами*, а элементы E — *дугами*. Для дуги $e = [v, u] \in E$ узел v — *начало*, а узел u — *конец* дуги.

Для каждого узла v имеются две звезды: $\delta^+(v)$, $\delta^-(v)$ — множества выходящих и входящих в v дуг; при этом $|\delta^+(v)|$ — положительная, а $|\delta^-(v)|$ — отрицательная степени узла v .

Для орграфов имеет место теорема, аналогичная теореме Эйлера (см. теорему 2.1).

Теорема 2.6. Справедливо равенство

$$|E| = \sum_{v \in V} |\delta^+(v)| = \sum_{v \in V} |\delta^-(v)|.$$

Доказательство. Каждому ребру нужно сопоставить его начало или его конец. ■

Определение 2.6. *Направленным путем* длиной l называют последовательность $v_0, e_1, v_1, e_2, \dots, v_{l-1}, e_l, v_l$, где e_i — дуги $[v_{i-1}, v_i]$.

Аналогично определяют понятия открытого, замкнутого, простого пути, цепи и цикла.

Для орграфов справедливы теоремы, аналогичные теоремам 2.2 и 2.3.

Теорема 2.7. Если различные узлы v и u соединены ($v \rightarrow u$) путем, то существует ($v \rightarrow u$) цепь, построенная из дуг этого пути.

Теорема 2.8. Если в графе существует замкнутый ориентированный путь, то из дуг этого пути можно построить ориентированный цикл.

Замечание 2.4

Доказательства этих теорем аналогичны доказательствам теорем 2.2 и 2.3, но требование простоты для теоремы 2.3 в теореме 2.8 не нужно. Это связано с тем, что в ориентированном графе нет замкнутых путей, подобных пути v_0, e, v_1, e, v_0 . В пути v_0, e, v_1, y, v_0 две дуги $e = [v_0, v_1]$, $y = [v_1, v_0]$, $e \neq y$.

Теореме 2.4 соответствует следующая теорема.

Теорема 2.9. Пусть $G = (V, E)$ — ориентированный граф и для любого узла v справедливо $|\delta^+(v)| = |\delta^-(v)|$. Тогда для любой дуги существует замкнутый простой ориентированный путь, содержащий эту дугу.

Доказательство. Следует повторить рассуждения, использованные в доказательстве теоремы 2.4. При этом всегда следует двигаться согласно ориентации дуг: пусть дуга $e_1 = [v_0, v_1]$. Так как $|\delta^+(v_1)| = |\delta^-(v_1)|$, то существует дуга $e_2 \neq e_1$, исходящая из v_1 , и т. д. ■

Аналог теоремы 2.5 можно получить, если считать все дуги при доказательстве ориентированными.

Введем несколько важных *метрических* понятий в теории графов.

Определение 2.7. *Длиной цепи* называется число ребер (дуг) цепи.

Расстоянием между вершинами (узлами) u и v графа называется длина кратчайшей цепи ($u \rightarrow v$), обозначается $d(u, v)$. Если цепи нет, то принято считать расстояние бесконечным. Очевидно, что для неориентированных графов расстоянием является симметрическая функция: $d(u, v) = d(v, u)$.

Эксцентриситетом вершины u называется наибольшее расстояние между u и любой другой вершиной графа $G = G(V, E)$.

$$\epsilon(u) = \max_{v \in V} d(v, u)$$

Радиусом графа $G = G(V, E)$ называется минимальный эксцентриситет среди всех вершин графа:

$$R(G) = \min_{u \in V} \epsilon(u). \quad (2.1)$$

Вершина, на которой достигается равенство в (2.1), называется *центральной*.

Диаметром графа $G = G(V, E)$ называется максимальный эксцентриситет среди всех вершин графа.

$$D(G) = \max_{u \in V} \epsilon(u). \quad (2.2)$$

Вершина, на которой достигается равенство в (2.2), называется *периферийной*.

Центральных и периферийных вершин в графе может быть несколько. Для неориентированных графов их, очевидно, четное число.

Пример 2.2. Рассмотрим введенные метрические понятия на примере неориентированного графа на рисунке 2.2.

Тогда диаметр графа $D(\mathbf{G}) = 4$, периферийные вершины v_1 и v_5 , радиус $R(\mathbf{G}) = 2$, центральные вершины v_3 и v_6 .

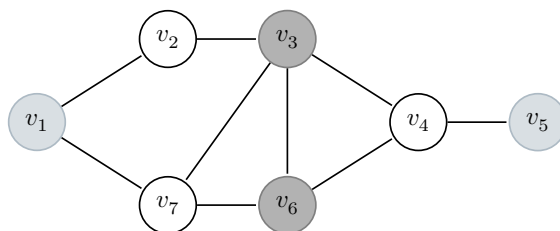


Рис. 2.2.

Историческая справка

Термин «граф» впервые ввел английский математик Джеймс Джозеф Сильвестр в 1878 г., хотя начальные задачи теории графов восходят еще к Леонарду Эйлеру [74].

Математический термин «граф» с дворянским титулом «граф» связывает общее происхождение от латинского слова «графико» — пишу.

Первые задачи теории графов были связаны с решением математических развлекательных задач и головоломок, например:

— задача о кенигсбергских мостах (задача Эйлера), развитие которой привело к циклу задач об обходах графов. Эти задачи будут рассмотрены далее в разделе 2.1.5;
— задачи о перевозках, решение которых привело к созданию эффективных методов решения транспортных задач и др.

Попытки решить сформулированную в середине XIX в. задачу четырех красок привели к появлению некоторых исследований графов, имеющих теоретическое и прикладное значение. Многие результаты, относящиеся к теории графов, были получены при решении практических проблем. Так, Г. Кирхгоф при составлении полной системы уравнений для токов и напряжений в электрической схеме предложил, по существу, представлять такую схему графом и находить в нем деревья, с помощью которых выделяются линейно независимые системы контуров.

В XX в. задачи, связанные с графами, начали возникать не только в физике, электротехнике, химии, биологии, экономике и т. д., но и внутри математики, в таких ее разделах, как алгебра, топология, теория вероятностей, теория чисел. Методы этих разделов стали успешно применяться для решения задач теории графов. В настоящее время графы эффективно используются в теории планирования и управления, теории расписаний, социологии, математической лингвистике, медицине, географии. Широкое применение находят графы в таких областях, как программирование, теория конечных автоматов (будет подробно рассмотрена в разделе 3.7), электроника.

Первый учебник по теории графов — «Теория конечных и бесконечных графов» — был опубликован в 1936 г. венгерским математиком Денишем Кенигом (см. теорему 2.33).

Наряду с термином «граф» в начале XX в. употреблялись в качестве синонимов и другие термины, например «карта», «комплекс», «диаграмма», «сеть», «лабиринт».

2.1.2. Машинное представление графов

Машинное представление графов допускает большое разнообразие. Сложность получения ответа на тот или иной вопрос относительно данного графа во многом зависит от способа представления графа. Поэтому в алгоритмах на графах связь *алгоритм + структура данных* проявляется очень сильно. Один и тот же алгоритм, реализованный на различных структурах данных, очень часто приводит к совершенно разным программам. Во многих задачах на графах выбор представления является решающим для повышения эффективности алгоритма.

2.1.2.1. Матрица инциденций

Граф G , имеющий n вершин и m ребер, представляется матрицей размером $n \times m$. Строки матрицы соответствуют вершинам, а столбцы — ребрам графа. Для неориентированного графа матрица строится следующим образом. Если u, v — смежные вершины, то столбец (u, v) содержит 1 в строках u и v . Остальные элементы равны нулю.

В случае ориентированного графа при наличии ориентированного ребра $[u, v]$ для столбца $[u, v]$ в строке u ставится -1 , в строке v — соответственно 1, а остальные элементы равны нулю.

Пример 2.3. Матрицы инциденций для неориентированного и ориентированного графов на рисунке 2.3а и б представлены соответственно в таблицах 2.1 и 2.2.

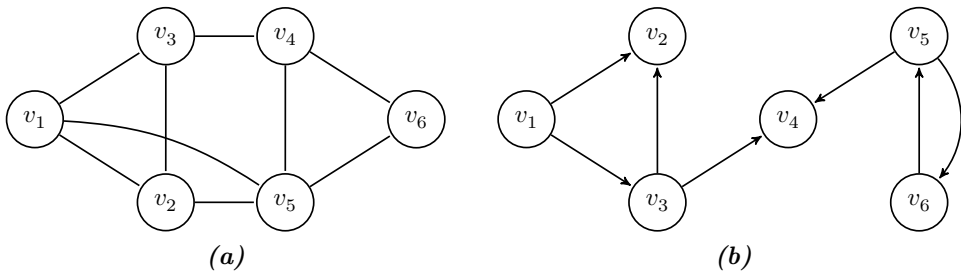


Рис. 2.3.

Замечание 2.5

Недостатками такого представления графов являются большой объем данных и неудобство доступа к информации. Для ответа на вопрос, существует ли в графе ребро (x, y) , в худшем случае нужно осуществить полный перебор столбцов, то есть сделать m шагов.

Таблица 2.1.

	(v_1, v_2)	(v_1, v_3)	(v_1, v_5)	(v_2, v_3)	(v_2, v_5)	(v_3, v_4)	(v_4, v_5)	(v_4, v_6)	(v_5, v_6)
v_1	1	1	1	0	0	0	0	0	0
v_2	1	0	0	1	1	0	0	0	0
v_3	0	1	0	1	0	1	0	0	0
v_4	0	0	0	0	0	1	1	1	0
v_5	0	0	1	0	1	0	1	0	1
v_6	0	0	0	0	0	0	0	1	1

Таблица 2.2.

	$[v_1, v_2]$	$[v_1, v_3]$	$[v_3, v_2]$	$[v_3, v_4]$	$[v_5, v_4]$	$[v_5, v_6]$	$[v_6, v_5]$
v_1	-1	-1	0	0	0	0	0
v_2	1	0	1	0	0	0	0
v_3	0	1	-1	-1	0	0	0
v_4	0	0	0	1	1	0	0
v_5	0	0	0	0	-1	-1	1
v_6	0	0	0	0	0	1	-1

2.1.2.2. Матрица смежности

Матрица смежности для графа G , имеющего n вершин, представляется $n \times n$ матрицей. Ее строки и столбцы соответствуют вершинам графа. Элемент матрицы, стоящий на пересечении строки v_i и столбца v_j , равен 1, если существует ребро (v_i, v_j) (дуга $[v_i, v_j]$ в случае ориентированного графа). Остальные элементы матрицы равны нулю.

Пример 2.4. Матрицы смежности для неориентированного и ориентированного графов из примера 2.3 соответственно приведены в таблице 2.3а и б.

Замечание 2.6

1. Очевидно, что для неориентированного графа матрица смежности симметричная.

Таблица 2.3.

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	1	0	1	0
v_2	1	0	1	0	1	0
v_3	1	1	0	1	0	0
v_4	0	0	1	0	1	1
v_5	1	1	0	1	0	1
v_6	0	0	0	1	1	0

(a)

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	1	0	0	0
v_2	0	0	0	0	0	0
v_3	0	1	0	1	0	0
v_4	0	0	0	0	0	0
v_5	0	0	0	1	0	1
v_6	0	0	0	0	1	0

(b)

2. Преимуществом такого представления является решение за один шаг вопроса: существует ли в графе ребро (x, y) ?

3. Недостатком матрицы смежности является то, что объем данных не зависит от числа ребер и всегда равен n^2 .

2.1.2.3. Списки инцидентности

Для каждой вершины графа $v \in V$ вводится список вершин $u \in V$ таких, что существует ребро (v, u) (дуга $[v, u]$ для ориентированного графа). Начало каждого списка хранится в таблице указателей $begin[\]$.

Таким образом, $begin[v]$ — это указатель на начало соответствующего списка инцидентности для вершины v . Весь такой список будем обозначать $entry[v]$. Каждый элемент списка — это структура из двух полей (рисунок 2.4).

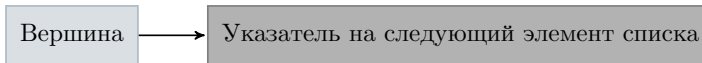


Рис. 2.4.

Замечание 2.7

Для неориентированных графов каждое ребро (u, v) представлено дважды: через вершину v в списке $entry[u]$ и через вершину u в списке $entry[v]$.

Пример 2.5. Составим списки инцидентности для неориентированного и ориентированного графов из примера 2.3 (рисунок 2.5а и б).

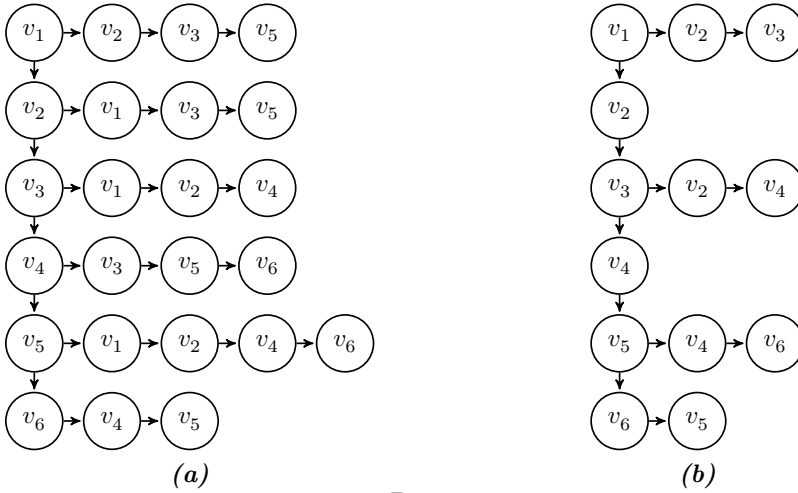


Рис. 2.5.

2.1.2.4. Список ребер

Представляем граф в виде списка ребер или дуг как пар вершин с учетом направления дуги для ориентированных графов. Иногда для ускорения поиска список упорядочивают в лексикографическом порядке.

Пример 2.6. Составим списки ребер для неориентированного и ориентированного графов из примера 2.3.

$$(v_1, v_2) \rightarrow (v_1, v_3) \rightarrow (v_1, v_5) \rightarrow (v_2, v_3) \rightarrow (v_2, v_5) \rightarrow (v_3, v_4) \rightarrow (v_4, v_5) \rightarrow \\ \rightarrow (v_4, v_6) \rightarrow (v_5, v_6),$$

$$(v_1, v_2) \rightarrow (v_1, v_3) \rightarrow (v_3, v_2) \rightarrow (v_3, v_4) \rightarrow (v_5, v_4) \rightarrow (v_5, v_6) \rightarrow (v_6, v_5).$$

Замечание 2.8

В таком представлении удобно добавлять и удалять ребра, представлять сильно разреженные графы. Неудобно определять смежность вершин и ребер, осуществлять перебор инцидентных заданной вершине ребер.

2.1.3. Поиск в глубину и ширину в графе

Методы поиска в графе являются основой для построения алгоритмов на графах. Под поиском на графах будем понимать процесс просмотра всех вершин графа в целях отыскания вершин, удовлетворяющих некоторому условию.

Рассмотрим два подхода, базирующихся на двух основных идеях в программировании:

- 1) стеке¹ (магазине) — «первым пришел, последним ушел»;
- 2) очереди — «первым пришел, первым ушел».

Будем считать, что для описания графа используется одно из представлений предыдущего подраздела, например списки инцидентности.

Первый алгоритм основан на идее использования стека и называется *поиск в глубину в графе*² (другие названия — *возвратный ход*, *бектрекинг*, *обход графа в глубину*).

Применение правила LIFO (Last In First Out — последним пришел, первым обслужен) приводит к следующей стратегии: идти «вглубь» графа, пока это возможно (есть непросмотренные вершины), возвращаться и искать другой путь, когда таких вершин нет. Поиск продолжается, пока не просмотрены все вершины, достижимые из исходной.

Замечание 2.9

Как правило, стек реализуется в виде однонаправленного списка — каждый элемент списка указывает только на следующий. При этом доступ возможен только к верхнему элементу (вершине стека).

Обозначим стек S , вершину стека — $\text{top}(S)$, список просмотренных вершин — L .

Алгоритм 2.1. Поиск в глубину в графе «DFS»

```

 $S \leftarrow v_0$  // записываем в стек произвольную вершину  $v_0$ , помечаем ее как
просмотренную
 $L \leftarrow v_0$ 
while  $S \neq \{\emptyset\}$  do
   $v \leftarrow \text{top}(S)$ 
  if существует смежная с  $v$  непросмотренная вершина  $u \notin L$  then
     $S \leftarrow u$  // записываем в стек вершину  $u$ , помечаем ее как просмотренную
     $L \leftarrow u$ 
  else
    Удаляем вершину  $v$  из стека
  end if
end while

```

Пример 2.7. Рассмотрим работу алгоритма на примере графа, представленного на рисунке 2.6. Поиск начнем с вершины v_1 .

Будем схематично изображать стек развернутым по горизонтали. Считаем, что вершина стека находится слева.

¹ Понятие стека (англ. stack — стопка) было введено Аланом Тьюрингом в 1946 г.

² DFS: Depth-first search.

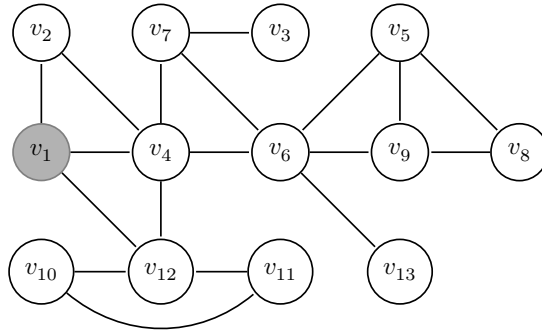


Рис. 2.6.

Порядок просмотра вершин следующий:

$$v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6 \rightarrow v_5 \rightarrow v_8 \rightarrow v_9 \rightarrow v_7 \rightarrow v_3 \rightarrow v_{13} \rightarrow v_{12} \rightarrow v_{10} \rightarrow v_{11}.$$

Изменения состояния стека показаны в таблице 2.4.

Таблица 2.4.

Текущая вершина	Действие	Состояние стека
v_1	Добавляем вершину v_1 в стек	v_1
v_2	Добавляем вершину v_2 в стек	$v_2 v_1$
v_4	Добавляем вершину v_4 в стек	$v_4 v_2 v_1$
v_6	Добавляем вершину v_6 в стек	$v_6 v_4 v_2 v_1$
v_5	Добавляем вершину v_5 в стек	$v_5 v_6 v_4 v_2 v_1$
v_8	Добавляем вершину v_8 в стек	$v_8 v_5 v_6 v_4 v_2 v_1$
v_9	Добавляем вершину v_9 в стек	$v_9 v_8 v_5 v_6 v_4 v_2 v_1$
v_9	Удаляем вершину v_9 из стека	$v_8 v_5 v_6 v_4 v_2 v_1$
v_8	Удаляем вершину v_8 из стека	$v_5 v_6 v_4 v_2 v_1$
v_5	Удаляем вершину v_5 из стека	$v_6 v_4 v_2 v_1$
v_7	Добавляем вершину v_7 в стек	$v_7 v_6 v_4 v_2 v_1$
v_3	Добавляем вершину v_3 в стек	$v_3 v_7 v_6 v_4 v_2 v_1$
v_3	Удаляем вершину v_3 из стека	$v_7 v_6 v_4 v_2 v_1$
v_7	Удаляем вершину v_7 из стека	$v_6 v_4 v_2 v_1$
v_{13}	Добавляем вершину v_{13} в стек	$v_{13} v_6 v_4 v_2 v_1$
v_{13}	Удаляем вершину v_{13} из стека	$v_6 v_4 v_2 v_1$
v_6	Удаляем вершину v_6 из стека	$v_4 v_2 v_1$
v_{12}	Добавляем вершину v_{12} в стек	$v_{12} v_4 v_2 v_1$
v_{10}	Добавляем вершину v_{10} в стек	$v_{10} v_{12} v_4 v_2 v_1$
v_{11}	Добавляем вершину v_{11} в стек	$v_{11} v_{10} v_{12} v_4 v_2 v_1$
v_{11}	Удаляем вершину v_{11} из стека	$v_{10} v_{12} v_4 v_2 v_1$

Продолжение таблицы 2.4

Текущая вершина	Действие	Состояние стека
v_{10}	Удаляем вершину v_{10} из стека	$v_{12} v_4 v_2 v_1$
v_{12}	Удаляем вершину v_{12} из стека	$v_4 v_2 v_1$
v_4	Удаляем вершину v_4 из стека	$v_2 v_1$
v_2	Удаляем вершину v_2 из стека	v_1
v_1	Удаляем вершину v_1 из стека	Стек пуст

Замена стека, используемого при обходе в глубину, очередью FIFO (First In First Out — первым пришел, первым обнаружен) приводит к другому классическому алгоритму — *поиск в ширину в графе*¹ (другое название — *обход графа в ширину*).

Алгоритм поиска в ширину просматривает все вершины, достижимые из начальной. Происходит движение «вширь» графа (сначала просматриваются все соседние вершины, затем соседи соседей и т. д.).

 **Замечание 2.10**

Добавление вершины возможно лишь в конец очереди, выборка — только из начала очереди.

Обозначим очередь Q , начало и конец очереди — $\text{start}(Q)$ и $\text{end}(Q)$, список просмотренных вершин — L .

Алгоритм 2.2. Поиск в ширину в графе «BFS»

```

end(Q) ← v0           // записываем в конец очереди произвольную вершину v0
L ← v0                // помечаем ее как просмотренную
while Q ≠ {∅} do
  v ← start(Q)         // v — вершина, находящаяся в начале очереди
  if существует смежная с v непросмотренная вершина u ∉ L then
    end(Q) ← u         // записываем в конец очереди вершину u
    L ← u              // помечаем вершину u как просмотренную
  else
    Удаляем вершину v из начала очереди
  end if
end while

```

¹ BFS: Breadth first search.

 **Замечание 2.11**

В отличие от предыдущего алгоритма, в силу конструктивных особенностей реализации очереди (замечание 2.10) здесь просматриваем все вершины одного уровня и затем переходим на следующий.

Рассмотрим работу алгоритма на графе из примера 2.7 (см. рисунки 2.6).

Пример 2.8. Будем схематично изображать очередь развернутой по горизонтали. Считаем, что хвост очереди находится слева, а начало — справа.

Порядок просмотра вершин следующий:

$$v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_{12} \rightarrow v_6 \rightarrow v_7 \rightarrow v_{10} \rightarrow v_{11} \rightarrow v_5 \rightarrow v_9 \rightarrow v_{13} \rightarrow v_3 \rightarrow v_8.$$

Изменения состояния очереди показаны в таблице 2.5.

Таблица 2.5.

Вершина на выходе очереди	Действие	Состояние очереди
	Добавляем вершину v_1 в очередь	v_1
v_1	Добавляем вершину v_2 в очередь	$v_2 v_1$
v_1	Добавляем вершину v_4 в очередь	$v_4 v_2 v_1$
v_1	Добавляем вершину v_{12} в очередь	$v_{12} v_4 v_2 v_1$
v_2	Удаляем вершину v_1 из очереди	$v_{12} v_4 v_2$
v_4	Удаляем вершину v_2 из очереди	$v_{12} v_4$
v_4	Добавляем вершину v_6 в очередь	$v_6 v_{12} v_4$
v_4	Добавляем вершину v_7 в очередь	$v_7 v_6 v_{12} v_4$
v_{12}	Удаляем вершину v_4 из очереди	$v_7 v_6 v_{12}$
v_{12}	Добавляем вершину v_{10} в очередь	$v_{10} v_7 v_6 v_{12}$
v_{12}	Добавляем вершину v_{11} в очередь	$v_{11} v_{10} v_7 v_6 v_{12}$
v_6	Удаляем вершину v_{12} из очереди	$v_{11} v_{10} v_7 v_6$
v_6	Добавляем вершину v_5 в очередь	$v_5 v_{11} v_{10} v_7 v_6$
v_6	Добавляем вершину v_9 в очередь	$v_9 v_5 v_{11} v_{10} v_7 v_6$
v_6	Добавляем вершину v_{13} в очередь	$v_{13} v_9 v_5 v_{11} v_{10} v_7 v_6$
v_7	Удаляем вершину v_6 из очереди	$v_{13} v_9 v_5 v_{11} v_{10} v_7$
v_7	Добавляем вершину v_3 в очередь	$v_3 v_{13} v_9 v_5 v_{11} v_{10} v_7$
v_{10}	Удаляем вершину v_7 из очереди	$v_3 v_{13} v_9 v_5 v_{11} v_{10}$
v_{11}	Удаляем вершину v_{10} из очереди	$v_3 v_{13} v_9 v_5 v_{11}$
v_5	Удаляем вершину v_{11} из очереди	$v_3 v_{13} v_9 v_5$
v_5	Добавляем вершину v_8 в очередь	$v_8 v_3 v_{13} v_9 v_5$

Продолжение таблицы 2.5

Вершина на выходе очереди	Действие	Состояние очереди
v_9	Удаляем вершину v_5 из очереди	$v_8 v_3 v_{13} v_9$
v_{13}	Удаляем вершину v_9 из очереди	$v_8 v_3 v_{13}$
v_3	Удаляем вершину v_{13} из очереди	$v_8 v_3$
v_8	Удаляем вершину v_3 из очереди	v_8
	Удаляем вершину v_8 из очереди	Очередь пуста

2.1.4. Связность

Определение 2.8. Граф G называют *связным*, если для любых двух различных вершин существует соединяющая их цепь. (Согласно теореме 2.2 можно говорить о пути.)

Определение 2.9. Пусть S — непустое подмножество вершин графа $G = (V, E)$. Обозначим через $E(S) \subseteq E$ подмножество ребер, концевые вершины которых принадлежат S . Граф $G' = (S, E(S))$ называют *подграфом* G , *порожденным* S .

Определение 2.10. Определим бинарное отношение \equiv на множестве вершин графа $G(V, E)$: $v \equiv u$, если существует $(v - u)$ -цепь.

Теорема 2.10. Отношение \equiv есть отношение эквивалентности.

Доказательство. Действительно, из определения 2.10 очевидны рефлексивность и симметричность. Далее, если $v \equiv u$, $u \equiv w$, то, соединяя последовательно две цепи $(v - u)$, $(u - w)$, получаем цепь $(v - w)$. Доказана транзитивность. ■

Это отношение разбивает множество вершин V на непересекающиеся классы эквивалентности V_1, V_2, \dots, V_k .

Определение 2.11. Подграфы графа G , порожденные V_1, \dots, V_k : $(V_1, E(V_1)), \dots, (V_k, E(V_k))$, называют *связными компонентами* или *компонентами связности* графа G , а k — *степенью связности* графа G .

 **Замечание 2.12**

Любая вершина и любое ребро принадлежат одной и только одной компоненте связности.

Для определения всех компонент связности неориентированного графа можно воспользоваться поиском в глубину (алгоритм 2.1).

Алгоритм 2.3. Поиск компонент связности неориентированного графа

```

while существуют непросмотренные вершины графа do
    u ← первая непросмотренная вершина
    Поиск в глубину (DFS) из вершины u
    Очередная компонента связности ← все просмотренные на данном шаге
    вершины
end while

```

Определение 2.12. Рассмотрим граф $G = (V, E)$ и его ребро e . Определим подграф $G - e = (V, E \setminus \{e\})$. Заметим, что при этом концевые точки ребра e остаются в графе $G - e$.

Теорема 2.11. Если в связном графе $G = (V, E)$ ребро e принадлежит некоторому циклу, то граф $G - e$ остается связным.

Доказательство. Пусть $v_0, e, v_1, e_2, \dots, e_l, v_l (= v_0)$ — цикл, содержащий ребро e . В любом пути, который включал удаленное ребро $e = (v_0, v_1)$, это ребро можно заменить на путь $(v_0 =)v_l, e_l, \dots, e_2, v_1$. ■

Справедливо и обратное утверждение.

Теорема 2.12. Пусть $G = (V, E)$ — связный граф, $e \in E$. Если $G - e$ также связный граф, то существует цикл в G , содержащий ребро e .

Доказательство. Пусть $e = (v, u)$, тогда существует $(v - u)$ -цепь в графе $G - e$: $(v =)v_0, e_1, v_1, e_2, \dots, v_{l-1}, e_l, v_l (= u)$. Добавив к этой цепи ребро e и вершину v , получим цикл в графе G . ■

Определение 2.13. Ребро e графа $G = (V, E)$ называют *мостом*, или *разделяющим ребром*, или *ребром отделимости*, или *перешейком*, если степень связности графа $G - e$ больше степени связности графа G .

Если при удалении вершины вместе с инцидентными ей ребрами степень связности графа увеличивается, вершина называется *точкой сочленения* или *разделяющей вершиной* или *шарниром*.

Граф называется *реберно связным*, если в нем отсутствуют мосты. В противном случае граф называется *реберно отделимым*.

При удалении всех мостов граф распадается на компоненты связности, которые называются *компонентами реберной двусвязности*.

Замечание 2.13

Если граф G несвязный, то при удалении ребра e видоизменяется лишь компонента связности, содержащая e .

Опираясь на теоремы 2.11 и 2.12, легко получить критерий моста.

Теорема 2.13. Ребро e графа G — мост тогда и только тогда, когда в G нет циклов, содержащих это ребро.

Упражнение 2.3. Докажите самостоятельно теорему 2.13.

Пример 2.9. Рассмотрим граф, представленный на рисунке 2.7.

Для данного графа мостом является каждое из ребер (v_4, v_5) , (v_5, v_6) и (v_5, v_7) . На рисунке они имеют большую «толщину». Действительно, в отличие от остальных ребер, для этих трех в графе нет циклов, их содержащих.

Вершины v_4 и v_5 — точки сочленения. При удалении всех мостов получаем четыре компоненты реберной двусвязности $\{v_1, v_2, v_3, v_4\}$, $\{v_5\}$, $\{v_6\}$ и $\{v_7\}$.

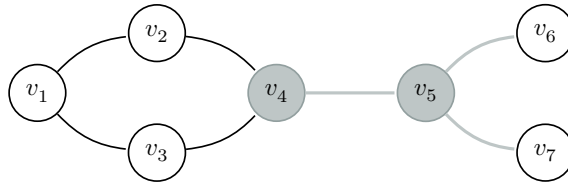


Рис. 2.7.

Рассмотрим задачу нахождения всех мостов графа G . Эта задача решается с помощью двух поисков в глубину.

Алгоритм 2.4. Нахождение всех мостов графа

// Инициализация

$L \leftarrow \{\emptyset\}$

$color \leftarrow 1$

// Первый поиск в глубину

while существуют непросмотренные вершины графа **do**

$u \leftarrow$ первая непросмотренная вершина

 Поиск в глубину: DFS(u)

 При прохождении ребра (v_i, v_j) ориентируем его против направления движения: запрещаем прохождение по ребру в направлении от вершины v_i к вершине v_j

$L \leftarrow$ все просмотренные на данном шаге вершины

end while

// Второй поиск в глубину с учетом ориентированности ребер

// u в порядке записи в L

for each $u \in L$ и u — непросмотрена **do**

Поиск в глубину: DFS(u)

Вершины получившийся компоненты красим в цвет $color$

$color \leftarrow color + 1$

end for

Ребра графа G , соединяющие вершины разного цвета, будут искомыми мостами.

Замечание 2.14

При поиске в глубину для неориентированного графа ребра, просмотренные в процессе обхода, называются прямыми, остальные — обратными. Для обратного ребра движение в алгоритме разрешено в обе стороны.

Очевидно, что обратное ребро не может быть мостом, так как всегда принадлежит некоторому циклу.

Пример 2.10. Рассмотрим работу алгоритма 2.4 на примере графа, показанного на рисунке 2.7.

Запускаем поиск в глубину из вершины v_1 . После первого шага получаем следующий граф (рисунок 2.8). Список L имеет вид:

$$L = \{1, 2, 4, 3, 5, 6, 7\}.$$

Ребро (v_1, v_3) является обратным (отмечено пунктиром на рисунке 2.8), остальные ребра — прямые.

Проводим второй поиск в глубину для графа, показанного на рисунке 2.8, согласно списку L . Получаем следующие четыре компоненты: $\{v_1, v_3, v_4, v_2\}$ — цвет 1, $\{v_5\}$ — цвет 2, $\{v_6\}$ — цвет 3 и $\{v_7\}$ — цвет 4 (рисунок 2.9).

Отсюда определяем мосты графа: (v_4, v_5) , (v_5, v_6) , (v_5, v_7) .

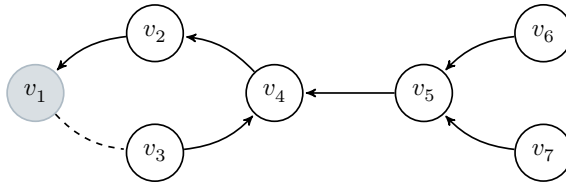


Рис. 2.8.

Пример 2.11. Рассмотрим еще один пример работы алгоритма 2.4 для графа, показанного на рисунке 2.10.

Начинаем поиск в глубину из вершины v_0 . После первого шага получаем следующий граф (рисунок 2.11). Список L имеет вид:

$$L = \{0, 1, 2, 5, 3, 4, 7, 6, 8, 9\}.$$

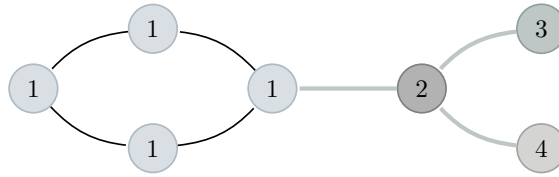


Рис. 2.9.

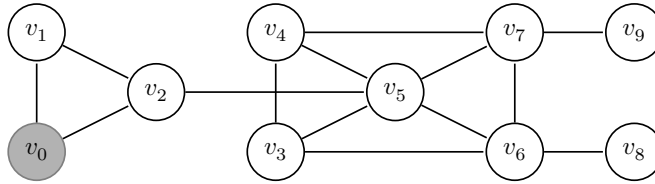


Рис. 2.10.

Обратные ребра $(v_0, v_2), (v_3, v_6), (v_5, v_6), (v_4, v_5), (v_5, v_7)$ отмечены пунктиром на рисунке 2.11.

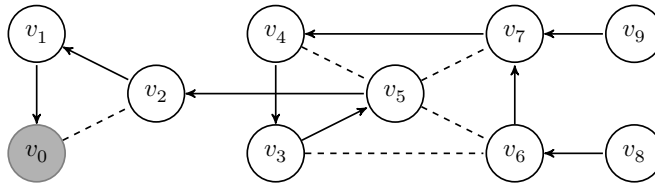


Рис. 2.11.

Проводим второй поиск в глубину для графа на рисунке 2.11 согласно списку L . Получаем следующие компоненты: $\{v_0, v_1, v_2\}$ — цвет 1, $\{v_3, v_4, v_5, v_6, v_7\}$ — цвет 2, $\{v_8\}$ — цвет 3 и $\{v_9\}$ — цвет 4 (рисунок 2.12).

Отсюда определяем мосты графа: $(v_2, v_5), (v_6, v_8), (v_7, v_9)$.

Для ориентированных графов понятие связности вводится аналогичным образом.

Определение 2.14. Пусть $G = (V, E)$ — ориентированный граф. Рассмотрим неориентированный граф $G' = (V, E')$. Граф G' получается из графа G , если убрать ориентацию дуг. Если получившийся граф G' является связным, то G называют *слабосвязным* (т. е. можно попасть из любой вершины в любую, если не обращать внимания на ориентацию).

Ориентированный граф G называют *сильносвязным*, если для любой пары узлов $v, u \in V$ существуют $(v \rightarrow u)$ - и $(u \rightarrow v)$ -цепи.

Компоненты связности сильносвязного графа иногда называют *сильными компонентами*, или *бикомпонентами*.

Для ориентированных графов существует еще одно важное понятие.

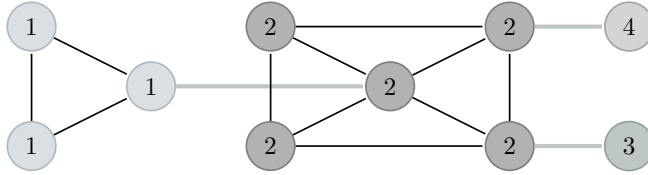


Рис. 2.12.

Определение 2.15. *Графом конденсации, или графом Герца, или конденсацией* ориентированного графа G , называют такой ориентированный граф G' , вершинами которого служат компоненты сильной связности G , а дуга в G' между двумя вершинами проводится, только если в графе G существует хотя бы одна дуга между вершинами, входящими в соответствующие компоненты связности.

Замечание 2.15

| Граф конденсации не содержит кратных ребер по построению.

Теорема 2.14. Граф конденсации не содержит циклов.


Доказательство. Вершины графа конденсации G' (то есть компоненты сильной связности исходного графа G) будем обозначать C_i . Предположим, что есть путь из $C_i \in G'$ в $C_j \in G'$. Покажем, что в этом случае нет пути из C_j в C_i .

Наличие пути в графе G' из C_i в C_j означает, что в исходном графе есть путь из вершины $v \in C_i$ в вершину $u \in C_j$. Из предположения о наличии пути из C_j в C_i следует, что в исходном графе есть путь из вершины $u_1 \in C_j$ в вершину $v_1 \in C_i$.

Но пары вершин v, v_1 находятся в одной компоненте сильной связности C_i , поэтому между ними есть путь; аналогично для u, u_1 . В итоге, объединяя пути, получаем, что есть путь от v до u и одновременно путь от u до v . Следовательно, вершины u и v должны принадлежать одной компоненте сильной связности, полученное противоречие доказывает теорему. ■

Для алгоритма построения компонент сильной связности введем следующие понятия.

Определение 2.16. *Транспонированием* графа $G = (V, E)$ называется граф $G^T = (V, E^T)$, в котором $E^T = \{(u, v) : (v, u) \in E\}$, т. е. граф, полученный из G изменением направления каждого ребра на противоположное.

 **Замечание 2.16**

Очевидно, что в транспонированном графе будут те же компоненты сильной связности, что и в исходном графе. Более того, граф конденсации G^T совпадает с транспонированным графом конденсации исходного графа G .

Первый шаг алгоритма повторяет алгоритм 2.3 — проходим по всем вершинам графа и из каждой еще не просмотренной вершины u вызываем поиск в глубину $DFS(u)$. При этом для каждой вершины u будем запоминать время выхода из вершины $T_{out}(u)$. Эти времена выхода играют ключевую роль в алгоритме.

В начале алгоритма запускается таймер времени, и при завершении поиска в глубину из очередной вершины время увеличивается.

Определение 2.17. Время выхода $T_{out}(C)$ из компоненты C сильной связности определим как $\max\{T_{out}(v) \mid v \in C\}$.

Справедлива следующая теорема.

Теорема 2.15. Пусть C и C' — две различные компоненты сильной связности, и в графе конденсации между ними есть дуга (C, C') . Тогда $T_{out}(C) > T_{out}(C')$.

Доказательство. Возможны два случая в зависимости от того, в какую из компонент в первую очередь зайдет поиск в глубину.

1. Первой была достигнута компонента C . Это означает, что в какой-то момент времени поиск в глубину зашел в некоторую вершину $v \in C$, при этом все остальные вершины компонент C и C' еще не просмотрены. Но, так как по условию теоремы в графе конденсаций есть дуга (C, C') , то из вершины v есть путь не только ко всем вершинам *своей* компоненты C , но и ко всем вершинам компоненты C' . Это означает, что при запуске из вершины v обход в глубину пройдет по всем вершинам компонент C и C' , то есть

$$T_{out}(v) > T_{out}(u), \quad \forall u \in C \cup C', u \neq v.$$

2. Первой была достигнута компонента C' . Аналогично предыдущему случаю, в какой-то момент времени поиск в глубину зашел в некоторую вершину $v \in C'$, причем все остальные вершины компонент C и C' еще не просмотрены.

В силу существования дуги (C, C') и ацикличности графа конденсаций (теорема 2.14) не существует обратного пути из C' в C , т. е. поиск в глубину из вершины v не достигнет вершин компоненты C . Это означает, что они будут просмотрены поиском в глубину позже и время выхода у них будет больше. ■

Следствие 2.3. Любое ребро (C, C') в графе конденсаций идет из компоненты с большей величиной T_{out} в компоненту с меньшей величиной.

Если мы отсортируем все вершины $v \in V$ в порядке убывания времени выхода $T_{out}(v)$, то первой окажется некоторая вершина u , принадлежащая *корневой* компоненте сильной связности, т. е. такой, в которую не входит ни одно ребро в графе конденсаций.

Теперь нам нужно попытаться запустить такой поиск в глубину из этой вершины u , который обошел бы только эту компоненту сильной связности и не зашел бы ни в какую другую.

Если мы научимся это делать, то будем постепенно выделять все компоненты сильной связности: удалив из графа вершины первой выделенной компоненты, мы снова найдем среди оставшихся вершину с наибольшей величиной T_{out} , снова запустим из нее DFS и т. д.

Чтобы научиться делать такой обход, используем понятие транспонированного графа G^T .

Как уже отмечалось (замечание 2.16), в этом графе будут те же компоненты сильной связности, что и в исходном графе, и граф конденсации графа G^T равен транспонированному графу конденсации исходного графа G . Это означает, что теперь из рассматриваемой нами *корневой* компоненты уже не будут выходить дуги в другие компоненты.

Таким образом, чтобы обойти всю *корневую* компоненту сильной связности, содержащую некоторую вершину v , достаточно запустить поиск в глубину из вершины v в графе G^T . Этот обход посетит все вершины этой компоненты сильной связности и только их.

Как уже говорилось, дальше мы можем мысленно удалить эти вершины из графа, найти очередную вершину с максимальным значением $T_{out}(u)$ и запустить поиск в глубину на транспонированном графе из нее и т. д.

Алгоритм 2.5. Поиск компонент сильной связности

// Инициализация

$L \leftarrow \{\emptyset\}$

// Серия поисков в глубину графа G

while существуют непросмотренные вершины графа **do**

$u \leftarrow$ первая непросмотренная вершина

 Поиск в глубину: DFS(u)

 // Создаем список вершин в порядке увеличения времени выхода T_{out}

$L \leftarrow$ вершины в порядке увеличения времени выхода T_{out} .

end while

Строим транспонированный граф G^T .

Инвертируем список L (в порядке уменьшения времени выхода T_{out}).

// Серия поисков в глубину графа G^T в порядке уменьшения времени выхода

for each $u \in L$ и u — непросмотрена **do**

Поиск в глубину: DFS(u)

Очередная компонента сильной связности \leftarrow множество просмотренных на данном шаге вершин

end for

Замечание 2.17

Алгоритм 2.5 был предложен независимо Косарайю (Kosaraju) и Шариром (Sharir) в 1979 г.

Пример 2.12. Рассмотрим работу алгоритма 2.5 на примере графа, показанного на рисунке 2.13.

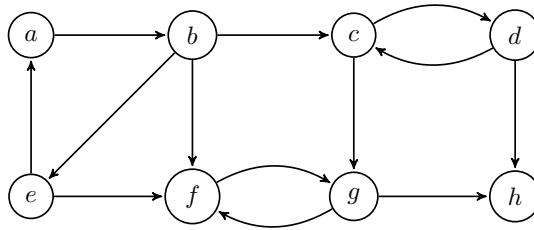


Рис. 2.13.

Проводим серию поисков в глубину. Запускаем DFS(a). Протокол работы алгоритма показан в таблице 2.6.

Таблица 2.6.

Текущая вершина	Действие	Состояние стека	T	T_{out}
a	Добавляем вершину a в стек	a	1	
b	Добавляем вершину b в стек	ba	2	
c	Добавляем вершину c в стек	cba	3	
d	Добавляем вершину d в стек	$dcba$	4	
h	Добавляем вершину h в стек	$hdcba$	5	
d	Удаляем вершину h из стека	$dcba$	6	$T_{out}(h) = 6$
c	Удаляем вершину d из стека	cba	7	$T_{out}(d) = 7$
g	Добавляем вершину g в стек	$gcb a$	8	
f	Добавляем вершину f в стек	$fgcb a$	9	
g	Удаляем вершину f из стека	$gcb a$	10	$T_{out}(f) = 10$

Продолжение таблицы 2.6

Текущая вершина	Действие	Состояние стека	T	T_{out}
c	Удаляем вершину g из стека	$c b a$	11	$T_{out}(g) = 11$
b	Удаляем вершину c из стека	$b a$	12	$T_{out}(c) = 12$
e	Добавляем вершину e в стек	$e b a$	13	
b	Удаляем вершину e из стека	$b a$	14	$T_{out}(e) = 14$
a	Удаляем вершину b из стека	a	15	$T_{out}(b) = 15$
	Удаляем вершину a из стека	Стек пуст	16	$T_{out}(a) = 16$

 **Замечание 2.18**

После окончания поиска в глубину $DFS(a)$ все вершины графа были просмотрены. Серия на шаге 1 состоит из одного поиска.

Запишем последовательность вершин в порядке убывания T_{out} :

$$a(16), b(15), e(14), c(12), g(11), f(10), d(7), h(5).$$

Транспонируем исходный граф и проведем серию поисков в глубину в порядке уменьшения T_{out} . Имеем:

$$\begin{aligned} a \rightarrow e \rightarrow b, \quad C_1 &= \{a, e, b\}, \\ c \rightarrow d \quad C_2 &= \{c, d\}, \\ g \rightarrow f \quad C_3 &= \{g, f\}, \\ h \quad C_4 &= \{h\}. \end{aligned}$$

Граф конденсации исходного графа представлен на рисунке 2.14.

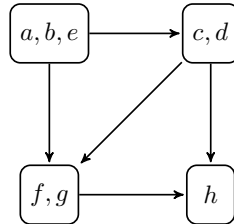


Рис. 2.14.

Пример 2.13. Рассмотрим еще один пример работы алгоритма 2.5 для графа на рисунке 2.15.

Проводим серию поисков в глубину. Запускаем $DFS(a)$. Протокол работы алгоритма показан в таблице 2.7.

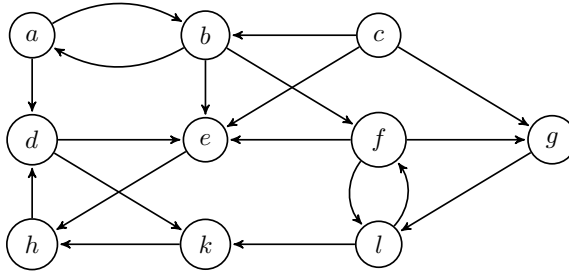


Рис. 2.15.

Таблица 2.7.

Текущая вершина	Действие	Состояние стека	T	T_{out}
a	Добавляем вершину a в стек	a	1	
b	Добавляем вершину b в стек	ba	2	
e	Добавляем вершину e в стек	eba	3	
h	Добавляем вершину h в стек	$heba$	4	
d	Добавляем вершину d в стек	$dheba$	5	
k	Добавляем вершину k в стек	$kdheba$	6	
d	Удаляем вершину k из стека	$dheba$	7	$T_{out}(k) = 7$
h	Удаляем вершину d из стека	$heba$	8	$T_{out}(d) = 8$
e	Удаляем вершину h из стека	eba	9	$T_{out}(h) = 9$
b	Удаляем вершину e из стека	ba	10	$T_{out}(e) = 10$
f	Добавляем вершину f в стек	fba	11	
g	Добавляем вершину g в стек	$gfba$	12	
l	Добавляем вершину l в стек	$lgfba$	13	
g	Удаляем вершину l из стека	$gfba$	14	$T_{out}(l) = 14$
f	Удаляем вершину g из стека	fba	15	$T_{out}(g) = 15$
b	Удаляем вершину f из стека	ba	16	$T_{out}(f) = 16$
a	Удаляем вершину b из стека	a	17	$T_{out}(b) = 17$
	Удаляем вершину a из стека	Стек пуст	18	$T_{out}(a) = 18$
c	Добавляем вершину c в стек	c	19	
	Удаляем вершину c из стека	Стек пуст	20	$T_{out}(c) = 20$

Замечание 2.19

На шаге 18 алгоритма стек стал пустым, но в графе еще осталась не просмотренная вершина (c). Из нее был запущен новый поиск в глубину DFS(c). Серия на шаге 1 состоит из двух поисков.

Согласно таблице 2.7 запишем последовательность вершин в порядке убывания T_{out} :

$$c(20), a(18), b(17), f(16), g(15), l(14), e(10), h(9), d(8), k(7).$$

Транспонируем граф и проводим серию поисков в глубину в порядке уменьшения T_{out} . Имеем:

$$\begin{aligned} c & C_1 = \{c\}, \\ a \rightarrow b & C_2 = \{a, b\}, \\ f \rightarrow l \rightarrow g & C_3 = \{f, l, g\}, \\ e \rightarrow d \rightarrow h \rightarrow k & C_4 = \{e, d, h, k\}. \end{aligned}$$

Получаем граф конденсации исходного графа (рисунок 2.16).

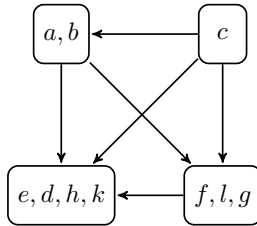


Рис. 2.16.

2.1.5. Эйлеровы графы

Определение 2.18. *Разомкнутой (замкнутой) эйлеровой цепью* графа называют простой разомкнутый (замкнутый) путь, включающий все ребра графа.

Граф, имеющий замкнутую эйлерову цепь, называют *эйлеровым*, а разомкнутую — *полуэйлеровым*.

Теорема 2.16. В графе $G = (V, E)$ существует замкнутая эйлерова цепь тогда и только тогда, когда:

- а) граф связный;
- б) все его вершины имеют четную степень.

Доказательство. Доказательство достаточности условия теоремы будет следствием анализа алгоритма 2.6 нахождения замкнутой эйлеровой цепи, который будет описан далее. Необходимость условия очевидна, так как появление в замкнутой эйлеровой цепи некоторой вершины k раз означает, что степень этой вершины в графе составляет $2k$. ■

Следствие 2.4. Граф $G = (V, E)$ имеет открытую эйлерову цепь тогда и только тогда, когда

- 1) G — связный граф;
- 2) существует ровно две вершины нечетной степени.

Доказательство. Соединив две вершины нечетной степени фиктивным ребром, попадаем в условия теоремы 2.16, следовательно, в новом графе существует замкнутая эйлерова цепь. Удалив фиктивное ребро, получим открытую эйлерову цепь для исходного графа. ■

Как и ранее, вершину стека S будем обозначать как $\text{top}(S)$.

Алгоритм 2.6. Нахождение замкнутой эйлеровой цепи (два стека)

Исходные данные: связный граф $G = (V, E)$ без вершин нечетной степени, представленный списками инцидентности $\text{entry}[v], v \in V$ (см. раздел 2.1.2)

Результат: эйлеров цикл, представленный последовательностью вершин в списке L

```

S ← v0 // записываем в стек S произвольную вершину v0
L ← {∅}
while S ≠ {∅} do
  v ← top(S)
  if entry[v] ≠ {∅} then
    u ← первая вершина списка entry[v]
    S ← u // записываем вершину u в стек S
    entry[v] ← entry[v] − u
    entry[u] ← entry[u] − v // удаляем ребро (v, u) из графа
  else
    Удаляем вершину v из стека S
    L ← v // записываем вершину v в выходной список L
  end if
end while

```

Обоснуем корректность алгоритма 2.6.

Принцип действия алгоритма можно объяснить следующим образом: пусть v_0 — верхний элемент стека, выбранный на шаге S2. Строим путь с началом в v_0 , причем вершины этого пути помещаются в стек S , а ребра удаляются из графа. Эти действия продолжают вплоть до того момента, когда путь нельзя удлинить, включив в него новую вершину, т. е. когда список $\text{entry}[v]$ пуст.

Отметим, что тогда должно быть $v = v_0$, так как в любом другом случае это означало бы, что степень вершины v нечетная. Таким образом, из графа был удален цикл, а вершины этого цикла находятся в стеке S .

Заметим, что в графе, модифицированном таким способом, степень произвольной вершины останется четной. Вершина $v = v_0$ переносится теперь из стека S в список L , а очередной вершиной v становится верхний элемент стека S .

Процесс повторяется, начиная с этой вершины (если текущий список $entry[v]$ не пуст), в результате чего вследствие четности степени всех вершин находится и помещается в стек S некоторый цикл, проходящий через вершину v . Это продолжается до того момента, пока стек S не станет пустым.

Очевидно, что вершины, помещаемые в стек S , образуют некоторый путь, причем вершина v переносится в список L только тогда, когда список $entry[v]$ пуст, т. е. когда все ребра, инцидентные с этой вершиной, представлены (парами соседних вершин) в стеке S или в списке L . Отсюда легко следует, что по окончании работы алгоритма список L содержит эйлеров цикл.

Пример 2.14. Рассмотрим работу алгоритма на примере графа на рисунке 2.17а. Заметим, что граф удовлетворяет условиям теоремы 2.16. Будем считать, что в каждом списке инцидентности $entry[v]$, $v \in \mathbf{V}$, вершины расположены по возрастанию номеров.

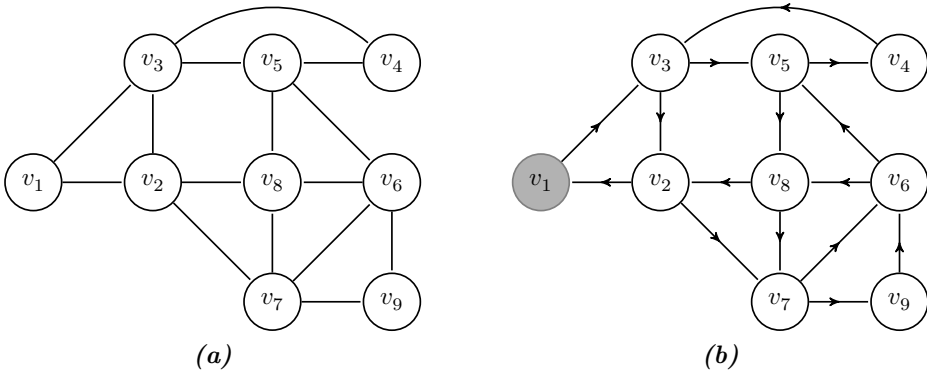


Рис. 2.17.

Начнем работу алгоритма с вершины v_1 . Как и ранее, будем схематично изображать стек развернутым по горизонтали. Считаем, что вершина стека находится слева. Дальнейшие шаги представлены в таблице 2.8.

Таблица 2.8.

Состояние стека S	Список L	Текущее действие
v_1		Добавляем вершину v_1 в стек S
$v_2 v_1$		Добавляем вершину v_2 в стек S

Продолжение таблицы 2.8

Состояние стека S	Список L	Текущее действие
$v_3 v_2 v_1$		Добавляем вершину v_3 в стек S
$v_1 v_3 v_2 v_1$		Добавляем вершину v_1 в стек S
$v_3 v_2 v_1$	v_1	Переносим вершину v_1 из стека S в список L
$v_4 v_3 v_2 v_1$	v_1	Добавляем вершину v_4 в стек S
$v_5 v_4 v_3 v_2 v_1$	v_1	Добавляем вершину v_5 в стек S
$v_3 v_5 v_4 v_3 v_2 v_1$	v_1	Добавляем вершину v_3 в стек S
$v_5 v_4 v_3 v_2 v_1$	$v_3 v_1$	Переносим вершину v_3 из стека S в список L
$v_6 v_5 v_4 v_3 v_2 v_1$	$v_3 v_1$	Добавляем вершину v_6 в стек S
$v_7 v_6 v_5 v_4 v_3 v_2 v_1$	$v_3 v_1$	Добавляем вершину v_7 в стек S
$v_2 v_7 v_6 v_5 v_4 v_3 v_2 v_1$	$v_3 v_1$	Добавляем вершину v_2 в стек S
$v_8 v_2 v_7 v_6 v_5 v_4 v_3 v_2 v_1$	$v_3 v_1$	Добавляем вершину v_8 в стек S
$v_5 v_8 v_2 v_7 v_6 v_5 v_4 v_3 v_2 v_1$	$v_3 v_1$	Добавляем вершину v_5 в стек S
$v_8 v_2 v_7 v_6 v_5 v_4 v_3 v_2 v_1$	$v_5 v_3 v_1$	Переносим вершину v_5 из стека S в список L
$v_6 v_8 v_2 v_7 v_6 v_5 v_4 v_3 v_2 v_1$	$v_5 v_3 v_1$	Добавляем вершину v_6 в стек S
$v_9 v_6 v_8 v_2 v_7 v_6 v_5 v_4 v_3 v_2 v_1$	$v_5 v_3 v_1$	Добавляем вершину v_9 в стек S
$v_7 v_9 v_6 v_8 v_2 v_7 v_6 v_5 v_4 v_3 v_2 v_1$	$v_5 v_3 v_1$	Добавляем вершину v_7 в стек S
$v_8 v_7 v_9 v_6 v_8 v_2 v_7 v_6 v_5 v_4 v_3 v_2 v_1$	$v_5 v_3 v_1$	Добавляем вершину v_8 в стек S
$v_7 v_9 v_6 v_8 v_2 v_7 v_6 v_5 v_4 v_3 v_2 v_1$	$v_8 v_5 v_3 v_1$	Переносим вершину v_8 из стека S в список L

Далее поочередно все оставшиеся вершины из стека S будут перенесены в список L . Получившаяся эйлерова цепь:

$$v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_8 \rightarrow v_7 \rightarrow v_9 \rightarrow v_6 \rightarrow v_8 \rightarrow v_2 \rightarrow v_7 \rightarrow v_6 \rightarrow v_5 \rightarrow v_4 \rightarrow \\ \rightarrow v_3 \rightarrow v_2 \rightarrow v_1$$

указана направлениями на ребрах на рисунке 2.17b.

Историческая справка

Задача существования эйлерова пути в заданном графе была решена Леонардом Эйлером в 1736 г., и представленное им необходимое и достаточное условие существования такого пути считается первой в истории теоремой теории графов.

На рисунке 2.18а изображена старинная схема мостов города Кенигсберга через реку Прегель. На берегах реки находятся части города Альтштадт и Форштадт, на островах — Кнайпхоф и Ломзе. Можно ли прогуляться по городу, пройдя ровно один

раз по каждому мосту? Размышления над этой задачей привели Леонарда Эйлера к открытию критерия существования пути в графе, впоследствии названного его именем.

Если части города считать вершинами графа (A — Альтштадт, B — Кнайпхоф, C — Ломзе, D — Форштадт), а каждый мост — ребром, то схему можно представить в виде мультиграфа (2-графа), как показано на рисунке 2.18b. Заметим, что теорема 2.16 справедлива и для мультиграфов. Таким образом, ответ в задаче о кенигсбергских мостах отрицательный.

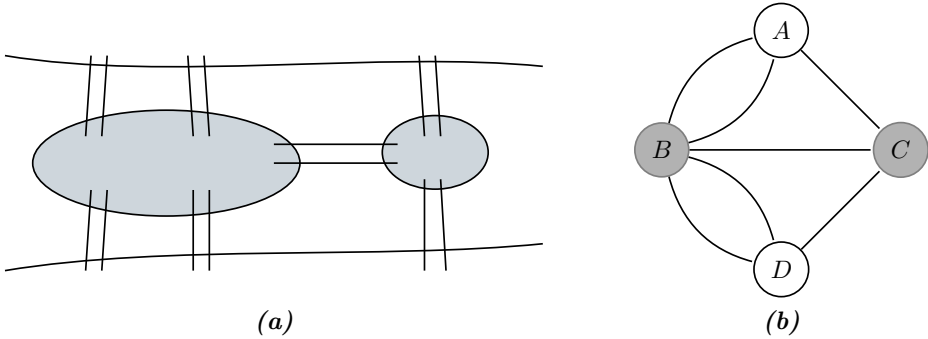


Рис. 2.18.

Рассмотрим еще один алгоритм построения замкнутой эйлеровой цепи. Представление входных и выходных данных такое же, как в алгоритме 2.6.

Считаем, что имеется процедура, позволяющая для любого ребра графа определить, является ли это ребро мостом. Например, можно удалить ребро (u, v) из графа и проверить, существует ли в оставшемся графе путь из вершины u в вершину v , или воспользоваться алгоритмом 2.4.

Историческая справка

Алгоритм 2.7 был предложен французским математиком Флери в 1883 г. [75] и является, пожалуй, одним из первых алгоритмов для построения эйлеровых цепей.

Алгоритм 2.7. Нахождение замкнутой эйлеровой цепи (Флери)

```

 $u \leftarrow$  произвольная вершина графа
 $L \leftarrow u$  // записываем вершину  $u$  в выходной список  $L$ 
while  $E \neq \{\emptyset\}$  do
     $W \leftarrow \{v \in \text{entry}[u] \mid (u, v) \text{ не является мостом}\}$ 
    if  $W \neq \{\emptyset\}$  then
         $v \leftarrow$  любая вершина из  $W$ 
    else
         $v \leftarrow$  любая вершина из  $\text{entry}[u]$ 
    end if
    // ребро мост выбираем только в том случае, когда нет других возможностей.
     $\text{entry}[v] \leftarrow \text{entry}[v] - u$ 

```

```

entry[u] ← entry[u] - v           // удаляем ребро (v, u) из графа

if entry[u] = ∅ then
    удаляем entry[u]             // удаляем изолированную вершину u из графа
end if
L ← v                             // записываем вершину v в выходной список L
u ← v
end while

```

Корректность алгоритма Флери устанавливает следующая теорема.

Теорема 2.17. Алгоритм 2.7 строит замкнутую эйлерову цепь графа при выполнении условий теоремы 2.16.

Доказательство. Покажем сначала, что алгоритм корректно работает на каждом шаге.

Пусть мы достигли некоторой вершины v , начав с вершины $u, v \neq u$. Удалив ребра пути из v в u , видим, что оставшийся граф G_1 связан и содержит ровно две вершины нечетной степени v и u . Согласно следствию 2.4 из теоремы 2.16 граф G_1 имеет открытую эйлерову цепь P из v в u .

Поскольку удаление первого ребра, инцидентного u , цепи P либо не нарушает связности графа G_1 , либо происходит удаление изолированной вершины u и оставшийся граф G_2 связан с двумя нечетными вершинами, то отсюда получаем, что описанное в алгоритме 2.7 действие всегда возможно на каждом шаге. (Если $v = u$, то доказательство не меняется, если имеются ребра, инцидентные u .)

Покажем, что данная последовательность шагов приводит к эйлерову пути. Действительно, в G не может быть ребер, оставшихся непройденными после использования последнего ребра, инцидентного u , поскольку в противном случае удаление ребра, смежного одному из оставшихся, привело бы к несвязному графу, что противоречит условию. ■

Пример 2.15. Рассмотрим работу алгоритма 2.7 для графа из примера 2.14 (см. рисунок 2.17).

Начнем работу с вершины v_1 . При этом условимся выбирать из возможных вершину с наименьшим номером. Протокол работы алгоритма запишем в виде таблицы 2.9).

Таблица 2.9.

Выбираемая вершина	Комментарий
v_2	

Продолжение таблицы 2.9

Выбираемая вершина	Комментарий
v_3	
v_4	Вершину v_1 выбрать нельзя, так как ребро (v_3, v_1) является мостом
v_5	Ребро (v_4, v_5) является мостом, но других ребер из вершины v_4 уже нет. Удаляем изолированную вершину v_4
v_6	Вершину v_3 выбрать нельзя, так как ребро (v_5, v_3) является мостом
v_7	
v_2	
v_8	Ребро (v_2, v_8) является мостом, но других ребер из вершины v_2 уже нет. Удаляем изолированную вершину v_2
v_6	Вершину v_5 выбрать нельзя, так как ребро (v_8, v_5) является мостом. Далее все оставшиеся ребра являются мостами
v_9	Удаляем изолированную вершину v_6
v_7	Удаляем изолированную вершину v_9
v_8	Удаляем изолированную вершину v_7
v_5	Удаляем изолированную вершину v_8
v_3	Удаляем изолированную вершину v_5
v_1	Удаляем изолированную вершину v_3

Таким образом получаем тот же эйлеров цикл, что и в примере 2.14.

Пример 2.16. Рассмотрим еще один пример работы алгоритма 2.7 для графа на рисунке 2.19а. Очевидно, что граф удовлетворяет условиям теоремы 2.16.

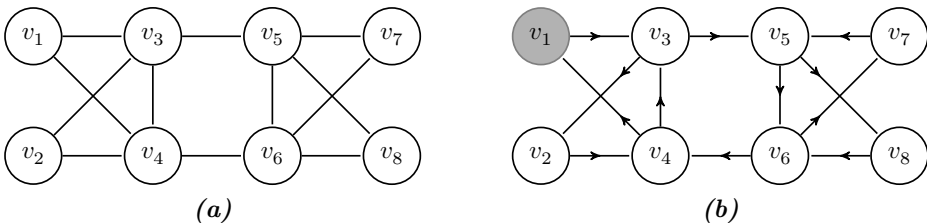


Рис. 2.19.

Протокол работы алгоритма запишем в виде таблицы 2.10.

Таблица 2.10.

Выбираемая вершина	Комментарий
v_3	
v_2	
v_4	Ребро (v_2, v_4) является мостом, но других ребер из вершины v_2 уже нет. Удаляем изолированную вершину v_2
v_3	Вершину v_1 выбрать нельзя, так как ребро (v_4, v_1) является мостом
v_5	Ребро (v_3, v_5) является мостом, но других ребер из вершины v_3 уже нет. Удаляем изолированную вершину v_3
v_6	
v_7	Вершину v_4 выбрать нельзя, так как ребро (v_6, v_4) является мостом. Далее все оставшиеся ребра являются мостами
v_5	Удаляем изолированную вершину v_7
v_8	Удаляем изолированную вершину v_5
v_6	Удаляем изолированную вершину v_8
v_4	Удаляем изолированную вершину v_6
v_1	Удаляем изолированную вершину v_4

Получившаяся эйлерова цепь:

$$v_1 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5 \rightarrow v_8 \rightarrow v_6 \rightarrow v_4 \rightarrow v_1$$

указана направлениями на ребрах на рисунке 2.19b.

Задача 2.1. Классическая задача на построение эйлерова цикла — задача о домино.

Имеется N костяшек домино, на двух концах каждой костяшки записано по одному числу (от 1 до n). Требуется выложить все домино в ряд так, чтобы у любых двух соседних домино числа, записанные на их общей стороне, совпадали. Домино разрешается переворачивать.

Переформулируем задачу. Пусть числа, записанные на домино, — вершины графа, а сами костяшки — ребра этого графа (каждая домино с числами (a, b) — это ребра (a, b) и (b, a)). Тогда задача сводится к нахождению эйлерова пути в этом графе.

Упражнение 2.4. Постройте эйлеров цикл в графе на рисунке 2.20.

Для ориентированных графов понятие эйлеровой цепи вводится аналогично определению 2.18.

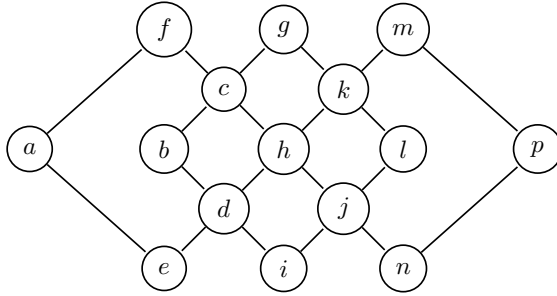


Рис. 2.20.

Определение 2.19. *Разомкнутой (замкнутой) эйлеровой цепью* орграфа называют разомкнутый (замкнутый) простой ориентированный путь, содержащий все дуги графа.

Справедлива также теорема, аналогичная теореме 2.16.

Теорема 2.18. Ориентированный граф $G = (V, E)$ имеет замкнутую эйлерову цепь тогда и только тогда, когда он слабосвязный и

$$|\delta^+(v)| = |\delta^-(v)|, \forall v \in V.$$

Следствие 2.5. Ориентированный граф $G = (V, E)$ имеет разомкнутую эйлерову цепь тогда и только тогда, когда он слабосвязный и выполнены условия:

- 1) существует и притом единственный узел $v_0 \in V$:

$$|\delta^+(v_0)| = |\delta^-(v_0)| + 1;$$

- 2) существует и притом единственный узел $v_1 \in V$:

$$|\delta^-(v_1)| = |\delta^+(v_1)| + 1;$$

- 3) для всех остальных узлов графа выполнено $|\delta^+(v)| = |\delta^-(v)|$.

Доказательство. Добавляем фиктивную дугу $e = [v_1, v_0]$ и применяем теорему 2.18. ■

 **Замечание 2.20**

Подробно различные задачи по эйлеровым графам рассмотрены в монографии Герберта Фляйшнера [64].

2.1.6. Графы де Брюина

Рассмотрим несколько интересных и важных применений эйлеровых путей. Дадим необходимые определения.

Определение 2.20. Конечное множество символов $\mathcal{A} = \{\alpha_1, \dots, \alpha_n\}$ будем называть *алфавитом*, символы α_i — *буквами*, последовательность символов — *словом*. Число символов в слове w называют *длиной слова* и обозначают $|w|$. Иногда слова называют *цепочками*.

Множество слов алфавита \mathcal{A} обозначают \mathcal{A}^* .

Определение 2.21. Пусть задан алфавит \mathcal{A} , состоящий из n букв.

Графом де Брюина (обозначается $B(n, k)$) для алфавита \mathcal{A} называется ориентированный граф $\mathbf{G} = \mathbf{G}(\mathbf{V}, \mathbf{E})$. Его узлы \mathbf{V} есть множество слов длиной k из \mathcal{A}^* . Дуга $[u, v] \in \mathbf{E}$ из узла $u = \{x_1 \dots x_k\}$ в узел $v = \{y_1 \dots y_k\}$ проводится, если $x_i = y_{i-1}, i \in 2 : k$.

Дуге $[u, v]$ сопоставляют слово w длиной $k + 1$:

$$w = \{x_1 \dots x_k y_k\} = \{x_1 y_1 \dots y_k\}.$$

Узел u называют *префиксом*, а узел v — *суффиксом* слова w , символически это можно записать:

$$w = \{u\} + y_k = x_1 + \{v\}. \quad (2.3)$$

Очевидно, что разным дугам $[u_1, v_1]$ и $[u_2, v_2]$ соответствуют два различных слова w_1 и w_2 . То есть кратных дуг в графе де Брюина нет, но очевидным образом есть петли.

Замечание 2.21

Графы названы по имени голландского математика Николаса де Брюина [72]. Иногда в литературе встречаются транскрипции «де Бройна» или «де Брёйна».

Рассмотрим некоторые свойства графов де Брюина.

Лемма 2.1. Граф $B(n, k)$ является эйлеровым графом.

Доказательство. В соответствии с теоремой 2.18 нужно доказать, что для каждого узла число входящих в него ребер равно числу исходящих. Покажем, что

$$|\delta^+(v)| = n = |\delta^-(v)|, \quad \forall v \in \mathbf{V}. \quad (2.4)$$

Согласно (2.3) можно выбрать ровно n символов из \mathcal{A} для исходящих из узла u дуг и столько же для входящих. Равенство (2.4) установлено. ■

Лемма 2.2. Число узлов графа $B(n, k)$ равно n^k , число дуг — соответственно n^{k+1} .

Доказательство. Первое утверждение очевидно из определения графа де Брюина. Из теоремы 2.6 о рукопожатиях для ориентированных графов и равенства (2.4) получаем второе утверждение. ■

Пусть у нас есть алфавит $\mathcal{A} = \{\alpha_1, \dots, \alpha_n\}$ и задано число k . Рассмотрим алгоритм построения соответствующего графа де Брюина.

Алгоритм 2.8. Построение графа де Брюина

Генерируем все слова w длиной $k + 1$ из \mathcal{A}

// Генерация в любом порядке, например в лексикографическом, по алгоритму 1.27

for each $w = \{x_1 \dots x_k x_{k+1}\} \in \mathcal{A}$ **do**

 определяем пару узлов u, v и дугу $[u, v]$ графа

$u = \{x_1 \dots x_k\}, v = \{x_2 \dots x_{k+1}\}$

end for

Пример 2.17. Пусть $\mathcal{A} = \{a, b\}$ и $k = 2$. Протокол работы алгоритма 2.8 приведен в таблице 2.11.

Таблица 2.11.

Слово	Узлы	Слово	Узлы
{aaa}	{aa} → {aa}	{baa}	{ba} → {aa}
{aab}	{aa} → {ab}	{bab}	{ba} → {ab}
{aba}	{ab} → {ba}	{bba}	{bb} → {ba}
{abb}	{ab} → {bb}	{bbb}	{bb} → {bb}

Получившийся граф представлен на рисунке 2.21.

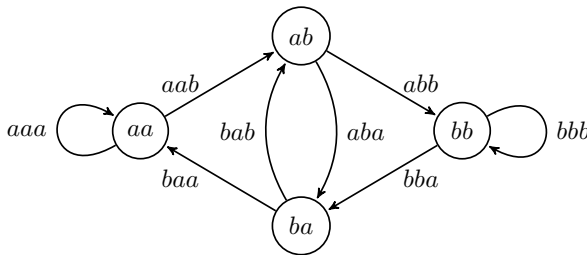


Рис. 2.21.

Рассмотрим несколько применений графов де Брюина¹.

¹ Примеры взяты с сайта: <http://neerc.ifmo.ru/wiki>.

Пример 2.18. Пусть задан алфавит $\mathcal{A} = \{\alpha_1, \dots, \alpha_n\}$. Неизвестное слово-пароль имеет вид $\{x_1 \dots x_k\} \in \mathcal{A}^*$. Найти самое короткое слово s , которое гарантированно содержит пароль как подслово.

Построим граф де Брюина $B(n, k - 1)$. Напомним, что узлы графа — слова длиной $k - 1$ из \mathcal{A}^* . В этом графе найдем эйлеров цикл. Согласно лемме 2.1 он существует.

Изначально слово s пусто, $|s| = 0$. Возьмем произвольный узел эйлерова цикла v и добавим его слово ($k - 1$ символ) в s . Далее, переходя по дугам к следующим узлам, будем добавлять в s последнюю букву слова узла.

Согласно лемме 2.2 в графе n^k дуг, поэтому длина искомой последовательности $|s| = n^k + k - 1$.

Установим корректность решения. Очевидно, что слово меньшей длины составить нельзя: слово s содержит n^k подстрок длиной k , именно столько слов такой длины можно составить из символов \mathcal{A} . Поскольку разным дугам соответствуют разные слова длиной k , получаем корректность построения.

Рассмотрим задачу в условиях примера 2.17. Построим эйлеров цикл (с учетом петель) для графа на рисунке 2.21:

$$aa \rightarrow aa \rightarrow ab \rightarrow ba \rightarrow ab \rightarrow bb \rightarrow bb \rightarrow ba \rightarrow aa.$$

Тогда искомое слово:

$$s = aa + a + b + a + b + b + b + a + a = a a a b a b b b a a.$$

Пример 2.19. Решим задачу построения кратчайшей битовой последовательности, в которой встречаются все бинарные строки длиной 4.

Для алфавита $\mathcal{A} = \{0, 1\}$ построим граф де Брюина $B(2, 3)$. Аналогично примеру 2.17 строим таблицу 2.12.

На дуге из узла префикса в узел суффикса записываем последний символ последовательности. Получившийся граф представлен на рисунке 2.22.

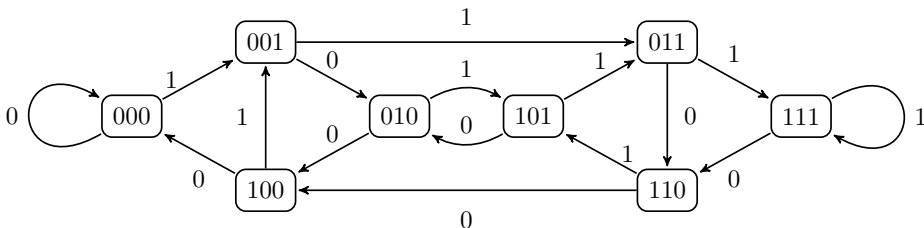


Рис. 2.22.

Таблица 2.12.

Строка	Узлы	Строка	Узлы
{0000}	{000} → {000}	{1000}	{100} → {000}
{0001}	{000} → {001}	{1001}	{100} → {001}
{0010}	{001} → {010}	{1010}	{101} → {010}
{0011}	{001} → {011}	{1011}	{101} → {011}
{0100}	{010} → {100}	{1100}	{110} → {100}
{0101}	{010} → {101}	{1101}	{110} → {101}
{0110}	{011} → {110}	{1110}	{111} → {110}
{0111}	{011} → {111}	{1111}	{111} → {111}

Находим эйлеров цикл в графе, начиная, например, с вершины 000:

$$000 \rightarrow 000 \rightarrow 001 \rightarrow 010 \rightarrow 101 \rightarrow 010 \rightarrow 100 \rightarrow 001 \rightarrow 011 \rightarrow 110 \rightarrow \\ \rightarrow 101 \rightarrow 011 \rightarrow 111 \rightarrow 111 \rightarrow 110 \rightarrow 100 \rightarrow 000.$$

Тогда соответствующая последовательность:

$$000+0+1+0+1+0+0+1+1+0+1+1+1+1+0+0+0 = 0000101001101111000$$

содержит все бинарные строки длиной 4.

Рассмотрим задачу построения генома (в упрощенном варианте). Дадим соответствующие определения.

Определение 2.22. Алфавит состоит из 4 символов $\mathcal{A} = \{A, G, C, T\}$, символы также называются *нуклеотидами*.

Геном и его части — *риды* — представляют собой слова из алфавита \mathcal{A}^* .

Пример 2.20. Задано множество R неповторяющихся ридов длиной k . Известно, что все риды генома длиной k входят в данное множество. Построить возможный геном.

Используем результаты предыдущего примера 2.18.

Построим граф \mathbf{G} , узлами которого будут суффиксы и префиксы длиной $k-1$ всех ридов из R . Получим подграф графа де Брюина $B(4, k-1)$, в котором каждой дуге соответствует рид из R . Это подграф, так как в нем не обязательно есть все 4^{k-1} узлов.

Построим в G эйлеров путь. Он существует, так как на геном было наложено условие о том, что все его подстроки длиной k входят в множество ридов R . Этот путь и будет *возможным* ответом.

Возможным потому, что единственно верный ответ (реальный геном) получить можно не всегда, так как не всегда в графе есть единственный эйлеров путь.

Пусть $k = 3$ и $R = \{ACT, CTG, TGA, GAC, ACG, CGA\}$. Построим возможный геном.

Строим граф G — подграф графа де Брюина $B(4, 2)$ (рисунок 2.23).

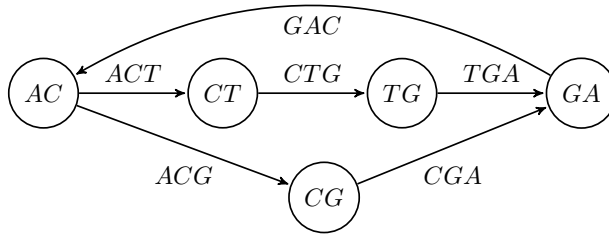


Рис. 2.23.

Строим в графе G эйлеров путь. Условия следствия 2.5 к теореме 2.18 выполнены:

$$|\delta^+(AC)| = 2 = |\delta^-(AC)| + 1, \quad |\delta^-(GA)| = |\delta^+(GA)| + 1,$$

$$|\delta^+(v)| = |\delta^-(v)| = 1, \quad v = CT, TG, CG.$$

Тогда эйлеров путь из узла AC в узел GA имеет вид:

$$AC \rightarrow CT \rightarrow TG \rightarrow GA \rightarrow AC \rightarrow CG \rightarrow GA.$$

Искомый геном: $AC + T + G + A + C + G + A = ACTGACGA$.

Замечание 2.22

Задача построения генома в примере 2.20 существенно упрощена.

1. Риды не обязательно будут повторяющимися.
2. Данные о ридах могут быть получены лишь с некоторой вероятностью.
3. Риды не имеют фиксированной длины в силу особенностей метода их определения (секвенирования).

Тем не менее рассмотренный подход используется при сборке генома или его больших ридов, но с заметными усложнениями.

2.1.7. Гамильтоновы графы

Определение 2.23. Если граф имеет простой цикл, содержащий все вершины графа, то такой цикл называется *гамильтоновым циклом*, а граф называется *гамильтоновым графом*. Граф, который содержит простую цепь, проходящую через каждую его вершину, называется *полугамильтоновым*. Это определение можно распространить на ориентированные графы, если считать цепь ориентированной.

Историческая справка

Слово «гамильтонов» в этом определении связано с именем известного ирландского математика У. Гамильтона, который в 1859 г. предложил игру «Кругосветное путешествие». Каждой из 20 вершин додекаэдра приписано название одного из крупных городов мира. Требуется, переходя от одного города к другому по ребрам додекаэдра, посетить каждый город в точности один раз и вернуться в исходный город.

Гамильтонов цикл не обязательно содержит все ребра графа. Ясно, что гамильтоновым может быть только связный граф и что всякий гамильтонов граф является полугамильтоновым. Заметим, что гамильтонов цикл существует далеко не в каждом графе.

Для гамильтоновых циклов (и путей) неизвестно никаких просто проверяемых необходимых и достаточных условий их существования, а все известные алгоритмы требуют для некоторых графов перебора большого числа вариантов. В отличие от эйлеровых графов, где имеется необходимый и достаточный критерий для графа быть эйлеровым, для гамильтоновых графов такого критерия нет. Большинство известных теорем имеет вид: *если граф G имеет достаточное количество ребер, то граф является гамильтоновым*.

Приведем несколько таких теорем.

Теорема 2.19 (Оре). Если для любой пары u и v несмежных вершин графа G порядка $n \geq 3$ выполняется неравенство

$$\deg(u) + \deg(v) \geq n, \quad (2.5)$$

то G — гамильтонов граф.

Доказательство. Предположим, что существует граф G , который удовлетворяет (2.5), но не является гамильтоновым графом.

Будем добавлять к нему новые ребра, не образующие гамильтонов цикл, до тех пор, пока есть возможность такие ребра добавлять. Обозначим полученный негамильтонов граф G_1 . Заметим, что такой граф существует, так последним в такой последовательности действий является полный граф,

который, разумеется, гамильтонов. Поскольку мы только добавляли ребра, условие (2.5) для графа G_1 не нарушилось.

Рассмотри две несмежные вершины u_s и u_f графа G_1 . Такая пара вершин существует, иначе граф G_1 был бы полным. В силу максимальности графа G_1 при добавлении к нему ребра (u_s, u_f) получим гамильтонов цикл. Тогда путь $u_s \rightarrow u_f$ является гамильтоновым путем. Пусть

$$u_s = v_1 \rightarrow \dots \rightarrow v_i \rightarrow v_{i+1} \rightarrow \dots \rightarrow v_n = u_f.$$

Для каждого индекса i ($2 \leq i \leq n$) рассмотрим пару «возможных» ребер (u_s, v_{i+1}) и (v_i, u_f) . «Возможных», так как в графе G_1 не может одновременно присутствовать ни одна такая пара ребер. Иначе в графе G_1 существовал бы гамильтонов цикл:

$$u_s \rightarrow v_2 \rightarrow \dots \rightarrow v_i \rightarrow u_f \rightarrow v_{n-1} \rightarrow \dots \rightarrow v_{i+1} \rightarrow u_s.$$

Таким образом,

$$\deg(u_s) + \deg(u_f) \leq \max i = n - 1$$

для графа G_1 и соответственно для графа G (при построении G_1 валентности вершин не уменьшались). Противоречие с (2.5). ■

Теорема Оре является обобщением следующей теоремы Дирака.

Теорема 2.20 (Дирак). Если для любой вершины v графа G порядка $n \geq 3$ выполняется неравенство $\deg(v) \geq n/2$, то G — гамильтонов граф.

Доказательство. Очевидно, что из условия теоремы следует неравенство (2.5). ■

Историческая справка

Теорема 2.19 предложена в 1960 г. норвежским математиком Ойстином Оре — автором одного из классических трудов по теории графов [46], имеющих в мировой литературе.

Теорема 2.20 доказана в 1952 г. венгерско-британским математиком Габриэлем Эндрю Дираком и является первым достаточным условием гамильтоновости графа.

Определение 2.24. Графы, удовлетворяющие условию теоремы 2.19, называют *графами Оре*, а графы, удовлетворяющие теореме 2.20, — *графами Дирака*.

Нетрудно заметить, что любой граф Дирака является графом Оре. Обратное не всегда верно.

Пример 2.21. В графе G_a , изображенном на рисунке 2.24а, есть гамильтонов цикл, например последовательность вершин $v_1, v_2, v_3, v_5, v_4, v_1$.

В графе G_b на рисунке 2.24б нет гамильтоновых циклов, но есть гамильтоновы пути например v_2, v_1, v_3, v_5, v_4 .

В графе G_c на рисунке 2.24с нет и гамильтоновых путей.

Заметим, что граф G_a является графом Оре, а графы G_b и G_c нет. Все три графа не являются графами Дирака.

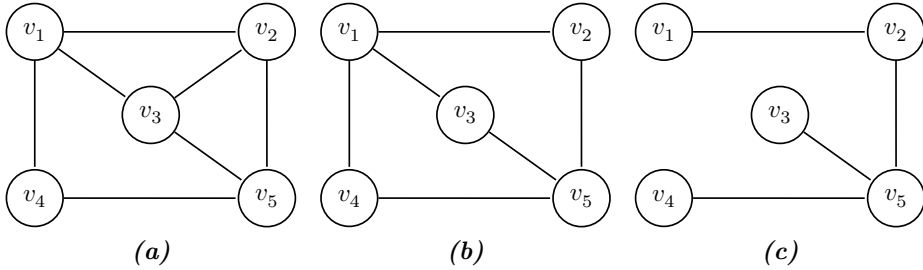


Рис. 2.24.

К сожалению, теоремы Оре и Дирака дают только достаточные условия существования гамильтонова цикла.

На рисунке 2.25 изображен гамильтонов граф, не удовлетворяющий условию теоремы Оре ни для одной пары несмежных вершин. Однако гамильтонов цикл (и не один) в нем существует, например (2.6). Путь в графе отмечен стрелками (начиная с вершины v_1).

$$v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_3 \rightarrow v_6 \rightarrow v_7 \rightarrow v_1. \quad (2.6)$$

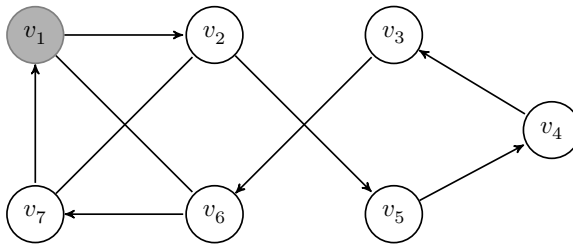


Рис. 2.25.

Рассмотрим алгоритм построения гамильтонова цикла в условиях теоремы Оре или теоремы Дирака.

Введем обозначения: $\{v_1, \dots, v_n\}$ — все вершины графа $G = G(V, E)$, Q — очередь вершин, пуста в начале работы алгоритма. Через u_i обозначим вершину, находящуюся в данный момент на позиции i в очереди Q .

Нам потребуется понятие *реверса* части очереди.

Определение 2.25. Пусть текущая очередь вершин Q имеет вид:

$$Q = u_1, \dots, u_{i-1}, \underbrace{u_i, u_{i+1}, \dots, u_{i+k-1}, u_{i+k}}_{k+1}, u_{i+k+1}, \dots, u_n, \quad k \geq 1.$$

Под записью в обратном порядке (*реверсе*) части очереди от элемента u_i до элемента u_{i+k} (обозначим $\text{rev}(u_i, u_{i+k})$) будем понимать следующее преобразование очереди Q :

$$Q = u_1, \dots, u_{i-1}, \underbrace{u_{i+k}, u_{i+k-1}, \dots, u_{i+1}, u_i}_{k+1}, u_{i+k+1}, \dots, u_n, \quad k \geq 1.$$

Замечание 2.23

| При $k = 1$ реверс означает перестановку в очереди двух соседних вершин u_i и u_{i+1} .

Алгоритм 2.9. Нахождение гамильтонова цикла

```

 $Q \leftarrow v_1, \dots, v_n$  // записываем все вершины в очередь, без учета смежности
 $iter \leftarrow 1$ 
while  $iter \leq n$  do
  if  $(u_1, u_2) \notin \mathbf{E}$  then // вершины  $u_1$  и  $u_2$  несмежные
    находим  $i > 2 \mid (u_1, u_i), (u_2, u_{i+1}) \in \mathbf{E}$  // существование  $i$  будет обосновано
    ниже
     $\text{rev}(u_2, u_i)$  // часть очереди  $Q$  от  $u_2$  до  $u_i$  записываем в обратном порядке
  end if
  перемещаем  $u_1$  в конец очереди  $Q$ 
   $iter \leftarrow iter + 1$ 
end while

```

Проведем обоснование алгоритма 2.9.

После n итераций в очереди Q находятся все вершины графа по одному разу, любые две соседние вершины соединены ребрами, и существует ребро между последней и первой вершинами очереди. Таким образом получаем гамильтонов цикл.

Обоснуем существование индекса i для двух несвязных вершин u_1 и u_2 .

Введем следующие обозначения. Пусть $I_1 = \{i \mid (u_1, v_i) \in \mathbf{E}\}$ — множество индексов вершин, смежных с u_1 , а $I_2 = \{i + 1 \mid (u_2, v_{i+1}) \in \mathbf{E}\}$ — множество индексов вершин, смежных с u_2 .

Заметим, что $I_1 \subseteq \{3, 4, \dots, n\}$, а $I_2 \subseteq \{2, 3, \dots, n - 1\}$. Тогда

$$I_1 \cup I_2 \subseteq \{2, 3, \dots, n\}, \quad |I_1 \cup I_2| \leq n - 1.$$

Но по условию теоремы Оре

$$|I_1| + |I_2| = \text{deg}(u_1) + \text{deg}(u_2) \geq n.$$

Следовательно, $I_1 \cap I_2 \neq \emptyset$, что и доказывает существование искомого индекса i .

Пример 2.22. Рассмотрим работу алгоритма 2.9 на примере графа G_a (рисунок 2.24а).

Заметим, что условие (2.5) теоремы Оре для несмежных вершин этого графа выполнено. Действительно:

$$\deg(v_4) = 2, \deg(v_i) = 3, i = 1, 2, 3, 5.$$

Протокол работы алгоритма приведен в таблице 2.13.

Таблица 2.13.

<i>iter</i>	Очередь	Действия
	v_1, v_2, v_3, v_4, v_5	
1	v_2, v_3, v_4, v_5, v_1	$(v_1, v_2) \in E, v_1 \rightarrow$ конец очереди
2	v_3, v_4, v_5, v_1, v_2	$(v_2, v_3) \in E, v_2 \rightarrow$ конец очереди
3	v_5, v_4, v_1, v_2, v_3	$(v_3, v_4) \notin E$, находим $i = 3 : (v_3, v_5), (v_4, v_1) \in E$ $\text{rev}(v_4, v_5), v_3 \rightarrow$ конец очереди
4	v_4, v_1, v_2, v_3, v_5	$(v_5, v_4) \in E, v_5 \rightarrow$ конец очереди
5	v_1, v_2, v_3, v_5, v_4	$(v_4, v_1) \in E, v_4 \rightarrow$ конец очереди

Получаем гамильтонов цикл:

$$v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_5 \rightarrow v_4 \rightarrow v_1.$$

Пример 2.23. С помощью алгоритма 2.9 построим гамильтонов цикл для графа на рисунке 2.26.

Заметим, что условие (2.5) теоремы Оре для несмежных вершин этого графа выполнено. Действительно:

$$\deg(v_3) = \deg(v_4) = 3, \deg(v_i) = 4, i = 1, 2, 5, 6, 7.$$

Протокол работы алгоритма запишем в таблице 2.14.

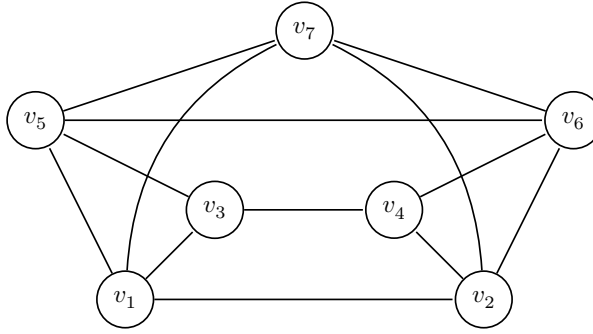


Рис. 2.26.

Таблица 2.14.

<i>iter</i>	Очередь	Действия
	$v_1, v_2, v_3, v_4, v_5, v_6, v_7$	
1	$v_2, v_3, v_4, v_5, v_6, v_7, v_1$	$(v_1, v_2) \in \mathbf{E}$, $v_1 \rightarrow$ конец очереди
2	$v_4, v_3, v_5, v_6, v_7, v_1, v_2$	$(v_2, v_3) \notin \mathbf{E}$, находим $i = 3 : (v_2, v_4), (v_3, v_5) \in \mathbf{E}$ $\text{rev}(v_3, v_4)$, $v_2 \rightarrow$ конец очереди
3	$v_3, v_5, v_6, v_7, v_1, v_2, v_4$	$(v_4, v_3) \in \mathbf{E}$, $v_4 \rightarrow$ конец очереди
4	$v_5, v_6, v_7, v_1, v_2, v_4, v_3$	$(v_3, v_5) \in \mathbf{E}$, $v_3 \rightarrow$ конец очереди
5	$v_6, v_7, v_1, v_2, v_4, v_3, v_5$	$(v_5, v_6) \in \mathbf{E}$, $v_5 \rightarrow$ конец очереди
6	$v_7, v_1, v_2, v_4, v_3, v_5, v_6$	$(v_6, v_7) \in \mathbf{E}$, $v_6 \rightarrow$ конец очереди
7	$v_1, v_2, v_4, v_3, v_5, v_6, v_7$	$(v_7, v_1) \in \mathbf{E}$, $v_7 \rightarrow$ конец очереди

Получаем гамильтонов цикл:

$$v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_1.$$

2.1.8. Турниры

Определение 2.26. Турниром будем называть ориентированный граф $G = G(V, E)$, в котором для каждой пары узлов $u, v \in V$ существует ровно одна дуга $[v, u]$ или $[u, v]$. Турнир с n вершинами будем обозначать T_n .

Если рассмотреть турнир без учета ориентации его дуг, то получается полный неориентированный граф (см. определение 2.2).

 **Замечание 2.24**

Название «турнир» происходит из интерпретации графа как кругового соревнования, проходящего в один круг без ничьих. Узлы графа соответствуют участникам турнира, а дуги — результатам встречи.

Дуга проводится от победителя к побежденному, при наличии дуги $[u, v]$ говорят, что u доминирует над v .

Связь турниров и гамильтоновых графов устанавливает следующая теорема.

Теорема 2.21 (Редери — Камиона). В любом турнире существует гамильтонов путь.

Доказательство. Проведем индукцию по числу вершин в турнире.

Очевидно, что для T_3 утверждение верно. База индукции установлена.

Обоснуем индукционный переход. Пусть предположение верно для всех турниров T_n .

Возьмем произвольную вершину $u \in T_{n+1}$ и рассмотрим турнир $T_n = T_{n+1} - \{u\}$. В этом турнире n вершин, следовательно, по индукционному предположению для него существует гамильтонов путь P .

$$P = v_1 \rightarrow \dots \rightarrow v_n, v_i \in T_n, i \in 1 : n.$$

Поскольку T_{n+1} — турнир, он содержит одну из двух дуг: $[u, v_1]$ или $[v_1, u]$. Если $[u, v_1] \in T_{n+1}$, то путь $u \rightarrow P$ — гамильтонов.

Рассмотрим второй случай: $[v_1, u] \in T_{n+1}$. Пусть v_i — первая вершина в пути P , для которой есть дуга $[u, v_i] \in T_{n+1}$. Тогда, очевидно, есть дуга $[v_{i-1}, u]$ (иначе на роль v_i претендовала бы вершина v_{i-1}). Следовательно, путь

$$v_1 \rightarrow \dots \rightarrow v_{i-1} \rightarrow u \rightarrow v_i \rightarrow \dots \rightarrow v_n$$

— гамильтонов путь для турнира T_{n+1} . Если же такой вершины v_i в пути P нет, то есть дуга $[v_n, u]$ и $P \rightarrow u$ гамильтонов путь для турнира T_{n+1} . ■

Теорема Редери — Камиона устанавливает, что любой турнир — полугамильтонов граф. Однако гамильтонов цикл существует не для любого турнира. Пример¹ такого турнира, приведен на рисунке 2.27.

Определение 2.27. Турнир, в котором из существования дуг $[u, v]$ и $[v, w]$ следует существование дуги $[u, w]$, называется *транзитивным*.

Заметим, что турнир на рисунке 2.27 нетранзитивный: есть дуги $[v_4, v_3]$ и $[v_3, v_5]$, но дуги $[v_4, v_5]$ нет.

¹ Пример взят с сайта <http://neerc.ifmo.ru/wiki>.

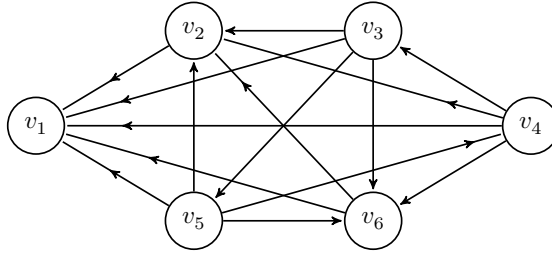


Рис. 2.27.

Игрок, выигравший все игры, будет победителем турнира. Однако, как показывает существование нетранзитивных турниров, такого игрока может не оказаться.

Определение 2.28. Турнир, в котором каждый игрок проигрывает хотя бы одну игру, называется *1-парадоксальным турниром*.

Пример такого турнира приведен на рисунке 2.27.

2.1.9. Деревья

Определение 2.29. Связный граф, не содержащий циклов, называют *деревом*. Произвольный граф, не содержащий циклов, называют *ациклическим*, или *лесом*.

Замечание 2.25

Термин «дерево» был введен в 1857 г. британским математиком Артуром Кэли.

Теорема 2.22. Граф является деревом тогда и только тогда, когда между любыми двумя его различными вершинами существует одна и только одна цепь.

Доказательство. Установим необходимость. Граф G является деревом, следовательно, он связный и любые две его вершины можно соединить цепью. Поскольку граф G не содержит циклов, по следствию 2.2 к теореме 2.5 такая цепь только одна.

Для доказательства достаточности нужно провести те же рассуждения в обратном порядке. ■

Теорема 2.23. Дерево, содержащее не менее двух вершин, имеет по крайней мере две концевые вершины.

Доказательство. Пусть $v_0, e_1, v_1, \dots, e_l, v_l$ — цепь максимальной длины в графе. Покажем, что $\deg(v_0) = \deg(v_l) = 1$.

Пусть существует ребро $e_{l+1} \neq e_l$, также инцидентное v_l , и v_{l+1} — его вторая концевая точка. Но в дереве нет циклов, тогда $v_{l+1} \neq v_i$ ($i = 0, 1, \dots$), следовательно, $v_0, e_1, v_1, \dots, e_l, v_l, e_{l+1}, v_{l+1}$ — цепь. Получаем противоречие максимальности длины цепи.

Аналогичное доказательство можно провести для вершины v_0 . ■

Лемма 2.3. После удаления из дерева концевой вершины вместе с инцидентным ей ребром вновь получается дерево.

Доказательство. Пусть v — концевая вершина, а e — инцидентное ей ребро в дереве $G = (V, E)$. Рассмотрим подграф $G' = (V \setminus \{v\}, E \setminus \{e\})$.

Очевидно, что граф G' не содержит циклов. Любую пару вершин графа G' в исходном графе G можно соединить цепью. Очевидно, что эта цепь не содержит ни вершину v , ни ребро e . Таким образом, граф G' — дерево. ■

Теорема 2.24. Дерево с p вершинами имеет $p - 1$ ребро.

Доказательство. Проводим индукцию по числу вершин p . При $p = 1$ утверждение очевидно. Пусть теорема верна для всех $p \leq k$.

По теореме 2.23 дерево с $k + 1 \geq 2$ вершин имеет концевую вершину. Исключая ее вместе с инцидентным ей ребром, по лемме 2.3 получаем дерево с k вершинами, которое (по индукционному предположению) имеет $k - 1$ ребро. ■

Определение 2.30. Пусть $G = (V, E)$ — связный граф. Дерево, являющееся подграфом G и содержащее все его вершины, называют деревом, *покрывающим* граф G (стягивающим деревом, каркасом, остовом, остовым деревом).

Теорема 2.25. В связном графе G всегда существует по крайней мере одно стягивающее его дерево.

Доказательство. Если в графе G нет циклов, то теорема доказана. В противном случае исключаем любое ребро e , принадлежащее циклу. По теореме 2.11 получившийся граф $G \setminus \{e\}$ связный. Продолжаем эту процедуру до тех пор, пока в графе есть циклы. ■

Следствие 2.6. Пусть F — подмножество ребер графа $G = (V, E)$, обладающее следующим свойством: для любого цикла графа G все ребра этого цикла не лежат полностью в F . Тогда существует стягивающее дерево $G' = (V, E') : F \subseteq E'$.

Доказательство. При доказательстве теоремы 2.25 всегда можно исключать ребра, не лежащие в F , поэтому ребра F останутся в стягивающем дереве. ■

Следствие 2.7. Связный граф с p вершинами и $p-1$ ребрами является деревом.

Доказательство. Если бы граф имел цикл, то при построении его стягивающего дерева получилось бы дерево с p вершинами, а число ребер (после исключения цикла) было бы меньше чем $(p-1)$. Противоречие с теоремой 2.24. ■

Следствие 2.8. Если связный граф имеет p вершин и q ребер, то

$$q - p + 1 \geq 0 \quad (q \geq p - 1).$$

Доказательство. В процессе построения стягивающего дерева (при доказательстве теоремы 2.25) при исключении m циклов ($m \geq 0$) удаляется m ребер. В получившемся стягивающем дереве остается $p-1$ ребро. Таким образом, исходный граф имеет $q = p - 1 + m \geq p - 1$ ребер. ■

Алгоритм 2.10. Построение стягивающего дерева

Воспользуемся алгоритмом поиска в глубину или поиска в ширину. Добавим к нему одну операцию. При нахождении новой *непросмотренной* вершины u из текущей вершины v включаем в дерево ребро (v, u) .

Замечание 2.26

В силу применяемых алгоритмов просматриваются все вершины графа. В полученном графе нет циклов, так как последнее ребро, замыкающее цикл, должно было бы соединить уже просмотренные вершины.

Определение 2.31. Каркас, строящийся поиском в глубину в алгоритме 2.10, называется *DFS-деревом* данного графа.

Пример 2.24. Алгоритмом 2.10 построим стягивающее дерево для графа на рисунке 2.28а.

Для построения в алгоритме 2.10 применим поиск в глубину. Начало работы — с вершины v_1 .

Протокол работы алгоритма запишем в виде таблицы 2.15.

Таблица 2.15.

Состояние стека	Добавляемое ребро	Комментарий
$\{v_1\}$		Добавили в стек вершину v_1
$\{v_2 v_1\}$	(v_1, v_2)	Добавили в стек вершину v_2
$\{v_3 v_2 v_1\}$	(v_2, v_3)	Добавили в стек вершину v_3

Продолжение таблицы 2.15

Состояние стека	Добавляемое ребро	Комментарий
$\{v_4 v_3 v_2 v_1\}$	(v_3, v_4)	Добавили в стек вершину v_4
$\{v_6 v_4 v_3 v_2 v_1\}$	(v_4, v_6)	Добавили в стек вершину v_6
$\{v_5 v_6 v_4 v_3 v_2 v_1\}$	(v_6, v_5)	Добавили в стек вершину v_5
$\{v_8 v_5 v_6 v_4 v_3 v_2 v_1\}$	(v_5, v_8)	Добавили в стек вершину v_8
$\{v_5, v_6, v_4 v_3 v_2 v_1\}$		Удалили из стека вершину v_8
$\{v_9 v_5 v_6 v_4 v_3 v_2 v_1\}$	(v_5, v_9)	Добавили в стек вершину v_9
$\{v_5 v_6, v_4 v_3 v_2 v_1\}$		Удалили из стека вершину v_9
$\{v_6 v_4 v_3 v_2 v_1\}$		Удалили из стека вершину v_5
$\{v_7 v_6 v_4 v_3 v_2 v_1\}$	(v_6, v_7)	Добавили в стек вершину v_7

Удаляем из стека оставшиеся вершины $v_7, v_6, v_4, v_3, v_2, v_1$. Стек пуст, алгоритм заканчивает работу. Ребра, вошедшие в стягивающее дерево, имеют большую «толщину» на рисунке 2.28b.

Упражнение 2.5. Постройте стягивающее дерево для графа из примера 2.24 (см. рисунок 2.28a), используя алгоритм 2.2 с поиском в ширину.

Упражнение 2.6. Предложенный алгоритм дает одно из существующих стягивающих деревьев графа. Модифицируйте алгоритм 2.10 для построения всех стягивающих деревьев данного графа.

2.1.10. Корневые деревья

Сначала введем понятие корневого дерева для неориентированных графов.

Определение 2.32. *Корневым деревом* будем называть дерево с выделенной вершиной — корнем. Висячая вершина корневого дерева, не являющаяся корнем, называется *листом*. *Высота* корневого дерева — это расстояние от корня до самого удаленного листа.

Если в корневом дереве T путь, соединяющий вершину u с корнем, проходит через вершину v , то говорят, что вершина v — *предок* вершины u , а соответственно вершина u — *потомок* вершины v . В частности, каждая вершина является предком и потомком самой себя.

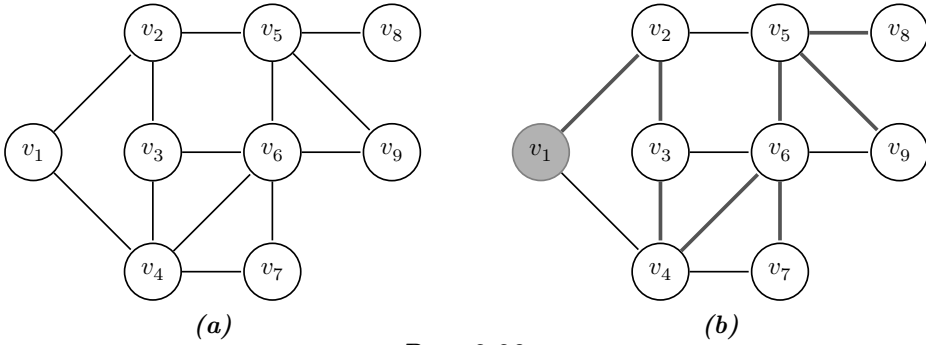


Рис. 2.28.

Множество всех предков вершины v порождает путь из корня в самую вершину v . Множество всех потомков вершины v порождает дерево с корнем v , оно называется *ветвью* дерева T в вершине v .

Если предок и потомок соединены ребром, то они называются соответственно *отцом* и *сыном*.

Замечание 2.27

Иногда термин «корневое дерево» заменяют на «корневое представление дерева».

Для ориентированных графов понятие дерева вводится аналогичным образом.

Определение 2.33. *Ориентированным деревом* называют ориентированный граф без циклов, в котором в каждый узел, кроме одного, называемого *корнем ориентированного дерева*, входит одна дуга. В корень ориентированного дерева не входит ни одной дуги (его отрицательная степень равна 0). При этом для любого узла существует ориентированный путь в него из корня. Иногда, если это не приводит к неоднозначности, ориентированное дерево называют просто деревом.

Любое неориентированное дерево можно представить в виде корневого. Это представление основано на следующей лемме.

Лемма 2.4 (о построении корневого дерева). Множество вершин произвольного дерева $T = (V, E)$ можно разбить в объединение непустых попарно непересекающихся подмножеств V_0, \dots, V_k так, что будут выполнены следующие условия:

- 1) множество V_0 состоит из одной вершины;
- 2) для любого $i \in 1 : k$ никакие вершины из V_i не смежны;
- 3) для любого $i \in 1 : k$ любая вершина из V_i смежна ровно с одной вершиной из V_{i-1} и не смежна ни с одной вершиной из V_{i-2}, \dots, V_0 .

Доказательство. Рассмотрим $v_0 \in V$ — произвольную вершину дерева T . Обозначим $\rho(v_0, v)$ — расстояние между вершинами v_0 и v . Положим

$$m = \max \{ \rho(v_0, v) | v \in V \}$$

и рассмотрим подмножества вершин $V_i \subset V$:

$$V_i = \{ v \in V | \rho(v_0, v) = i \}, \quad i = 0, \dots, m.$$

Очевидно, что никакие из множеств V_i не пересекаются, и $V_0 = \{v_0\}$. Далее, каждая вершина из множества V_i имеет смежную вершину во множестве V_{i-1} .

Действительно, если $\rho(v_0, v) = i$, то расстояние между v_0 и предпоследней вершиной кратчайшей $(v_0 - v)$ -цепи равно $i - 1$. Следовательно, все множества V_i непусты.

Пусть $v \in V_i$. Тогда все смежные с v вершины принадлежат одному из трех множеств V_{i-1}, V_i, V_{i+1} , поскольку расстояния от вершины v до двух смежных вершин не могут отличаться более чем на единицу.

Осталось показать, что у v нет смежной вершины в V_i и нет двух смежных вершин в V_{i-1} . На рисунке 2.29 изображены кратчайшие цепи из рассматриваемых вершин до вершины v_0 для первого случая, а на рисунке 2.30 — соответственно для второго.

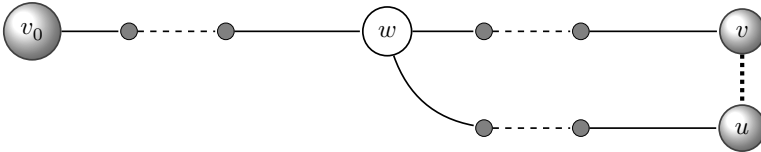


Рис. 2.29.

В первом случае если вершина v имеет смежную вершину $u \in V_i$, то ребро (v, u) очевидно замыкает цикл в дереве T .

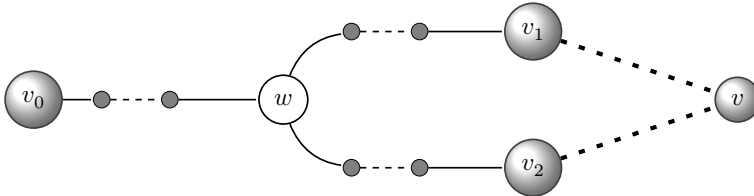


Рис. 2.30.

Для второго случая при наличии у вершины v двух смежных вершин $v_1, v_2 \in V_{i-1}$, ребра (v_1, v) и (v, v_2) замыкают цикл в дереве T . Полученные в обоих случаях противоречия завершают доказательство. ■

Рассмотрим алгоритм преобразования произвольного дерева в корневое.

Пусть $T = (V, E)$ — дерево, а V_0, \dots, V_k — множества вершин, построенные в лемме 2.4, будем считать, что $V_0 = \{v_0\}$.

Помещаем вершину v_0 вверху, вершины из множества V_1 — на одном уровне и ниже, чем v_0 , вершины из множества V_2 — на одном уровне и ниже, чем вершины из V_1 , и т. д. При этом, если на i -м уровне слева направо изображены вершины v_1, v_2, \dots , то на $(i + 1)$ -м уровне слева направо изображаются сначала вершины, смежные v_1 , затем смежные v_2 , и т. д. При таком построении ребра не будут пересекаться. Полученный граф и есть корневое изображение дерева T . Очевидно, что вершина v_0 есть корень построенного дерева.

Замечание 2.28

Одно и то же дерево имеет много корневых изображений, в зависимости от выбора корня (вершины v_0 в рассмотренном алгоритме).

Пример 2.25. Построим корневое дерево для каркаса из примера 2.24, изображенного на рисунке 2.31а.

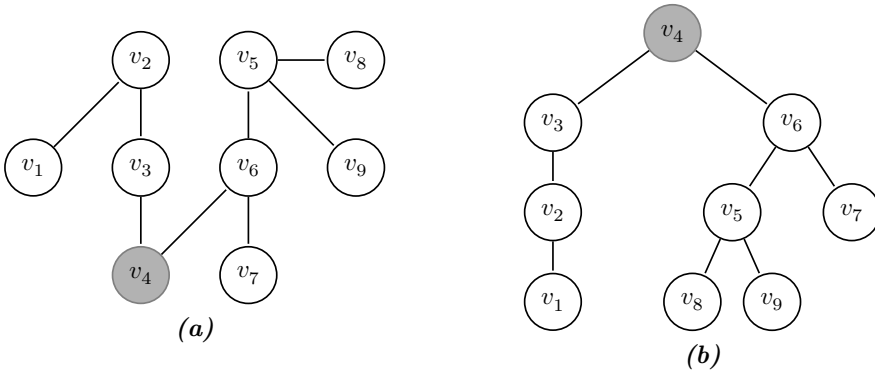


Рис. 2.31.

В качестве корневой вершины выберем v_4 . Получаем следующую последовательность разбиений:

$$V_0 = \{v_4\}, V_1 = \{v_3, v_6\}, V_2 = \{v_2, v_5, v_7\}, V_3 = \{v_1, v_8, v_9\}.$$

На рисунке 2.31b приведено построенное корневое изображение.

В настоящее время корневые деревья применяются очень широко. Например, это деревья вариантов в алгоритмах перебора с возвратом в разделе 2.2.11.

При оценке шахматной позиции (см. задачу 2.6) строится дерево, корнем которого является текущая позиция в игре, сыновьями корня — позиции, возникающие после различных ее ходов, сыновьями сыновей — позиции после ответов противника на эти ходы и т. д. На основе заранее определенной функции оценки позиции *отсекаются* определенные части дерева и выбирается лучший ход.

2.1.11. Код Прюфера

Рассмотрим задачу хранения деревьев. Для кодирования структуры дерева можно использовать *код Прюфера*. Код является оптимальным по объему.

Историческая справка

Код первоначально был предложен немецким математиком (учеником Фробениуса и Шура) Хайнцем Прюфером (1896–1934) в 1918 г. как доказательство формулы Кэли (см. следствие 2.9 к теореме 2.26).

Дадим формальное определение.

Определение 2.34. Кодом Прюфера длиной $n - 2$ называется последовательность из чисел от 1 до n с повторениями.

Теорема 2.26. Существует взаимно однозначное соответствие между стягивающими деревьями для графа из n вершин и кодами Прюфера длиной $n - 2$. По каждому дереву с n вершинами можно построить код Прюфера длиной $n - 2$ и наоборот.

Следствие 2.9 (формула Кэли). Количество пронумерованных деревьев из n вершин равно n^{n-2} .

В качестве доказательства теоремы 2.26 приведем алгоритмы кодирования и восстановления дерева по коду Прюфера.

Алгоритм 2.11. Построение кода Прюфера

Исходные данные: T — дерево с множеством вершин $\{v_1, \dots, v_n\}$. Считаем, что номер вершины v_k равен k .

Результат: последовательность кода Прюфера $P = \{p_1, \dots, p_{n-2}\}$

for $i \leftarrow 1, n - 2$ **do**

$v_k \leftarrow$ висячая вершина с минимальным номером

$p_i \leftarrow s =$ номер вершины v_s — единственного соседа вершины v_k

Удаляем из дерева висячую вершину v_k

end for

Пример 2.26. С помощью алгоритма 2.11 построим код Прюфера для графа на рисунке 2.32.

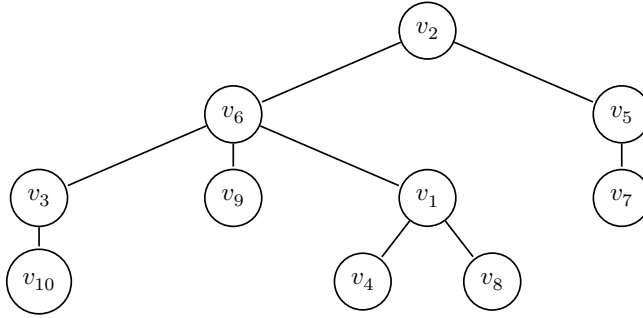


Рис. 2.32.

Протокол работы запишем в таблице 2.16.

Таблица 2.16.

i	Вершина	p_i	i	Вершина	p_i	i	Вершина	p_i	i	Вершина	p_i
1	v_4	1	3	v_5	2	5	v_8	1	7	v_9	6
2	v_7	5	4	v_2	6	6	v_1	6	8	v_6	3

Код Прюфера: $P = \{1, 5, 2, 6, 1, 6, 6, 3\}$.

Замечание 2.29

После завершения алгоритма 2.11 в дереве останутся неудаленными две вершины. Одной из них будет вершина с максимальным номером. Каждая вершина встречается в коде Прюфера определенное число раз, равное ее степени минус один. Это легко понять, если заметить, что вершина удаляется из дерева в момент, когда ее степень равна единице, т. е. к этому моменту все смежные с ней ребра, кроме одного, были удалены. (Для двух оставшихся после построения кода вершин это утверждение тоже верно.)

Рассмотрим алгоритм для решения обратной задачи.

Алгоритм 2.12. Построение дерева по коду Прюфера

Исходные данные: код Прюфера $P = \{p_1, \dots, p_{n-2}\}$, $L = \{1, \dots, n\}$

Результат: дерево T с n вершинами, пронумерованными от 1 до n

// Как и в алгоритме 2.11, считаем, что номер вершины v_k равен k

for $i \leftarrow 1, |P| = n - 2$ **do**

$k \leftarrow \min \{j \mid j \in L, j \notin P\}$

Соединяем вершины v_k и v_{p_i}

Удаляем элемент k из L .

Удаляем элемент p_i из P , $P = \{p_{i+1}, p_{i+2}, \dots, p_{n-2}\}$.

end for

Соединяем две оставшиеся вершины в T .

Пример 2.27. С помощью алгоритма 2.12 построим дерево по коду Прюфера $P = \{3, 3, 4, 5, 4, 6\}$. Протокол работы запишем в таблице 2.17.

Таблица 2.17.

i	P	L	k	Ребро
1	$\{3, 3, 4, 5, 4, 6\}$	$\{1, 2, 3, 4, 5, 6, 7, 8\}$	1	(v_1, v_3)
2	$\{3, 4, 5, 4, 6\}$	$\{2, 3, 4, 5, 6, 7, 8\}$	2	(v_2, v_3)
3	$\{4, 5, 4, 6\}$	$\{3, 4, 5, 6, 7, 8\}$	3	(v_3, v_4)
4	$\{5, 4, 6\}$	$\{4, 5, 6, 7, 8\}$	7	(v_7, v_5)
5	$\{4, 6\}$	$\{4, 5, 6, 8\}$	5	(v_5, v_4)
6	$\{6\}$	$\{4, 6, 8\}$	4	(v_4, v_6)
	$\{\}$	$\{6, 8\}$		(v_6, v_8)

Получившееся дерево представлено на рисунке 2.33.

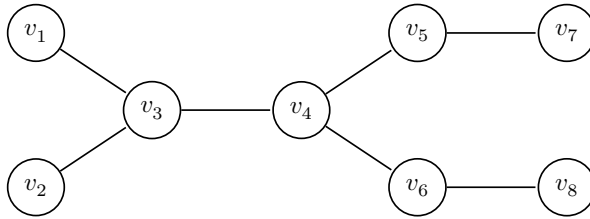


Рис. 2.33.

Пример 2.28. С помощью алгоритма 2.11 построим код Прюфера для графа на рисунке 2.34.

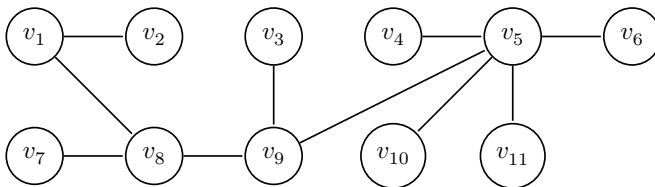


Рис. 2.34.

Протокол работы запишем в таблице 2.18.

Таблица 2.18.

i	Вершина	p_i	i	Вершина	p_i	i	Вершина	p_i
1	v_2	1	4	v_4	5	7	v_8	9
2	v_1	8	5	v_6	5	8	v_9	5
3	v_3	9	6	v_7	8	9	v_{10}	5

Получаем следующий код Прюфера:

$$P = \{1, 8, 9, 5, 5, 8, 9, 5, 5\}. \quad (2.7)$$

Пример 2.29. С помощью алгоритма 2.12 построим дерево по коду Прюфера (2.7). Протокол работы запишем в таблице 2.19.

Таблица 2.19.

i	P	L	k	Ребро
1	$\{1, 8, 9, 5, 5, 8, 9, 5, 5\}$	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$	2	(v_1, v_2)
2	$\{8, 9, 5, 5, 8, 9, 5, 5\}$	$\{1, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$	1	(v_1, v_8)
3	$\{9, 5, 5, 8, 9, 5, 5\}$	$\{3, 4, 5, 6, 7, 8, 9, 10, 11\}$	3	(v_3, v_9)
4	$\{5, 5, 8, 9, 5, 5\}$	$\{4, 5, 6, 7, 8, 9, 10, 11\}$	4	(v_4, v_5)
5	$\{5, 8, 9, 5, 5\}$	$\{5, 6, 7, 8, 9, 10, 11\}$	6	(v_5, v_6)
6	$\{8, 9, 5, 5\}$	$\{5, 7, 8, 9, 10, 11\}$	7	(v_7, v_8)
7	$\{9, 5, 5\}$	$\{5, 8, 9, 10, 11\}$	8	(v_9, v_8)
8	$\{5, 5\}$	$\{5, 9, 10, 11\}$	9	(v_5, v_9)
9	$\{5\}$	$\{5, 10, 11\}$	10	(v_5, v_{10})
	$\{\}$	$\{5, 11\}$		(v_5, v_{11})

Легко заметить, что построенное дерево совпадает с исходным графом (рисунок 2.34).

2.1.12. Главные циклы и коциклы

Определение 2.35. Пусть граф $G = (V, E)$ имеет p вершин и q ребер и число его компонент связности равно k . Тогда число $\varphi = p - k$ называют *корангом графа*, а $\mu = q - p + k$ — его *цикломатическим числом*.

Пусть $(V_1, E_1), \dots, (V_k, E_k)$ — компоненты связности графа, имеющие p_1, \dots, p_k вершин и q_1, q_2, \dots, q_k ребер. Таким образом, для каждой компоненты связности

$$\varphi_i = p_i - 1, \quad \mu_i = q_i - p_i + 1, \quad i = 1, \dots, k.$$

Очевидно, что $\varphi = \sum_{i=1}^k \varphi_i$, $\mu = \sum_{i=1}^k \mu_i$. Доказана следующая теорема.

Теорема 2.27.

- 1) $\mu \geq 0$ для любого графа G ;
- 2) связный граф G — дерево тогда и только тогда, когда $\mu = 0$.

Определение 2.36. Пусть $G = (V, E)$ — связный граф с p вершинами и q ребрами и $T = (V, E')$ — стягивающее дерево для G .

Ребра $e \in E'$ называют *ветвями*, ребра $y \in E \setminus E'$ — *хордами*.

Если к дереву T добавить хорду $y \in E \setminus E' : y = (v, u)$, то y и T образуют граф с циклом, который определяется единственным образом цепью, соединяющей вершины v, u . Этот цикл называют *главным циклом*, определяемым хордой y .

 **Замечание 2.30**

Ветвей в графе $p - 1$, хорд — $q - (p - 1) = q - p + 1 = \mu$, следовательно, всего существует μ главных циклов.

Определение 2.37. Введем операцию \oplus сложения по модулю 2 или симметрической разности над множествами M_1 и M_2 ребер графа $G = \{V, E\}$:

$$M_1 \oplus M_2 = \{e \in E \mid (e \in M_1 \text{ и } e \notin M_2) \text{ или } (e \notin M_1 \text{ и } e \in M_2)\}. \quad (2.8)$$

Определение 2.37 можно обобщить на произвольное число слагаемых M_1, \dots, M_k . В этом случае в 2.8 ребра, имеющие *парные* вхождения, будут удаляться (согласно свойству операции \oplus). То есть ребро $u \in M_1 \oplus \dots \oplus M_k$ тогда и только тогда, когда оно принадлежит нечетному количеству из M_1, \dots, M_k .

Теорема 2.28. Любой цикл Z графа G является суммой его некоторых главных циклов C_{i_1}, \dots, C_{i_k} :

$$Z = C_{i_1} \oplus \dots \oplus C_{i_k}. \quad (2.9)$$

Доказательство. Рассмотрим \mathbf{T} — стягивающее дерево для графа \mathbf{G} . Пусть C_1, \dots, C_n — главные циклы графа \mathbf{G} , определяемые стягивающим деревом \mathbf{T} .

Возьмем произвольный цикл Z . Пусть $S = \{e_1, \dots, e_k \mid e_i \in Z, e_i \notin \mathbf{T}\}$. Рассмотрим множество ребер $F = Z \oplus C_1 \oplus \dots \oplus C_n$.

Очевидно, что любое ребро $e_i \in S$ встречается в множестве F ровно в двух слагаемых: Z и C_j . По определению симметрической разности получаем, что множество F содержит только ребра из \mathbf{T} .

Заметим, что все вершины в $Z, C_1 \dots C_n$ четны (мы рассматриваем только простые циклы), следовательно, четны все вершины в F . Рассмотрим граф, образованный ребрами F . Если он не пуст, тогда в нем есть цикл, следовательно цикл есть и в \mathbf{T} . Получаем противоречие. Следовательно, множество F пусто, что доказывает равенство (2.9). ■

Пример 2.30. Рассмотрим граф на рисунке 2.35. Ребра, вошедшие в стягивающее дерево, имеют большую «толщину». Построим главные циклы графа \mathbf{G} для каждой хорды. Имеем:

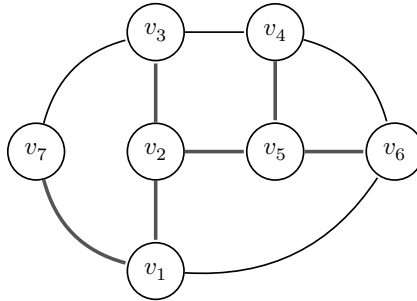


Рис. 2.35.

$$C_1 - (v_3, v_4) : v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_2 \rightarrow v_3,$$

$$C_2 - (v_4, v_6) : v_4 \rightarrow v_6 \rightarrow v_5 \rightarrow v_4,$$

$$C_3 - (v_1, v_6) : v_1 \rightarrow v_6 \rightarrow v_5 \rightarrow v_2 \rightarrow v_1,$$

$$C_4 - (v_3, v_7) : v_3 \rightarrow v_2 \rightarrow v_1 \rightarrow v_7 \rightarrow v_3.$$

Рассмотрим цикл $Z = \{v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_6 \rightarrow v_1\}$. Тогда

$$Z = C_1 \oplus C_2 \oplus C_3.$$

Для цикла $Z_1 = \{v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_3 \rightarrow v_7 \rightarrow v_1\}$ получим

$$Z_1 = C_1 \oplus C_4.$$

Пример 2.31. Рассмотрим граф на рисунке 2.36 и построим для него множество главных циклов.

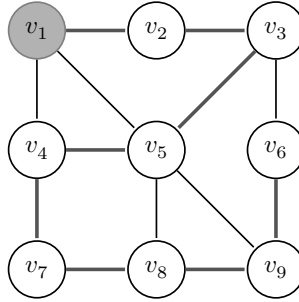


Рис. 2.36.

Построим стягивающее дерево поиском в глубину, начиная с вершины v_1 (выделена на рисунке) и выбирая из возможных вершину с меньшим номером.

Ребра полученного каркаса на рисунке 2.36 имеют большую «толщину». Построим для каждой хорды графа главные циклы. Имеем:

$$C_1 - (v_1, v_4) : v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_5 \rightarrow v_4 \rightarrow v_1,$$

$$C_2 - (v_1, v_5) : v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_5 \rightarrow v_1,$$

$$C_3 - (v_3, v_6) : v_3 \rightarrow v_6 \rightarrow v_9 \rightarrow v_8 \rightarrow v_7 \rightarrow v_4 \rightarrow v_5 \rightarrow v_3,$$

$$C_4 - (v_5, v_8) : v_5 \rightarrow v_8 \rightarrow v_7 \rightarrow v_4 \rightarrow v_5,$$

$$C_5 - (v_5, v_9) : v_5 \rightarrow v_9 \rightarrow v_8 \rightarrow v_7 \rightarrow v_4 \rightarrow v_5.$$

Рассмотрим цикл $Z = \{v_1 \rightarrow v_5 \rightarrow v_9 \rightarrow v_8 \rightarrow v_7 \rightarrow v_4 \rightarrow v_1\}$. Тогда

$$Z = C_1 \oplus C_2 \oplus C_5. \quad (2.10)$$

Замечание 2.31

Легко заметить, что для представления любого цикла в виде (2.9) нужно взять главные циклы тех хорд, которые входят в этот цикл. Например, для (2.10) в цикл Z входят три хорды:

$$(v_1, v_4) \in C_1, (v_1, v_5) \in C_2, (v_5, v_9) \in C_5.$$

Определение 2.38. Пусть x' — произвольная ветвь главного цикла, определяемого хордой x . Если к дереву \mathbf{T} добавить хорду x , то единственным простым циклом на полученном графе $(\mathbf{V}, \mathbf{E}' \cup \{x\})$ будет цикл, определяемый x .

Если из него исключить ветвь x' , то образуется новое дерево $(\mathbf{V}, \mathbf{E}' \cup \{x\} \setminus \{x'\})$, стягивающее граф \mathbf{G} . Такое преобразование называют элементарным преобразованием дерева.

Пример 2.32. На рисунке 2.37 показано элементарное преобразование дерева.

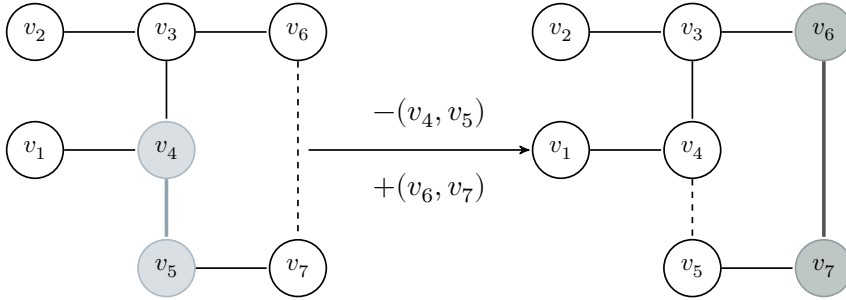


Рис. 2.37.

Теорема 2.29. Пусть $G = (V, E)$ — связный граф, $T = (V, E')$ — стягивающее его дерево. Пусть $x' \in E'$ — ветвь главного цикла, определяемого хордой $x \in E \setminus E'$. Тогда граф $(V, E' \cup \{x\} \setminus \{x'\})$ есть стягивающее дерево для G .

Определение 2.39. Разрезом связного графа $G = (V, E)$ называют множество ребер $C \subseteq E$, удаление которых делает граф несвязным. Простым, или минимальным, разрезом называют разрез, никакое собственное подмножество которого таким свойством не обладает.

Замечание 2.32

В этом разделе рассматриваются только простые разрезы, поэтому слово «простой» опускается. Определения цикла и разреза являются двойственными, поэтому разрез часто называют коциклом.

Пример 2.33. Рассмотрим примеры коциклов для графа на рисунке 2.38.

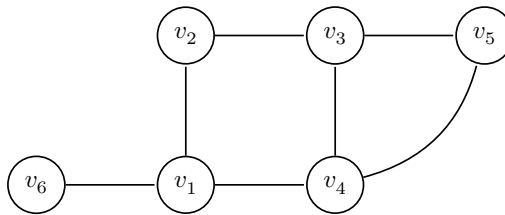


Рис. 2.38.

Коциклами для данного графа, например, являются одиночное ребро (v_1, v_6) , пары ребер $\{(v_1, v_2), (v_2, v_3)\}$ и $\{(v_5, v_3), (v_5, v_4)\}$. Заметим, что набор

$$\{(v_1, v_2), (v_1, v_6), (v_1, v_4)\}$$

коциклом не является, так как его собственное подмножество $\{(v_1, v_6)\}$ обладает свойством коцикла.

Пусть $G = (V, E)$ — связный граф, а C — его коцикл. Тогда $(V, E \setminus C)$ — несвязный граф.

Предположим, что его степень связности больше двух. Пусть

$$(V_1, E_1), (V_2, E_2), (V_3, E_3), \dots$$

— компоненты связности $(V, E \setminus C)$.

Так как граф G связан, существует $x \in C$, соединяющее вершины из V_i и V_j ($i \neq j$). Для простоты считаем, что $i = 1, j = 2$.

Тогда $C' = C \setminus \{x\}$ есть собственное подмножество C , но граф $(V, E \setminus C')$ несвязен. Действительно, вершины из V_1 и V_3 не связаны между собой. Противоречие с минимальностью множества C , следовательно, если C — коцикл, то степень связности графа $(V, E \setminus C)$ равна двум.

Пусть C — коцикл связного графа $G = (V, E)$, а $(V_1, E_1), (V_2, E_2)$ — две компоненты связности $(V, E \setminus C)$. Очевидно, что в C нет ребер, которые соединяли бы две вершины в V_1 или V_2 .

Введем обозначение:

$$E(V_1, V_2) = \{(v_i, v_j) \in E \mid v_i \in V_1, v_j \in V_2\}.$$

Тогда $C = E(V_1, V_2)$. Таким образом, доказана следующая теорема.

Теорема 2.30. Если C — коцикл связного графа $G = (V, E)$, то степень связности графа $(V, E \setminus C) = 2$. Пусть $(V_1, E_1), (V_2, E_2)$ — компоненты связности графа $(V, E \setminus C)$, тогда $C = E(V_1, V_2)$.

Следствие 2.10. Любой цикл и любой коцикл связного графа имеют четное число общих ребер.

Доказательство. Согласно теореме 2.30 произвольный коцикл можно представить в виде $C = E(V_1, V_2)$. Если некий цикл проходит только по вершинам V_1 или V_2 , то он не имеет общих ребер с коциклом C . Если же цикл проходит как по вершинам V_1 , так и по вершинам V_2 , то он имеет четное число ребер из C (выйдя из некоторой вершины цикл должен в нее вернуться). ■

Определение 2.40. Пусть $G = (V, E)$ — связный граф, $T = (V, E')$ — его стягивающее дерево.

Рассмотрим $x \in E'$ — ветвь этого дерева. Так как по теореме 2.13 x — мост, то граф $(V, E' \setminus \{x\})$ несвязен. Таким образом, множество $\{x\}$ — это коцикл дерева T .

Пусть $(V_1, E'_1), (V_2, E'_2)$ — компоненты связности $(V, E' \setminus \{x\})$. Обозначим через E'' ребра, которые являются хордами графа G и соединяют вершины из V_1 и V_2 , т. е. $E'' = (E \setminus E') \cap E(V_1, V_2)$. Очевидно, что $\{x\} \cup E''$ — коцикл графа G . Легко видеть, что E'' — это множество хорд, главные циклы которых включают ветвь x .

Этот коцикл $\{x\} \cup E''$ называют *главным коциклом*, определяемым ребром x . Так как число ветвей $\varphi = p - k$ (где p — множество вершин, а k — степень связности), существует φ главных коциклов.

Пример 2.34. Рассмотрим граф на рисунке 2.39. Ветви одного из возможных стягивающих деревьев графа отмечены большей «толщиной».

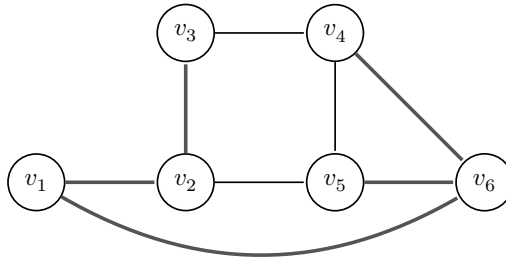


Рис. 2.39.

Рассмотрим построение главного коцикла для ветви (v_1, v_6) . Ее удаление разбивает дерево на две компоненты связности:

$$\{v_1, v_2, v_3\}, \{v_4, v_5, v_6\}.$$

На рисунке 2.40а они отмечены светло и темно-серыми цветами. Хорды графа показаны пунктиром.

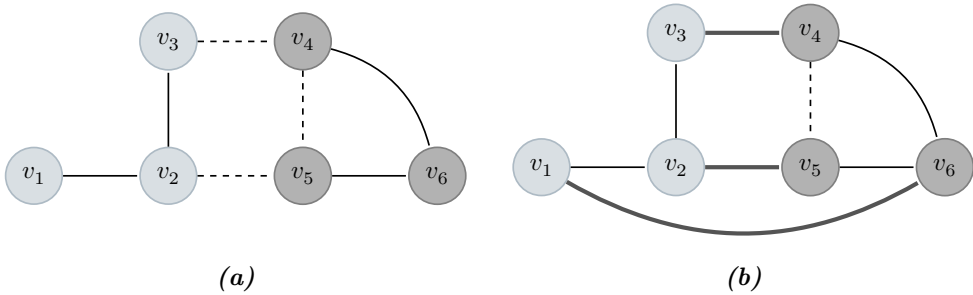


Рис. 2.40.

На рисунке 2.40b ребра главного коцикла отмечены большей «толщиной»: ветвь (v_1, v_6) и хорды $(v_2, v_5), (v_3, v_4)$.

Главные коциклы графа G для каждой ветви имеют вид:

$$\begin{aligned} (v_1, v_2) &: \{(v_3, v_4), (v_2, v_5), (v_1, v_2)\}, & (v_2, v_3) &: \{(v_3, v_4), (v_1, v_2), (v_2, v_3)\}, \\ (v_1, v_6) &: \{(v_3, v_4), (v_2, v_5), (v_1, v_6)\}, & (v_5, v_6) &: \{(v_4, v_5), (v_2, v_5), (v_5, v_6)\}, \\ (v_4, v_6) &: \{(v_3, v_4), (v_4, v_5), (v_4, v_6)\}. \end{aligned}$$

Пример 2.35. Построим множество главных коциклов для графа на рисунке 2.41.

Для построения его стягивающего дерева используем поиск в глубину, начиная с вершины v_1 (выделена на рисунке) и выбирая из возможных вершину с меньшим номером. Ребра стягивающего дерева графа отмечены большей «толщиной».

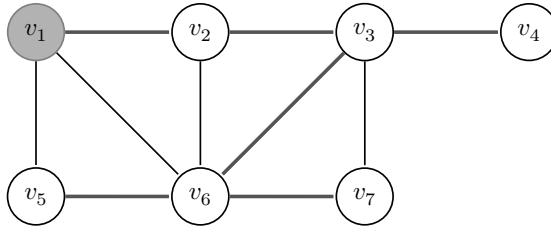


Рис. 2.41.

На рисунке 2.42 показано построение главного коцикла для ветви (v_2, v_3) . Удаление этой ветви разбивает дерево на две компоненты связности:

$$\{v_1, v_2\}, \{v_3, v_4, v_5, v_6, v_7\}.$$

На рисунке 2.42a они отмечены светло и темно-серыми цветами. Хорды графа показаны пунктиром.

На рисунке 2.42b ребра главного коцикла отмечены большей «толщиной»: ветвь (v_2, v_3) и хорды (v_1, v_5) , (v_1, v_6) , (v_2, v_6) .

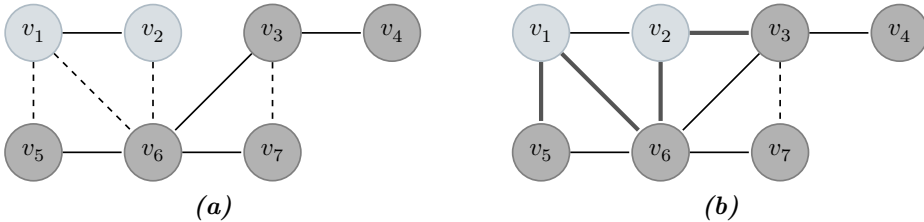


Рис. 2.42.

Остальные коциклы графа имеют вид:

$$(v_1, v_2) : \{(v_1, v_5), (v_1, v_6), (v_1, v_2)\},$$

$$\begin{aligned} (v_3, v_4) &: \{(v_3, v_4)\}, \\ (v_3, v_6) &: \{(v_1, v_5), (v_1, v_6), (v_2, v_6), (v_3, v_7), (v_3, v_6)\}, \\ (v_5, v_6) &: \{(v_1, v_5), (v_5, v_6)\}, \\ (v_6, v_7) &: \{(v_3, v_7), (v_6, v_7)\}. \end{aligned}$$

Определение 2.41. В теореме 2.5 было введено понятие объединения совокупности ребер циклов, никакие два из которых не имеют общих ребер.

Непустое объединение множеств ребер некоторого числа циклов, никакие два из которых не имеют общих ребер, называют *границей*.

Непустое объединение множеств ребер некоторого числа коциклов, никакие два из которых не имеют общих ребер, называют *кограницей*.

Лемма 2.5. Пусть $G = (V, E)$ — связный граф и множество ребер $B \subseteq E$ не содержит целиком ни одного коцикла. Тогда существует дерево, стягивающее G , которое содержит только ребра из $E \setminus B$.

Доказательство. Рассмотрим $T = (V, E')$ — стягивающее дерево графа G , содержащее наибольшее число ребер из $E \setminus B$. Покажем, что $E' \subset E \setminus B$. Пусть существует ребро $x \notin E' \setminus B$. Рассмотрим главный коцикл, определенный x (x является ветвью T).

Так как нет коциклов, полностью лежащих в B (по предположению), существует хорда $x' \in E \setminus B$, которая входит в этот коцикл. Другими словами, главный цикл, определяемый хордой x' , содержит ветвь x .

Сделаем элементарное преобразование дерева T (см. определение 2.38): включим хорду x' и выключим ветвь x . Тогда у нового дерева число ребер из $E \setminus B$ на одно больше, чем у T . Противоречие, следовательно, $E' \subset E \setminus B$. ■

Теорема 2.31. Пусть B — непустое подмножество ребер графа G , имеющее четное число общих ребер с любым его циклом. Тогда B — объединение множеств некоторых коциклов, никакие два из которых не имеют общих ребер (т. е. B — кограница).

Доказательство. Пусть множество B не содержит целиком ни одного коцикла, тогда по лемме 2.5 существует дерево $T = (V, E')$, стягивающее G , такое, что $E' \cap B = \emptyset$. Возьмем ребро $x \in B$ ($B \neq \emptyset$). Так как $x \notin E'$, то x — хорда.

Рассмотрим главный цикл, определенный x , тогда все остальные ребра цикла есть ветви T (по определению главного цикла). Таким образом, этот главный цикл не имеет с B других ребер, кроме x (так как $E' \cap B = \emptyset$). Противоречие.

Следовательно, существует коцикл C_1 , целиком лежащий в B . Рассмотрим множество $B_1 = B \setminus C_1$. Если множество $B_1 \neq \emptyset$, то по следствию 2.10

к теореме 2.30 множество B_1 обладает тем же свойством, что и B . Продолжая этот процесс, пока очередное множество $B_k \neq \emptyset$, получаем требуемое представление для B . ■

Следствие 2.11. Звезда $\delta(a)$ произвольной вершины является когранницей.

Доказательство. Звезда $\delta(a)$ и любой цикл графа либо не пересекаются, либо имеют два общих ребра. ■

Теорема 2.32. Пусть B — непустое подмножество ребер связного графа G , имеющее четное число общих ребер с любым его коциклом. Тогда B есть объединение множеств некоторых циклов, никакие два из которых не имеют общих ребер (т. е. B — граница).

Упражнение 2.7. Докажите теорему 2.32.

2.1.13. Двудольные графы

Определение 2.42. *Двудольным графом (биграфом, четным графом)* называют граф $G(V, E)$ такой что множество его вершин V есть объединение двух непересекающихся, непустых множеств V_1 и V_2 . При этом каждое ребро графа соединяет вершину из множества V_1 с вершиной из множества V_2 .

Множества V_1 и V_2 называют долями двудольного графа. Если двудольный граф содержит все ребра, соединяющие множества V_1 и V_2 , то это полный двудольный граф. При этом если $|V_1| = n$, а $|V_2| = m$, то такой граф обозначают как $K_{n,m}$.

Граф $K_{1,m}$ называется *графом-звездой* и обозначается S_{m-1} , при этом число $m - 1$ (число ребер) называют *порядком звезды*.

Замечание 2.33

Пусть $G(V, E)$ — двудольный граф. Его всегда можно изобразить так, чтобы вершины доли V_1 лежали слева, а доли V_2 — справа. Иногда вершины доли V_1 будем обозначать литерой «L», а доли V_2 — литерой «R».

Пример 2.36. Любое дерево — двудольный граф. Для доказательства достаточно расположить вершины по уровням. Нечетные уровни образуют множество $V_1(L)$, а четные — множество $V_2(R)$ (или наоборот) (рисунок 2.43a).

Любая прямоугольная «решетка» размером $n \times m$ является двудольным графом. Для доказательства посмотрите разметку вершин на рисунке 2.43b.

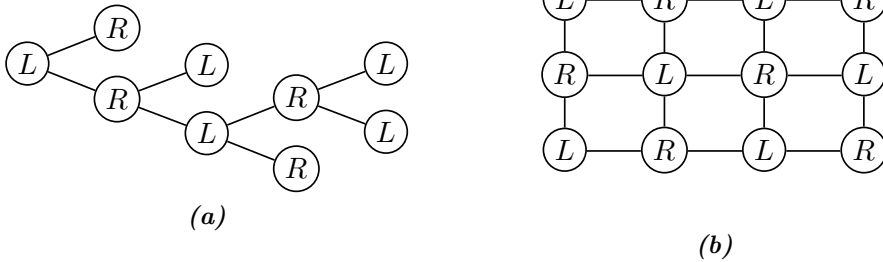


Рис. 2.43.

Теорема 2.33 (Кениг). Для двудольности графа необходимо и достаточно, чтобы он не содержал циклов нечетной длины.

Доказательство. Необходимость очевидна. Докажем достаточность.

Возьмем произвольную вершину $v \in V$. Разобьем множество вершин графа на два непересекающихся множества:

$$V_1 = \{u \in V \mid d(v, u) - \text{четно}\}, V_2 = \{u \in V \mid d(v, u) - \text{нечетно}\}.$$

Покажем, что в графе нет ребер с вершинами в V_1 или V_2 . Предположим, что существует ребро $(x, y) \in E$, $x, y \in V_1$ или $x, y \in V_2$.

Рассмотрим кратчайшие пути $v \rightarrow x$ и $v \rightarrow y$:

$$v \rightarrow v_1 \rightarrow \dots \rightarrow v_{i-1} \rightarrow x, \quad v \rightarrow u_1 \rightarrow \dots \rightarrow u_{j-1} \rightarrow y.$$

Поскольку i и j одинаковой четности, $i + j$ — четное число. Рассмотрим цикл:

$$v \rightarrow v_1 \rightarrow \dots \rightarrow v_{i-1} \rightarrow x \rightarrow y \rightarrow u_{j-1} \rightarrow \dots \rightarrow u_1 \rightarrow v.$$

Его длина $i + j + 1$ — нечетное число. Полученное противоречие устанавливает достаточность. ■

Рассмотрим алгоритм распознавания двудольности графа.

Введем следующие обозначения. Пусть $G'(V', E')$ — подграф графа $G(V, E)$. Если множество ребер E' совпадает с множеством всех ребер графа G , оба конца которых принадлежат V' , то G' называют *подграфом, порожденным множеством вершин V'* , и обозначают $G(V')$.

Для просмотра вершин графа будем использовать алгоритм поиска в ширину.

Как и ранее, обозначим очередь — Q , начало и конец очереди — $\text{start}(Q)$ и $\text{end}(Q)$, список просмотренных вершин — L , номера вершин — N .

Алгоритм 2.13. Распознавание двудольности графа

$\text{end}(Q) \leftarrow v_0$

// записываем в конец очереди произвольную вершину v_0

```

 $L \leftarrow v_0, N(v_0) \leftarrow 0$  // помечаем ее как просмотренную и присваиваем ей номер 0
while  $Q \neq \{\emptyset\}$  do
     $v \leftarrow \text{start}(Q)$  //  $v$  — вершина, находящаяся в начале очереди
    if существует смежная с  $v$  непросмотренная вершина  $u \notin L$  then
         $\text{end}(Q) \leftarrow u$  // записываем в конец очереди вершину  $u$ 
         $L \leftarrow u$  // помечаем вершину  $u$  как просмотренную
         $N(u) \leftarrow N(v) + 1$  // присваиваем ей номер  $N(v) + 1$ 
    else
        Удаляем вершину  $v$  из начала очереди
    end if
end while
Разбиваем множество вершин графа  $G(V, E)$  на две части:  $V_{\text{even}}$  и  $V_{\text{odd}}$ 
 $V_{\text{even}} \leftarrow$  вершины с четными номерами
 $V_{\text{odd}} \leftarrow$  вершины с нечетными номерами
if  $(G(V_{\text{even}}) = \{\emptyset\} \wedge G(V_{\text{odd}}) = \{\emptyset\})$  then
     $G(V, E)$  — двудольный граф
else
    граф  $G$  не является двудольным
end if

```

Пример 2.37. Применим алгоритм 2.13 для определения двудольности графа на рисунке 2.44а.

Нумерация вершин показана на рисунке 2.44b. Тогда множества вершин:

$$V_{\text{even}} = \{v_1, v_2, v_3\}, \quad V_{\text{odd}} = \{v_4, v_5, v_6\}.$$

Очевидно, что множества $G(V_{\text{even}})$ и $G(V_{\text{odd}})$ пусты. Следовательно, граф на рисунке 2.44а — двудольный.

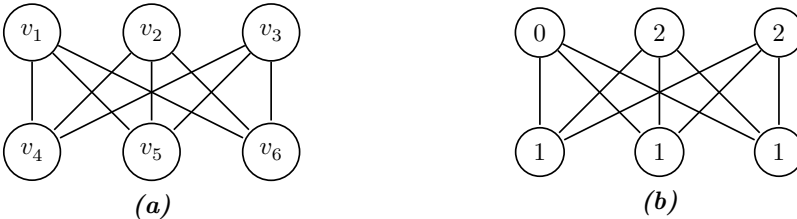


Рис. 2.44.

Замечание 2.34

Граф на рисунке 2.44а, двудольность которого была определена выше — это граф $K_{3,3}$ согласно определению 2.42.

Пример 2.38. С помощью алгоритма 2.13 решим вопрос о двудольности графа на рисунке 2.45а.

Поиск в ширину начнем с вершины v_0 . При вариативности выбора будем брать вершину с минимальным номером из всех возможных.

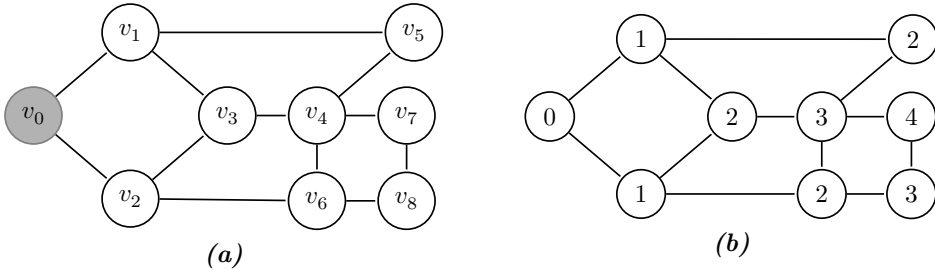


Рис. 2.45.

Последовательность просмотра вершин при поиске в ширину следующая (в скобках — цвет вершины):

$$v_0(0), v_1(1), v_2(1), v_3(2), v_5(2), v_6(2), v_4(3), v_8(3), v_7(4).$$

Нумерация вершин показана на рисунке 2.45b. Получаем двудольный граф (рисунок 2.46).

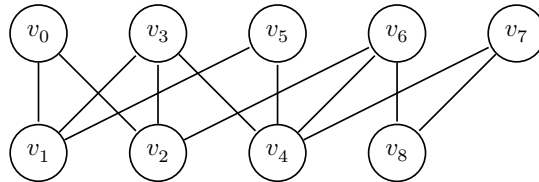


Рис. 2.46.

Теорема 2.34 (О сумме степеней двудольного графа). Суммы степеней вершин долей двудольного графа равны.

Доказательство. Пусть v_1, \dots, v_k — вершины одной доли, а u_1, \dots, u_p — вершины второй доли. Пусть $d(v)$ — валентность вершины v . Тогда из первой доли выходит $d(v_1) + \dots + d(v_k)$ ребер, а из второй — $d(u_1) + \dots + d(u_p)$ ребер. Так как это одни и те же ребра, получаем

$$\sum_{i=1}^k d(v_i) = \sum_{j=1}^p d(u_j).$$



2.1.14. Клики и независимые множества

Определение 2.43. *Кликой* в неориентированном графе $G = (V, E)$ называется подмножество вершин $C \subseteq V$ такое, что любые две вершины из C являются смежными.

Клика, которая не содержится в клике большего размера, называется *максимальной*.

Наибольшая клика — это клика максимального (по числу вершин) размера для данного графа. Число вершин в наибольшей клике графа G называется *кликовым числом* и обозначается $\omega(G)$.

Замечание 2.35

| Можно дать равносильное определение клики как множества вершин, порождающего полный подграф в графе.

В настоящее время понятие клики используется в социологии, биоинформатике, анализе химических соединений, коммуникационных сетей и др.

Рассмотрим двойственное понятие для клики.

Определение 2.44. Множество вершин графа называется *независимым*, если никакие две вершины этого множества не смежны.

Если при добавлении к независимому множеству любой вершины графа оно перестает быть независимым, такое множество называется *максимальным независимым множеством*.

Максимальное (по числу вершин) независимое множество называется *наибольшим независимым множеством*, а его размер называется *числом независимости* и обозначается $\alpha(G)$.

Для определения двойственности введенных определений нам потребуется понятие дополнения графа.

Определение 2.45. *Дополнением* графа, или *обратным* графом, для графа G называется граф, имеющий то же множество вершин, в котором две несовпадающие вершины смежны тогда и только тогда, когда они не смежны в исходном графе.

Замечание 2.36

| Очевидно, что граф и его дополнение образуют полный граф.

Пример 2.39. Примеры графа и его дополнения представлены соответственно на рисунке 2.47а и б.

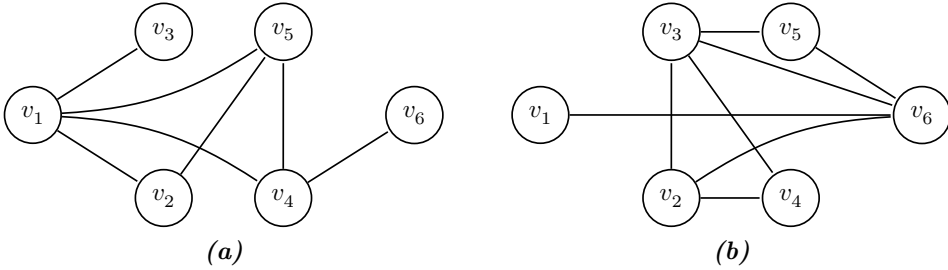


Рис. 2.47.

Очевидно следующее утверждение.

Теорема 2.35. Множество вершин графа независимо тогда и только тогда, когда оно является кликой в дополнении графа.

Замечание 2.37

К отысканию наибольшего независимого множества вершин в графе сводится, например, известная задача о восьми ферзях. Требуется так расставить на шахматной доске наибольшее число ферзей, чтобы они не атаковали друг друга.

Задачу придумал в 1848 г. шахматный композитор Макс Беззель, хотя в литературе ее часто приписывают Гауссу, который работал над ее решением.

Таких ферзей, очевидно, может быть не более восьми, так как никакие два из них не должны находиться на одной вертикали или горизонтали. Рассмотрим граф, вершины которого соответствуют клеткам доски, а ребра — парам клеток, лежащих на одной вертикали, горизонтали или диагонали. Ясно, что требуемой в задаче расстановке ферзей соответствует наибольшее независимое множество в этом графе.

Эта проблема будет подробно рассмотрена далее, в задаче 2.6.

Рассмотрим алгоритм для поиска всех клик неориентированного графа. Алгоритм был разработан голландскими математиками Броном и Кербощем в 1973 г. [71] и до сих пор является одним из самых эффективных алгоритмов поиска клик.

Описываемый подход основан на *методе ветвей и границ*, который подробно будет рассмотрен в разделе 2.2.12.

Идея алгоритма очень проста. Заметим, что любая клика в графе является его максимальным по включению полным подграфом. Начиная с одиночной вершины (образующей полный подграф), алгоритм на каждом шаге пытается увеличить уже построенный полный подграф, добавляя в

него вершины из множества кандидатов. При этом отсекаются вершины, которые ранее уже были использованы для увеличения полного подграфа.

Введем следующие обозначения.

1. Множество `cand` — вершины кандидаты для увеличения клики.
2. Множество `verif` — уже использованные вершины графа.
3. Множество `compsub` — текущий полный подграф.

Алгоритм является рекурсивной процедурой, применяемой к этим трем множествам.

Алгоритм 2.14. Брона — Кербоша

Инициализация:

`cand = V`

`verif = ∅`

`compsub = ∅`

procedure BRONKERBOSCH(`cand`, `verif`)

while (`cand` ≠ ∅) и (в `verif` нет вершины, смежной со всеми вершинами `cand`) **do**

// Добавляем в `compsub` вершину $v_i \in \text{cand}$

`compsub` ← $v_i \mid v_i \in \text{cand}$

// Формируем множества cand_1 и verif_1 из вершин из `cand` и `verif`, смежных с v_i

`cand`₁ = { $v_j \mid v_j \in \text{cand}, (v_j, v_i) \in E$ }

`verif`₁ = { $v_k \mid v_k \in \text{verif}, (v_k, v_i) \in E$ }

if (`cand`₁ = ∅) и (`verif`₁ = ∅) **then**

`compsub` — текущая клика

else

BRONKERBOSCH(`cand`₁, `verif`₁)

end if

// Переносим вершину v_i из множеств `compsub` и `cand` в множество `verif`

`compsub` → v_i , `cand` → v_i , `verif` ← v_i

end while

end procedure

Пример 2.40. Рассмотрим работу алгоритма 2.14 на примере графа на рисунке 2.48а. На рисунке 2.48б изображено дополнение к исходному графу.

Проанализируем работу алгоритма 2.14 по шагам.

1: **Инициализация:**

`cand` = { $v_1, v_2, v_3, v_4, v_5, v_6$ }, `verif` = ∅, `compsub` = ∅

2: BRONKERBOSCH(`cand`, `verif`)

```

3:   compsub = {v1}, cand1 = {v2}, verif1 = ∅
4:   BRONKERBOSCH(cand1, verif1)
5:       compsub = {v1, v2}, cand2 = ∅, verif2 = ∅
6:       return {v1, v2} — клика
7:       compsub = {v1}, cand1 = ∅, verif1 = {v2}
8:   end procedure
9:   compsub = ∅, cand = {v2, v3, v4, v5, v6}, verif = {v1}
10:  compsub = {v2}, cand1 = {v3, v4}, verif1 = {v1}
11:  BRONKERBOSCH(cand1, verif1)
12:      compsub = {v2, v3}, cand2 = {v4}, verif2 = ∅
13:      BRONKERBOSCH(cand2, verif2)
14:          compsub = {v2, v3, v4}, cand3 = ∅, verif3 = ∅
15:          return {v2, v3, v4} — клика
16:          compsub = {v2, v3}, cand2 = ∅, verif2 = {v4}
17:      end procedure
18:      compsub = {v2}, cand1 = {v4}, verif1 = {v2, v3}
19:      compsub = {v2, v4}, cand2 = ∅, verif2 = {v2, v3}
20:      BRONKERBOSCH(cand2, verif2)          // «холостой» вызов: cand2 = ∅
21:      end procedure
22:      compsub = {v2}, cand1 = ∅, verif1 = {v2, v3, v4}
23:  end procedure
24:  compsub = ∅, cand = {v3, v4, v5, v6}, verif = {v1, v2}
25:  compsub = {v3}, cand1 = {v4, v5, v6}, verif1 = {v2}
26:  BRONKERBOSCH(cand1, verif1)
27:      compsub = {v3, v4}, cand2 = {v5, v6}, verif2 = {v2}
28:      BRONKERBOSCH(cand2, verif2)
29:          compsub = {v3, v4, v5}, cand3 = {v6}, verif3 = ∅
30:          BRONKERBOSCH(cand3, verif3)
31:              compsub = {v3, v4, v5, v6}, cand4 = ∅, verif4 = ∅
32:              return {v3, v4, v5, v6} — клика
33:              compsub = {v3, v4, v5}, cand3 = ∅, verif3 = ∅
34:          end procedure
35:          compsub = {v3, v4}, cand2 = {v6}, verif2 = {v2, v5}
36:      end procedure          // v5 ∈ verif2 смежна со всеми вершинами cand2
37:      compsub = {v3}, cand1 = {v5, v6}, verif1 = {v2, v4}
38:  end procedure          // v4 ∈ verif1 смежна со всеми вершинами cand1
39:  compsub = ∅, cand = {v4, v5, v6}, verif = {v1, v2, v3}
40: end procedure          // v3 ∈ verif смежна со всеми вершинами cand

```

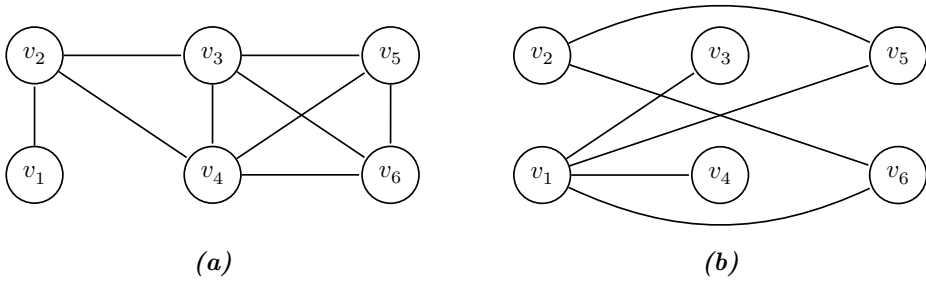



Рис. 2.48.

В результате работы алгоритма 2.14 найдены три клики графа на рисунке 2.48а.

$$\{v_1, v_2\}, \{v_2, v_3, v_4\}, \{v_3, v_4, v_5, v_6\}. \quad (2.11)$$

Последняя клика в (2.11) — наибольшая в графе. Заметим, что эти же наборы вершин являются независимыми множествами в графе-дополнении (рисунок 2.48b), а набор $\{v_3, v_4, v_5, v_6\}$ образует его наибольшее независимое множество.

2.1.15. Планарность

Часто встречаются ситуации, когда важно выяснить, возможно ли нарисовать граф на плоскости так, чтобы его ребра не пересекались. Например, в радиоэлектронике при изготовлении микросхем печатным способом электрические цепи наносят на плоскую поверхность изоляционного материала. А так как проводники не изолированы, то они не должны пересекаться. Аналогичная задача возникает при проектировании железнодорожных и других путей, где нежелательны пересечения. Таким образом возникает понятие плоского графа.

В дальнейшем нам понадобится одно из важнейших понятий в теории графов — изоморфизм графов.

Определение 2.46. Графы $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ называют *изоморфными* (обозначение: $G_1 \sim G_2$), если между ними существует взаимно однозначное отображение $\varphi: G_1 \rightarrow G_2 (V_1 \rightarrow V_2, E_1 \rightarrow E_2)$, которое сохраняет соответствие между ребрами (дугами) графов, т. е. для любого ребра (дуги) $e = (v, u)$ верно:

$$e' = \varphi(e) = (\varphi(v), \varphi(u)), \quad (e \in E_1, e' \in E_2).$$

Отображение φ есть изоморфное отображение.

Характеристики графов, инвариантные относительно изоморфизмов графов (т. е. принимающие одинаковые значения на изоморфных графах), называют *инвариантами* графов.

Таким образом, изоморфные графы различаются только обозначением вершин. К сожалению, установить изоморфизм двух графов визуально достаточно трудно, что показывает следующий пример.

Пример 2.41. На рисунке 2.49 *a* и *b* изображены два изоморфных графа. Приведем одно из возможных изоморфных отображений:

$$v_0 \rightarrow u_0, v_1 \rightarrow u_3, v_2 \rightarrow u_5, v_3 \rightarrow u_6, v_4 \rightarrow u_7, v_5 \rightarrow u_2, v_6 \rightarrow u_1, v_7 \rightarrow u_4, \\ v_8 \rightarrow u_9, v_9 \rightarrow u_8, v_{10} \rightarrow u_{10}.$$

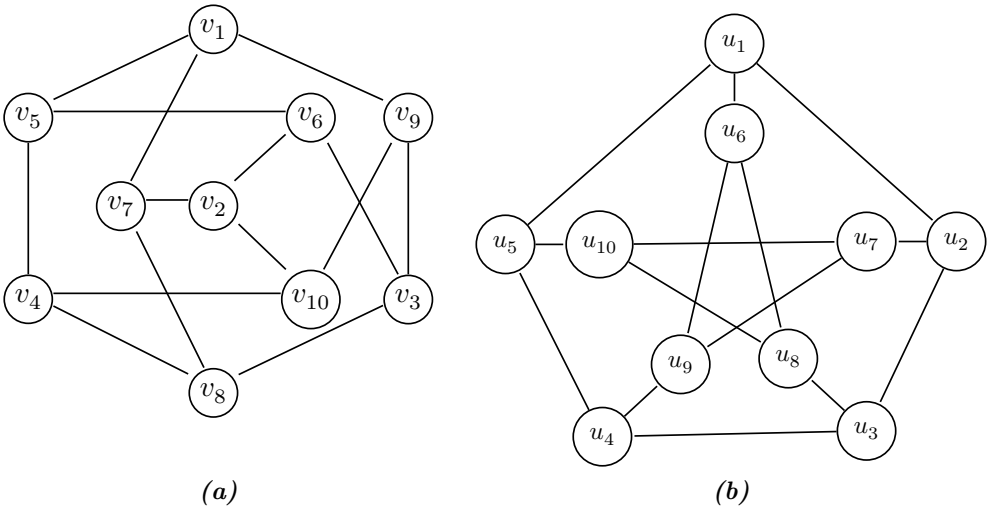


Рис. 2.49.

На рисунке 2.49*b* изображен *граф Петерсена*, он часто используется для примеров и контрпримеров в различных задачах теории графов. Граф Петерсена еще встретится далее как пример для теоремы 2.39.

Историческая справка

Граф Петерсена назван в честь Юлиуса Петерсена, датского математика, внесшего фундаментальный вклад в современную теорию графов. Опубликован им в 1898 г. в качестве контрпримера к гипотезе Тейта, связанной с задачей четырех красок.

К инвариантам графа можно отнести число ребер, число вершин, набор локальных степеней вершин, число компонент связности и т. д. Однако совпадение инвариантов является необходимым, но недостаточным условием наличия изоморфизма графов, что показывает следующий пример.

Пример 2.42. У графов на рисунке 2.50а и б одинаковое количество вершин и ребер и одинаковые наборы локальных степеней вершин, но эти графы неизоморфны.

В графе на рисунке 2.50а есть циклы длиной 3, а в графе на рисунке 2.50б таких циклов нет.

В изоморфных графах каждый цикл определенной длины в одном графе должен переходить в цикл соответствующей длины в другом графе.

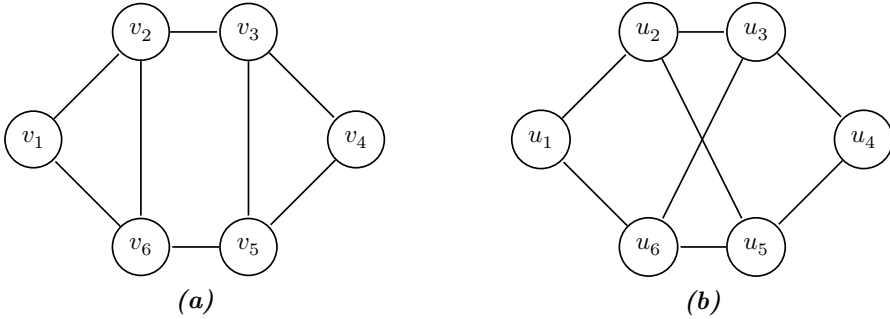


Рис. 2.50.

Казалось бы, можно установить изоморфизм графов G и G_1 , если путем перестановки строк и столбцов матрицы смежности графа G получить матрицу смежности графа G_1 . Однако перебор всех возможных перестановок имеет факториальную вычислительную сложность (раздел 3.5.8).

Это еще и оценка, не учитывающая зависимость времени сравнения матриц смежности от их размерности.

Задача 2.2. Определите, изоморфны ли графы на рисунке 2.51а и б.

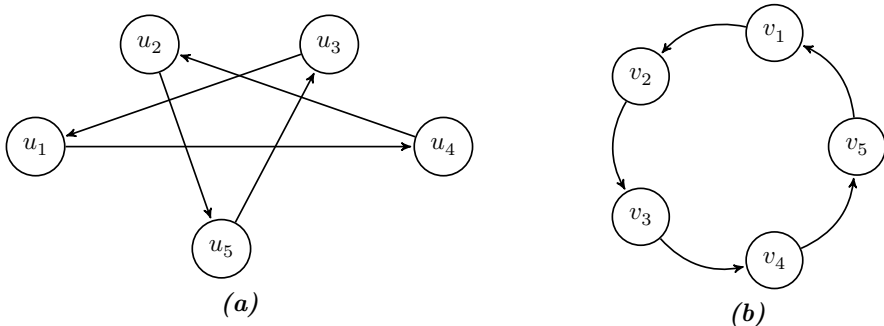


Рис. 2.51.

Определение 2.47. *Плоским* графом называют граф, вершины которого являются точками плоскости, а ребра — непрерывными плоскими линиями без самопересечений, соединяющими соответствующие вершины

так, что никакие два ребра не имеют общих точек, кроме инцидентной им обоим вершины.

Любой граф, изоморфный плоскому графу, называют *планарным*. О планарных графах говорят, что они укладываются на плоскости (имеют плоскую укладку).

Пример 2.43. Граф на рисунке 2.52а не является плоским, однако он планарный, поскольку изоморфен плоскому полному графу K_4 (рисунок 2.52б).

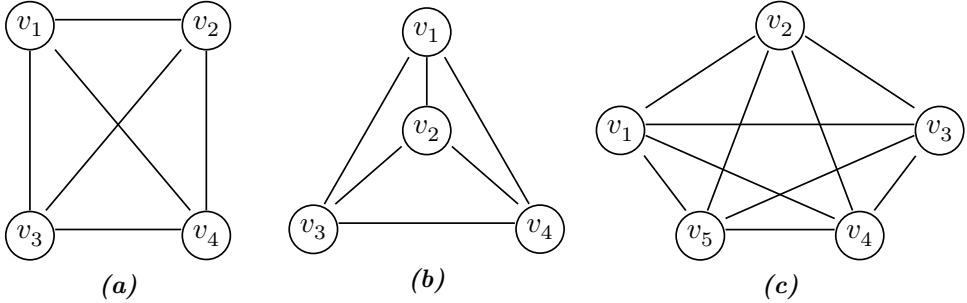


Рис. 2.52.

Заметим, что полный граф K_5 (рисунок 2.52с) не является планарным.

Если у графа K_5 удалить 2 ребра (v_1, v_5) и (v_3, v_4) (на рисунке 2.53а они изображены пунктиром), граф станет планарным, изоморфный ему граф показан на рисунке 2.53б.

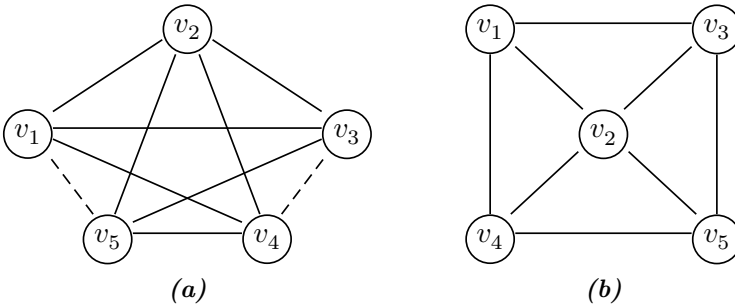


Рис. 2.53.

Далее будет строго доказано, что граф K_5 не планарен. Легко установить справедливость следующей теоремы.

Теорема 2.36. Каждый граф можно уложить в трехмерном евклидовом пространстве \mathbf{R}^3 .

Доказательство. Рассмотрим произвольный граф $G = (V, E)$. Все его вершины разместим в различных точках координатной оси x . Рассмотрим

пучок плоскостей, проходящих через x , и зафиксируем $|E|$ различных таких плоскостей. Теперь каждое ребро $(u, v) \in E$ изобразим полуокружностью, проходящей в соответствующей плоскости через вершины u и v . Очевидно, что различные ребра не будут пересекаться, кроме как в общих вершинах. ■

Очевидны следующие свойства планарных графов:

- 1) всякий подграф планарного графа планарен;
- 2) граф планарен тогда и только тогда, когда каждая его связная компонента — планарный граф.

Введем следующие определения.

Определение 2.48. Рассмотрим плоский граф. Каждый цикл такого графа ограничивает две части плоскости: одна — внутренняя (ограниченная), другая — внешняя (неограниченная). Если по части плоскости, ограниченной циклом плоского графа, не проходит ни одна цепь этого графа, начало и конец которой принадлежат обсуждаемому циклу, то эту часть плоскости называют *гранью* плоского графа.

Если грань неограниченная, то ее называют внешней; ограниченную грань плоского графа называют внутренней.

Границей грани будем считать множество вершин и ребер, принадлежащих данной грани. Отметим, что всякий плоский граф имеет одну, и притом единственную, неограниченную грань. Такую грань называют *внешней*, а все остальные — *внутренними*.

Замечание 2.38

Будем считать, что если в графе нет циклов, то внешняя грань ограничена всеми ребрами графа.

Пример 2.44. На рисунке 2.54 у графа три внутренние грани, образованные циклами:

$$(v_4, v_6, v_5, v_4), (v_2, v_3, v_8, v_7, v_2), (v_1, v_2, v_7, v_1),$$

и одна внешняя грань.

Пример 2.45. Граф на рисунке 2.55 имеет четыре внутренние грани, ограниченные циклами:

$$(v_1, v_2, v_3, v_1), (v_1, v_3, v_6, v_7, v_1), (v_1, v_7, v_6, v_1), (v_3, v_4, v_5, v_6),$$

и одну внешнюю грань, ограниченную циклом $(v_1, v_2, v_3, v_4, v_5, v_6, v_1)$.

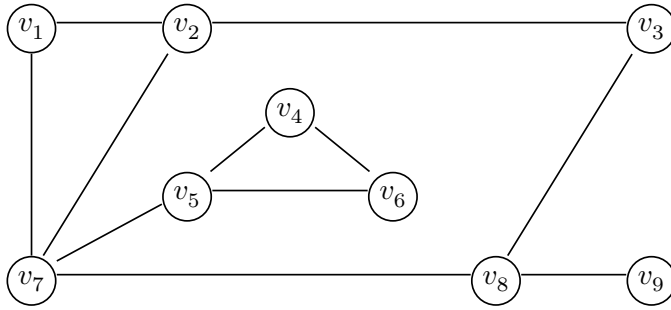


Рис. 2.54.

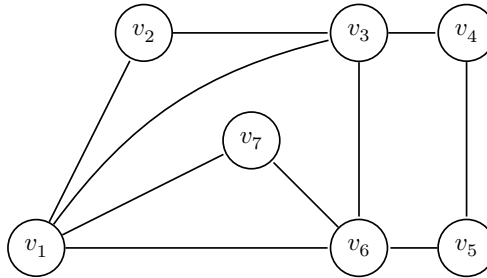


Рис. 2.55.

Оказывается, что число граней плоского графа связано с числом вершин и ребер постоянным соотношением, установленным Эйлером.

Теорема 2.37 (Эйлер). Пусть $G = (V, E)$ — плоский связный граф, имеющий p вершин, q ребер и f граней. Тогда $p - q + f = 2$.

Доказательство. Зафиксируем число вершин графа и проведем доказательство индукцией по числу ребер. Поскольку граф G связан, то $q \geq p - 1$.

Пусть $q = p - 1$, тогда в силу связности граф G является деревом, в нем нет циклов и, следовательно, $f = 1$ (существует одна внешняя грань). База индукции установлена.

Пусть теорема справедлива для всех q таких, что $p - 1 \leq q < q_1$. Докажем ее для q_1 .

Поскольку $q_1 > p - 1$, то граф G содержит цикл. Пусть $e \in E$ — некоторое ребро этого цикла. Тогда ребро e принадлежит разным граням. Удалим ребро e из графа G . Тогда в новом графе G_1 эти две грани сольются в одну, но при этом граф G_1 останется связным, так как удаленное ребро принадлежало циклу. Граф G_1 имеет p вершин, $q_1 - 1$ ребро и $f - 1$ грань. По индукционному предположению справедливо соотношение

$$p - (q_1 - 1) + (f - 1) = 2.$$

Отсюда следует, что $p - q_1 + f = 2$. Это доказывает теорему. ■

Следствие 2.12. В условиях теоремы 2.37 при $p > 3$ справедливо равенство

$$3p - q \geq 6. \quad (2.12)$$

Доказательство. Обозначим через φ_k число граней, ограниченных k ребрами. Так как в графе G нет петель и параллельных ребер, то в силу замечания 2.38:

$$\varphi_0 = \varphi_1 = \varphi_2 = 0.$$

Запишем очевидные соотношения:

$$f = \varphi_3 + \varphi_4 + \dots, \quad 2q = 3\varphi_3 + 4\varphi_4 + \dots. \quad (2.13)$$

В первом соотношении просуммированы все грани, во втором — ребра, ограничивающие каждую грань, при этом каждое ребро учитывается дважды.

Из равенств (2.13) легко получить, что $2q \geq 3f$. По теореме Эйлера $p - q + f = 2$. Отсюда получаем требуемое неравенство. ■

Используя теорему Эйлера, можно установить непланарность некоторых конкретных графов.

Рассмотрим полный граф K_5 из примера 2.43. Для него $p = 5, q = 10$. Если бы граф K_5 был плоским, то согласно (2.12) $15 - 10 \geq 6$. Полученное противоречие и доказывает непланарность графа K_5 .

Легко доказывается непланарность полного двудольного графа $K_{3,3}$, представленного на рисунке 2.44а. Действительно, для этого графа $p = 6, q = 9$. Если граф $K_{3,3}$ плоский, то число граней $f = 9 - 6 + 2 = 5$. Для двудольного графа $K_{3,3}$ нет циклов длиной 3, поэтому $\varphi_3 = 0$, следовательно, должны выполняться:

$$5 = \varphi_4 + \varphi_5 + \dots, \quad 18 = 4\varphi_4 + 5\varphi_5 + \dots$$

Отсюда легко получить противоречие.

Имеется несколько критериев непланарности графа. Приведем без доказательства *критерий Понтрягина — Куратовского*. Для этого введем понятие *гомеоморфизма* графа.

Определение 2.49. Операцией *разбиения ребра* $e = (v, u)$ называют операцию замены ребра e двумя ребрами $e_1 = (v, w)$ и $e_2 = (w, u)$, где w — новая вершина. Два графа называют гомеоморфными, если они могут быть получены из одного графа с помощью операций разбиения ребер.

Пример 2.46. Из графа на рисунке 2.56а можно получить два гомеоморфных графа (рисунок 2.56b и c):

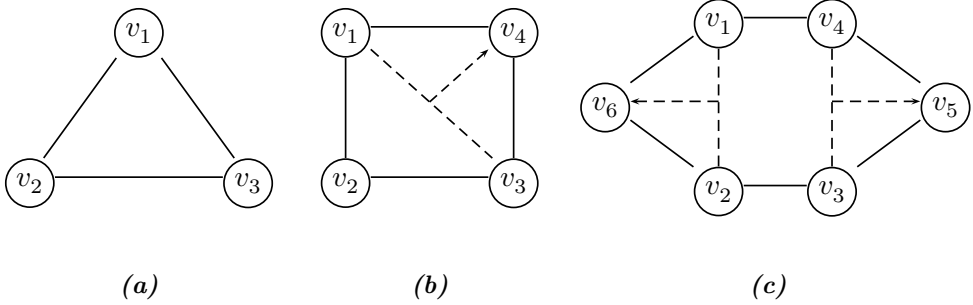


Рис. 2.56.

Разбиения ребер показаны на рисунке 2.56*b* и *c* штриховыми линиями. Справедлива следующая теорема.

Теорема 2.38 (Понтрягина — Куратовского). Граф планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных $K_{3,3}$ или K_5 .

Рассмотрим еще одно преобразование графа, позволяющее определить его планарность.

Определение 2.50. Операцией *стягивания ребра* (v, u) называют слияние смежных вершин u и v в новую вершину w , где ребра, инцидентные w , соответствуют ребрам, инцидентным либо u , либо v .

Граф G называется *стягиваемым к графу* G_1 , если последний получается из G в результате некоторой конечной последовательности операций стягиваний ребер.

Граф, получившийся в результате стягивания ребра (v, u) в графе G , будем обозначать как $G/(uv)$.

Справедлив альтернативный критерий планарности графа.

Теорема 2.39 (критерий планарности Вагнера). Граф планарен тогда и только тогда, когда он не содержит подграфов, стягиваемых к графам K_5 или $K_{3,3}$.

Пример 2.47. Используя критерий Вагнера, покажем, что граф Петерсена, изображенный на рисунке 2.57*a*, не является планарным. Граф Петерсена уже ранее встречался в примере 2.41 на изоморфизм.

Стянем ребра $(u_1, v_1), (u_2, v_2), (u_3, v_3), (u_4, v_4), (u_5, v_5)$ — на рисунке 2.57*a* они светло-серого цвета. При этом произойдет переход ребер:

$$\begin{aligned} (u_1, u_2) &\rightarrow (v_1, v_2), (u_2, u_2) \rightarrow (v_2, v_2), (u_3, u_4) \rightarrow (v_3, v_4), \\ (u_4, u_5) &\rightarrow (v_4, v_5), (u_5, u_1) \rightarrow (v_5, v_1). \end{aligned}$$

На рисунке 2.57а эти ребра отмечены темно-серым цветом. Тогда граф Петерсена стягивается к графу K_5 (рисунок 2.57б). По критерию Вагнера получаем непланарность.

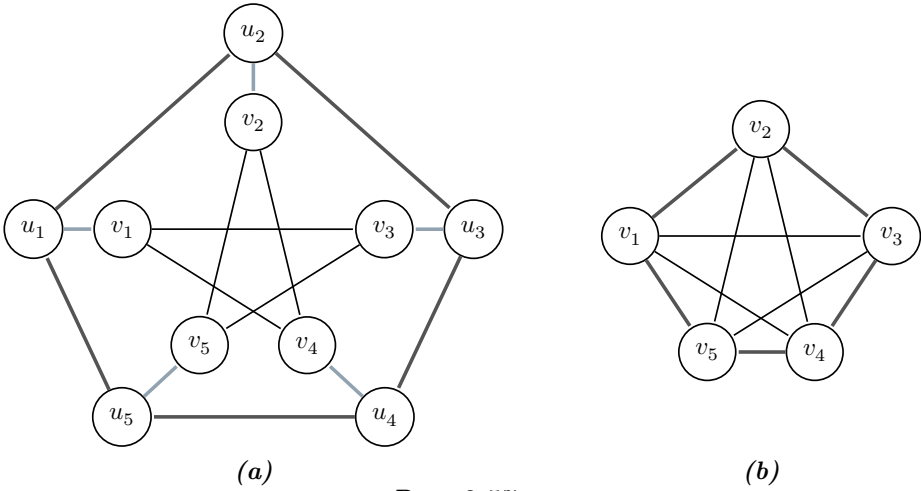


Рис. 2.57.

2.1.16. Раскраска графов

Определение 2.51. Пусть $G = (V, E)$ — некоторый граф. Пусть $\{1, 2, \dots, k\}$ — множество «цветов». Отображение $f : V \rightarrow \{1, 2, \dots, k\}$ называют *вершинной k -раскраской* графа G . Такую k -раскраску называют *правильной*, если для любого ребра $(v_1, v_2) \in E$ справедливо $f(v_1) \neq f(v_2)$, т. е. смежные вершины получают различную окраску.

Граф G называют *k -раскрашиваемым*, если для него существует правильная k -раскраска.

Наименьшее k , для которого граф G является k -раскрашиваемым, называют *хроматическим числом* графа G (обозначение: $\chi(G)$). Если $\chi(G) = k$, то граф G называют *k -хроматическим*.

Замечание 2.39

Пусть граф G получен из мультиграфа \widehat{G} (см. замечание 2.1) заменой всех ребер, соединяющих две данные вершины, на одно ребро. Очевидно, что вершинная раскраска мультиграфа \widehat{G} правильная тогда и только тогда, когда соответствующая вершинная раскраска является правильной для графа G .

Существует ряд задач, которые приводят к вершинной раскраске графа (задачи составления расписаний, обслуживания и др.).

Заметим сначала, что для любого натурального n существует граф \mathbf{G} , такой, что $\chi(\mathbf{G}) = n$. Примером такого графа является K_n . Очевидно, что $\chi(K_n) = n$.

Ясно, что 1-хроматические графы — это графы, состоящие из изолированных вершин, 2-хроматические графы — это двудольные графы и только они.

Заметим, что хроматическое число для графа Петерсена (рисунок 2.57a) равно 3.

В настоящее время нет описания k -хроматических графов при $k > 3$. Нет также эффективных алгоритмов нахождения хроматического числа графа. Однако имеются хорошие оценки хроматического числа.

Теорема 2.40 (Визинг). Справедливо неравенство

$$\chi(\mathbf{G}) \leq \Delta(\mathbf{G}) + 1,$$

где $\Delta(\mathbf{G})$ — максимальная степень вершин графа \mathbf{G} .

Доказательство. Проводим индукцию по n — числу вершин графа. Выберем произвольную вершину $v \in \mathbf{V}$ и удалим ее вместе с инцидентными ей ребрами. Получим граф \mathbf{G}' , для которого $\Delta(\mathbf{G}') \leq \Delta(\mathbf{G})$.

Число вершин у графа \mathbf{G}' равно $n - 1$, и, следовательно, по индукционному предположению для \mathbf{G}' имеется $(\Delta(\mathbf{G}') + 1)$ -раскраска, а значит, и $(\Delta(\mathbf{G}) + 1)$ -раскраска. Тогда $(\Delta(\mathbf{G}) + 1)$ -раскраску для \mathbf{G} можно получить так: окрасим вершину v в цвет, отличный от цветов вершин, смежных с ней, число которых не больше $\Delta(\mathbf{G})$. ■

Для планарных графов данную оценку можно уточнить.

Теорема 2.41 (Хивуд). Для любого планарного графа $\chi(\mathbf{G}) \leq 5$.

Доказательство. Покажем, что в планарном графе существует вершина степени не выше 5.

Предположим, что все вершины имеют степень не меньше, чем 6. Согласно теореме Эйлера $\sum \deg(v) = 2q$, следовательно, $2q \geq 6p$ или $q \geq 3p$. Последнее неравенство противоречит (2.12).

Теперь докажем утверждение теоремы индукцией по числу вершин графа $m = |\mathbf{V}|$.

Для графа с числом вершин, не превосходящим пяти, утверждение очевидно. Считая теорему доказанной для графов с числом вершин, не превосходящим m , рассмотрим граф с $m + 1$ вершиной. По доказанному выше у графа есть вершина v_0 степени не более пяти. Рассмотрим два случая.

Пусть степень вершины v_0 меньше пяти. Удалим из графа эту вершину вместе со всеми входящими в нее ребрами; получим граф с m вершинами. По предположению индукции для него существует правильная раскраска в 5 или менее цветов. По этой раскраске можно построить правильную раскраску исходного графа с тем же свойством — достаточно раскрасить вершину v_0 в цвет, отличный от цветов соседних вершин (которых не более четырех).

Перейдем к случаю, когда степень вершины v_0 равна пяти. Рассмотрим смежные с ней вершины. Среди них найдутся две, не соединенные ребром, иначе граф содержал бы полный граф K_5 , что противоречит предположению о планарности графа.

Обозначим эти вершины v_1 и v_2 . Удалим из графа вершину v_0 вместе с пятью смежными с ней ребрами, получим граф G_1 . Соединим в нем вершины v_1 и v_2 в одну. Получаем граф G_2 с $m - 1$ вершиной.

По индукционному предположению для него существует правильная раскраска не более чем в 5 цветов. Заметим, что получившийся граф G_2 не содержит петель, так как вершины v_1 и v_2 в исходном графе не были смежными.

При соединении вершин v_1 и v_2 в одну при наличии у них общих смежных вершин могут появиться кратные ребра (рисунок 2.58), но согласно замечанию 2.39 это не влияет на существование правильной раскраски.



Рис. 2.58.

Очевидно, что из этой раскраски получается правильная раскраска графа G_1 , так как вершины v_1 и v_2 не смежны, то их можно раскрасить в один цвет. Далее, по этой раскраске легко получить раскраску для исходного графа — смежные с v_0 вершины раскрашены не более чем в четыре цвета. ■

Замечание 2.40

Легко привести пример планарного графа, для которого $\chi(G) = 4$, например полный граф K_4 (рисунок 2.52b).

Попытки построить планарный граф, для которого недостаточно четырех цветов, длительное время не приводили к успеху. Поэтому, естественно, возникла гипотеза четырех красок. Она была сформулирована Артуром Кэли в 1878 г. и утверждала, что $\chi(G) \leq 4$.

Рассмотренная выше теорема Хивуда была доказана в 1890 г. Подтвердить гипотезу о четырех красках, не удавалось еще почти 90 лет.

Решение было предложено в 1978 г. Кеннетом Appelем и Вольфгангом Хакеном. Соответствующее доказательство было получено с применением ЭВМ, и пока оно не имеет проверки в традиционном понимании.

Задача определения хроматического числа и построения минимальной правильной раскраски для произвольного графа является достаточно сложной. Понятно, что для того, чтобы найти хроматическое число n -вершинного графа, нужно рассмотреть конечное число ситуаций.

Действительно, всего имеется k^n различных способов раскраски графа G в k цветов. Перебрав их, мы либо найдем правильную k -раскраску графа G либо убедимся, что ее не существует. Однако даже для сравнительно небольших n и k такой перебор нельзя провести за приемлемое время.

Рассмотрим простой эвристический алгоритм построения правильной раскраски, близкой к оптимальной, часто его называют алгоритмом *последовательной* раскраски.

Алгоритм 2.15. Последовательная раскраска

```
// Упорядочиваем вершины графа в список  $L$  в порядке невозрастания степеней
 $L = \{v_1, v_2, \dots, v_n\}$ 
 $p \leftarrow 1$  // начальный цвет окраски

// Окрашиваем первую вершину списка  $L$  в цвет  $p$ . Удаляем ее из списка
 $\text{color}(v_1) \leftarrow p$ 
 $L \leftarrow L \setminus \{v_1\}$ 
while  $L \neq \emptyset$  do
  for each  $u \in L$ , несмежной с вершинами, уже окрашенными в цвет  $p$  do
    // Окрашиваем вершину  $u$  в цвет  $p$ . Удаляем ее из списка  $L$ 
     $\text{color}(u) \leftarrow p$ 
     $L \leftarrow L \setminus \{u\}$ 
  end for
   $p \leftarrow p + 1$  // выбираем следующий цвет
end while
```

Замечание 2.41

Рассмотренный алгоритм 2.15 можно модифицировать. Для этого после каждого шага нужно заново упорядочить список еще неокрашенных вершин по невозрастанию их степеней.

При работе алгоритма предполагалось, что если две вершины имеют одинаковые степени, то порядок раскраски таких вершин случаен. Такие вершины можно также упорядочить, но уже по двойным степеням. Двойную степень вершины можно определить как сумму степеней инцидентных ей вершин. Этот процесс можно продолжать и далее.

Пример 2.48. Применим алгоритм 2.15 для раскраски графа, изображенного на рисунке 2.59а.

1. Упорядочим вершины графа в список в порядке невозрастания степеней (в скобках — степень вершины) Вершины с равными валентностями

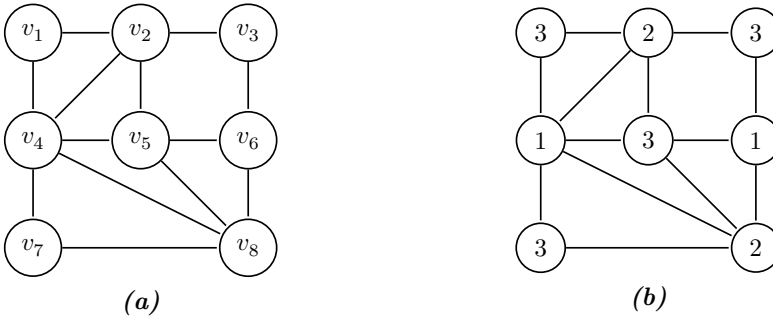


Рис. 2.59.

записываем в порядке возрастания номеров:

$$L = \{v_4(5), v_2(4), v_5(4), v_8(4), v_6(3), v_1(2), v_3(2), v_7(2)\}.$$

2. Окрашиваем в цвет 1 вершину v_4 и несмежную с ней, еще не окрашенную вершину v_6 . Удаляем их из списка, переходим к цвету 2. Вершину v_3 на этом шаге нельзя покрасить в цвет 1 после окраски в цвет 1 смежной с ней вершины v_6 .

3. Окрашиваем в цвет 2 вершину v_2 и несмежную с ней, еще не окрашенную вершину v_8 . Удаляем их из списка, переходим к цвету 3. Вершину v_7 на этом шаге нельзя покрасить в цвет 2 после окраски в цвет 2 смежной с ней вершины v_8 .

4. Окрашиваем в цвет 3 несмежные вершины v_5, v_1, v_3, v_7 . Удаляем их из списка. Список пуст (все вершины графа окрашены), алгоритм заканчивает работу.

Легко заметить, что в данном примере построена минимальная вершинная раскраска ($\chi(\mathbf{G}) = 3$). Результат работы алгоритма представлен на рисунке 2.59b.

Рассмотрим пример задачи, для решения которой нужно найти хроматическое число и наименьшую правильную вершинную раскраску графа. Это задача составления оптимального расписания.

Нужно прочесть ряд лекций группам студентов. Некоторые из лекций не могут читаться одновременно, например потому, что их читает один и тот же лектор или их надо читать одной и той же группе студентов. Требуется составить расписание занятий так, чтобы чтение всех лекций заняло минимально возможное время.

С точки зрения раскраски графов нужно построить граф \mathbf{G} , в котором вершины соответствуют лекциям. Вершины графа смежны тогда и только тогда, когда соответствующие лекции не могут читаться одновременно.

Очевидно, что любая правильная k -раскраска графа G определяет допустимое расписание. При этом вершины, раскрашенные i -м цветом, дают список лекций, которые надо читать на i -й паре. И наоборот, любое допустимое расписание определяет правильную раскраску графа G . Таким образом, составление оптимального расписания сводится к нахождению наименьшей раскраски построенного графа.

Пример 2.49. Необходимо провести занятия по математике (M), физике (F), иностранному языку (L) и истории России (I) в группах G_1 и G_2 . Занятия проводятся преподавателями T_1 и T_2 — по одному занятию по каждому предмету. Таблица 2.20 указывает, какие занятия надо провести в группах и какие преподаватели их могут провести.

Таблица 2.20.

Лекции	Группы		Преподаватели	
	G_1	G_2	T_1	T_2
Математика	+		+	
Физика	+	+	+	
Иностранный язык		+		+
История России	+	+		+

Требуется найти минимальное число пар, в которые можно *уложить* все занятия, и составить соответствующее расписание.

Построим граф с вершинами M_1, F_1, I_1, I_2 и L_2 (буква соответствует обозначению предмета, а цифра — номеру группы). Соединим ребрами вершины, соответствующие парам, которые нельзя проводить одновременно. Получим граф, изображенный на рисунке 2.60а. Применим для его раскраски алгоритм 2.15.

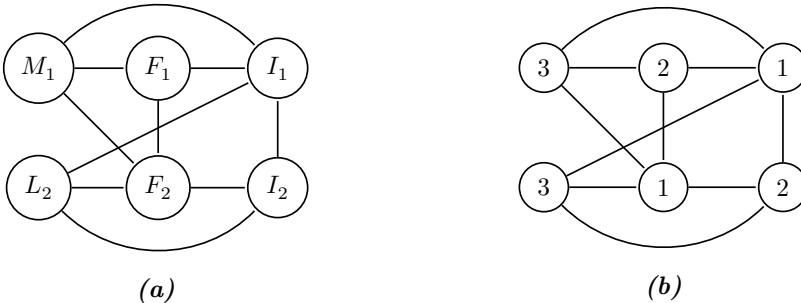


Рис. 2.60.

1. Упорядочим вершины графа в список в порядке невозрастания степеней:

$$L = \{I_1(4), F_2(4), F_1(3), M_1(3), L_2(3), I_2(3)\}.$$

2. Окрашиваем в цвет 1 вершины I_1 и F_2 .

3. Окрашиваем в цвет 2 вершины F_1 и I_2 .

4. Окрашиваем в цвет 3 оставшиеся вершины M_1 и L_2 . Все вершины графа окрашены, алгоритм заканчивает работу.

Результат работы алгоритма представлен на рисунке 2.60*b*.

Соответственно оптимальное расписание содержит три пары занятий (таблица 2.21).

Таблица 2.21.

Номер пары	Группа G_1	Группа G_2
1	История России (T_2)	Физика (T_1)
2	Физика (T_1)	История России (T_2)
3	Математика (T_1)	Иностранный язык (T_2)

Пример 2.50. Применим алгоритм 2.15 для раскраски эйлерова графа из примера 2.14 — рисунок 2.61*a*.

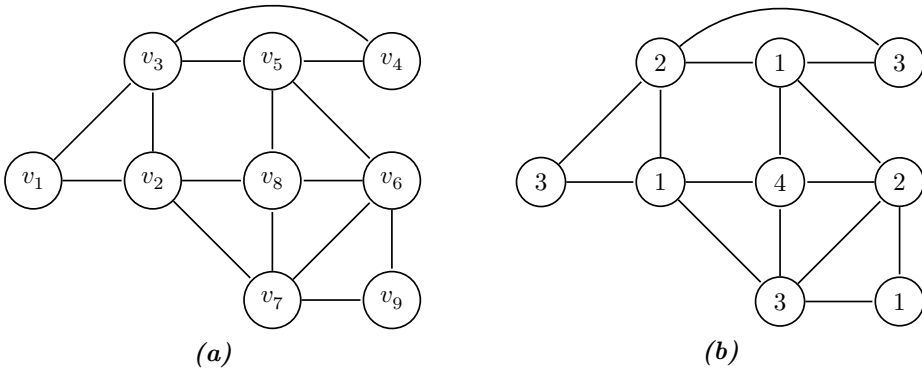


Рис. 2.61.

1. Упорядочим вершины графа в список в порядке невозрастания степеней (в скобках — степень вершины). Как и ранее, вершины с равными валентностями записываем в порядке возрастания номеров:

$$L = \{v_2(4), v_3(4), v_5(4), v_6(4), v_7(4), v_8(4), v_1(2), v_4(2), v_9(2)\}.$$

2. Окрашиваем в цвет 1 вершину v_2 и последовательно несмежные с ней, еще не окрашенные вершины v_5 и v_9 . Удаляем их из списка, переходим к цвету 2.

3. Окрашиваем в цвет 2 вершину v_3 и несмежную с ней, еще не окрашенную вершину v_6 . Удаляем их из списка, переходим к цвету 3.

4. Окрашиваем в цвет 3 вершину v_7 и последовательно несмежные с ней, еще не окрашенные вершины v_1 и v_4 . Удаляем их из списка, переходим к цвету 4.

5. Окрашиваем в цвет 4 последнюю оставшуюся вершину v_8 . Удаляем ее из списка. Список пуст (все вершины графа окрашены), алгоритм заканчивает работу.

Результат работы алгоритма представлен на рисунке 2.61b.

Наряду с раскраской вершин графа существует еще одно понятие раскраски.

Определение 2.52. *Реберной раскраской* графа $G = (V, E)$ в k цветов называется отображение $\varphi : E \rightarrow \{1, \dots, k\}$.

Раскраска ребер графа называется *правильной*, если любые два ребра, инцидентные одной вершине, покрашены в разные цвета.

Наименьшее количество цветов, для которого существует правильная раскраска ребер графа G , называется его *хроматическим индексом* или *реберным хроматическим числом* и обозначается $\chi'(G)$.

Пример 2.51. Для графа Петерсена (рисунок 2.57a) хроматический индекс равен 4.

Лемма 2.6. Для хроматического индекса графа G справедлива следующая оценка снизу:

$$\Delta(G) \leq \chi'(G).$$

Доказательство. Рассмотрим вершину, имеющую максимальную степень в графе. Этой вершине инцидентно ровно $\Delta(G)$ ребер, которые при правильной реберной раскраске все должны иметь различные цвета. Что и доказывает утверждение. ■

Более точную оценку дает теорема Визинга.

Теорема 2.42 (Визинга о реберной раскраске). Для любого графа G выполнена оценка сверху.

$$\chi'(G) \leq \Delta(G) + 1.$$

Для двудольного графа оценку Визинга можно улучшить.

Теорема 2.43 (Кенига о реберной раскраске). Пусть $G = (V, E)$ — двудольный граф. Тогда

$$\chi'(G) = \Delta(G) = \max\{\deg(v) \mid v \in V\}.$$

Для нахождения реберной раскраски существует *почти оптимальный* алгоритм полиномиального времени Мисры и Гриза [79]. Он дает раскраску в $\Delta(G) + 1$ цветов. По теореме Визинга это на один цвет больше, чем оптимальная раскраска, а для некоторых графов это значение является оптимальным.

Заметим, что задача реберной раскраски сводится к задаче вершинной раскраски (и обратно) с помощью понятия *реберного графа*.

Определение 2.53. *Реберным графом* графа G (обозначается $L(G)$) называется граф, любая вершина которого представляет собой ребро исходного графа G , и две вершины графа $L(G)$ смежны тогда и только тогда, когда соответствующие им ребра смежны в графе G .

Пример 2.52. На рисунке 2.62*a* представлен исходный граф, на рисунке 2.62*b* — реберный.

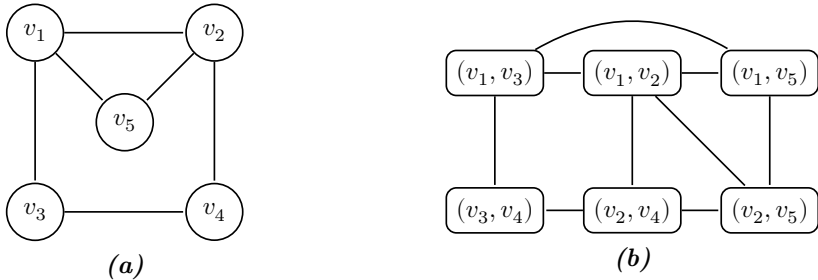


Рис. 2.62.

Теорема 2.44. Справедливы следующие свойства реберного графа.

1. Реберный граф связного графа связан.
2. Хроматический индекс графа G равен хроматическому числу его реберного графа $L(G)$:

$$\chi'(G) = \chi(L(G)).$$

3. Если граф эйлеров, то его реберный граф гамильтонов.

Доказательство. Все утверждения очевидным образом следуют из определения реберного графа. ■

2.1.17. Хроматические многочлены

Определение 2.54. Пусть $G = (V, E)$ — некоторый граф и t — заданное число цветов. Число способов правильной вершинной t -раскраски графа G называется его *хроматическим многочленом* и обозначается $P(G, t)$.

Рассмотрим хроматические многочлены для различных графов.

1. Хроматический многочлен нуля-графа с n вершинами равен t^n , так как все вершины этого графа можно раскрасить в любой из t заданных цветов.

$$P(Z_n, t) = t^n. \quad (2.14)$$

2. Пусть K_n — полный n -вершинный граф. Тогда

$$P(K_n, t) = t(t-1)(t-2)\dots(t-(n-1)).$$

Действительно, первую вершину полного графа можно раскрасить t цветами, следующую — $t-1$ цветом и т. д.

3. Пусть граф T_n — дерево с n вершинами. Тогда

$$P(T_n, t) = t(t-1)^{n-1}. \quad (2.15)$$

Докажем равенство (2.15) индукцией по числу вершин дерева. Для $n = 1, 2$ равенство очевидно, база индукции установлена.

Пусть (2.15) справедливо для всех деревьев T_{n-1} . При $n \geq 2$ согласно теореме 2.23 у дерева существует не менее двух висячих вершин. Пусть v — висячая вершина дерева. Удалим вершину v вместе с инцидентным ей ребром из дерева T_n . Согласно лемме 2.3 после этой операции граф останется деревом. В силу индукционного предположения

$$P(T_n \setminus \{v\}, t) = t(t-1)^{n-2}.$$

Вершину v можно раскрасить $t-1$ способом, так как ее цвет должен отличаться только от цвета единственной смежной вершины. Таким образом,

$$P(T_n, t) = (t-1)P(T_n \setminus \{v\}, t) = (t-1)t(t-1)^{n-2} = t(t-1)^{n-1}.$$

4. Пусть S_n — цепь графа, состоящая из n вершин (см. определение 2.4). Тогда

$$P(S_n, t) = t(t-1)^{n-1}. \quad (2.16)$$

Рассмотрим процесс раскраски S_n . Первую вершину можно покрасить в один из t цветов, вторую и последующие — в один из $t - 1$ цветов (т. е. так, чтобы цвет не совпадал с предыдущей вершиной).

5. Пусть C_n — цикл, состоящий из n вершин (см. определение 2.4). Тогда

$$P(C_n, t) = (t - 1)^n + (-1)^n(t - 1). \quad (2.17)$$

Докажем формулу (2.17) индукцией по количеству вершин цикла. Пусть $n = 3$, тогда цвета всех вершин должны быть различны:

$$P(C_3, t) = t(t - 1)(t - 2) = (t - 1)((t - 1)^2 - 1) = (t - 1)^3 - (t - 1).$$

База индукции установлена.

Число правильных раскрасок цикла C_n равно числу правильных раскрасок цепочки S_n , за вычетом числа тех, где первая и последняя вершина S_n имеют один цвет. Но последнее число, очевидно, равно $P(C_{n-1}, t)$.

Используя индукционное предположение для $P(C_{n-1}, t)$ и формулу (2.16), получаем

$$P(C_n, t) = t(t - 1)^{n-1} - (t - 1)^{n-1} - (-1)^{n-1}(t - 1).$$

Откуда очевидным образом следует (2.17).

Для хроматических многочленов справедливо рекуррентное соотношение (иногда его называют *формулой удаления и стягивания*).

Теорема 2.45 (Зыков). Пусть (u, v) — ребро графа \mathbf{G} . Тогда

$$P(\mathbf{G}, t) + P(\mathbf{G}/(uv), t) = P(\mathbf{G} \setminus \{(u, v)\}, t). \quad (2.18)$$

Доказательство. Выражение $P(\mathbf{G} \setminus \{(u, v)\}, t)$ в правой части равенства (2.18) есть число возможных правильных раскрасок графа \mathbf{G} с удаленным ребром (u, v) , следовательно, в нем вершины u и v могут иметь как одинаковый, так и разные цвета.

Рассмотрим слагаемые в левой части равенства (2.18). Значение $P(\mathbf{G}, t)$ — число возможных правильных раскрасок графа \mathbf{G} , где смежные вершины u и v имеют разные цвета. Во втором слагаемом эти вершины стянуты в одну, следовательно, имеют один цвет. ■

Теорема 2.45 дает основание для *хроматической редукции по пустым графам*. Под этим процессом понимается представление графа в виде нескольких пустых графов, сумма хроматических полиномов которых равна хроматическому полиному графа.

Перепишем равенство (2.18) в другом виде:

$$P(\mathbf{G}, t) = P(\mathbf{G} \setminus \{(u, v)\}, t) - P(\mathbf{G}/(uv), t), \quad (2.19)$$

и рассмотрим этот процесс на примере.

Пример 2.53. На рисунке 2.63 исходный граф \mathbf{G} схематично представлен в виде $\mathbf{G} = \mathbf{G}_1 - \mathbf{G}_2$ согласно (2.19).

Замечание 2.42

Знаки арифметических операций (здесь и далее в примере) относятся не к графам, а к их хроматическим многочленам.

Согласно (2.19) граф $\mathbf{G}_1 = \mathbf{G} \setminus \{(a, c)\}$, а граф $\mathbf{G}_2 = \mathbf{G}/(ac)$. Мы сделали один шаг в процессе хроматической редукции графа.

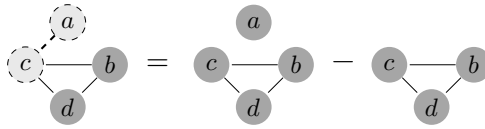


Рис. 2.63.

Продолжим процесс для графов \mathbf{G}_1 и \mathbf{G}_2 . Для графа \mathbf{G}_1 применим соотношение (2.19) для ребра (b, d) , а затем для ребра (c, d) . Получаем разложение (рисунок 2.64)

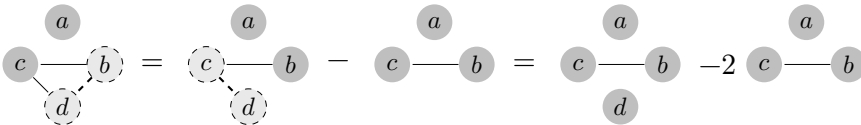


Рис. 2.64.

Применяя еще раз формулу (2.19) для ребра (c, b) , имеем:

$$P(\mathbf{G}_1, t) = P(Z_4, t) - P(Z_3, t) - 2(P(Z_3, t) - P(Z_2, t)) = t^4 - 3t^3 + 2t^2. \quad (2.20)$$

Проведем аналогичные преобразования для графа \mathbf{G}_2 . Как и ранее, последовательно используя соотношение (2.19) для ребер (b, d) , (c, d) и (b, c) , получаем разложение:

$$P(\mathbf{G}_2, t) = P(Z_3, t) - P(Z_2, t) - 2(P(Z_2, t) - P(Z_1, t)) = t^3 - 3t^2 + 2t. \quad (2.21)$$

Подставляя в (2.19) разложения (2.20) и (2.21), получаем окончательный результат:

$$P(\mathbf{G}, t) = t^4 - 4t^3 + 5t^2 - 2t.$$

Рассмотренный в примере 2.53 процесс хроматической редукции к нуль-графам по формуле (2.19) часто в литературе называют *алгоритмом Зыкова*.

Рассмотрим еще одно преобразование графа, лежащее в основе альтернативного подхода вычисления хроматического многочлена.

Определение 2.55. Операцией *стягивания* или *отождествления* несмежных вершин u и v называется замена их новой вершиной w , при этом все ребра, инцидентные u и v , становятся инцидентными новой вершине. В качестве новой вершины w часто рассматривают одну из вершин u или v .

Граф, получившийся в результате стягивания вершин u и v в графе G , будем обозначать как G/uv .

В терминах введенного преобразования, используя рассуждения для теоремы Зыкова (теорема 2.45), легко получить соотношение, аналогичное (2.19). Справедливо равенство:

$$P(G, t) = P(G \cup \{(u, v)\}, t) + P(G/uv, t). \quad (2.22)$$

На основании равенства (2.22) можно провести вычисление хроматического многочлена. Этот процесс носит название *хроматической редукции по полным графам*. Под этим понимается представление графа в виде нескольких полных графов, сумма хроматических полиномов которых равна хроматическому полиному графа.

Рассмотрим этот процесс на примере графа на рисунке 2.63 из примера 2.53.

Пример 2.54. Как и в примере 2.53, знаки арифметических операций на рисунках будут относиться не к графам, а к их хроматическим многочленам.

Применяя (2.22) для ребра (a, d) , получаем первый шаг редукции (рисунок 2.65).

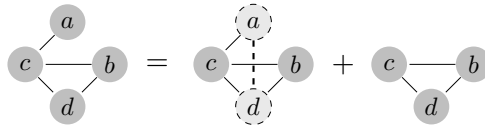


Рис. 2.65.

Второй граф в сумме на рисунке 2.65 справа — полный граф K_3 , его преобразовывать не нужно. Продолжим процесс для первого слагаемого. Применим (2.22) для ребра (a, b) , получим следующий шаг редукции (рисунок 2.66).

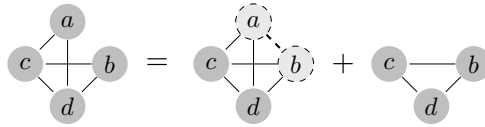


Рис. 2.66.

Таким образом,

$$P(\mathbf{G}, t) = P(K_4, t) + 2P(K_3, t) = t(t-1)(t-2)(t-3) + 2t(t-1)(t-2) = t^4 - 4t^3 + 5t^2 - 2t.$$

Замечание 2.43

В зависимости от числа ребер графа для вычисления хроматического многочлена можно использовать соотношение (2.19) или (2.22). Если граф близок к полному, оптимальнее использовать редукцию по полным графам. Для «разреженных» графов с малым числом ребер лучше применять разложение по пустым графам.

Пример 2.55. Рассмотрим еще один пример. У графа на рисунке 2.67 до полного графа K_5 «не хватает» 4 ребер. Используем редукцию по полным графам.

Будем использовать обозначения предыдущих примеров 2.53 и 2.54.

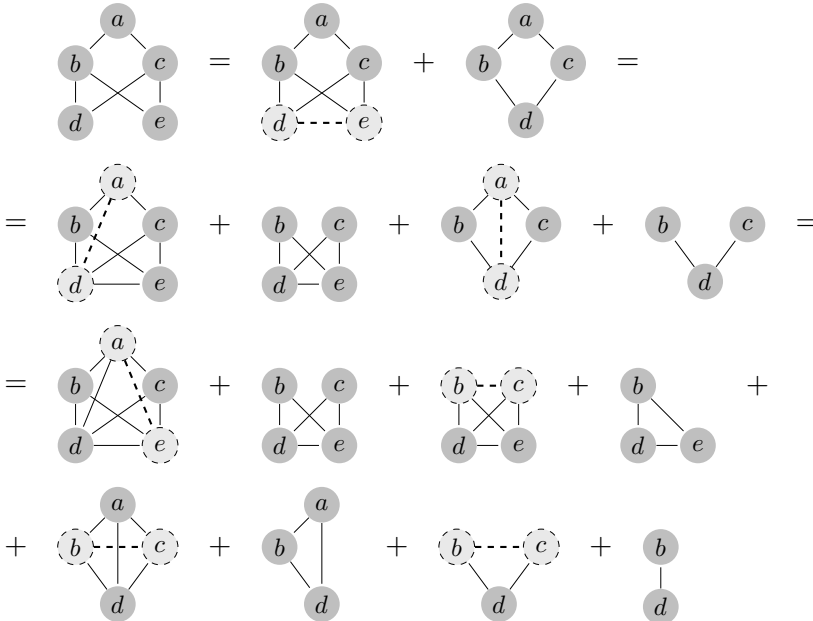


Рис. 2.67.

Применим в последнем равенстве стягивание для пары несмежных вершин b, c второго графа (справа от K_5). Тогда он преобразуется в два графа: $K_4 + K_3$. Окончательно получаем

$$P(\mathbf{G}, t) = P(K_5, t) + 3P(K_4, t) + 4P(K_3, t) + P(K_2, t).$$

Хроматический многочлен графа содержит некоторую дополнительную информацию о самом графе.

Теорема 2.46. Пусть хроматический многочлен графа \mathbf{G} имеет вид:

$$P(\mathbf{G}, t) = a_n t^n + a_{n-1} t^{n-1} + \dots + a_1 t + a_0.$$

Тогда справедливы следующие свойства хроматических многочленов.

1. Свободный член хроматического многочлена a_0 равен 0.
2. Старший коэффициент хроматического многочлена a_n равен 1.
3. Степень хроматического многочлена равна числу вершин графа.

4. Пусть в графе \mathbf{G} есть висячая вершина v . Обозначим через \mathbf{G}_1 граф, полученный удалением из исходного графа вершины v и инцидентного ей ребра. Тогда

$$P(\mathbf{G}, t) = (t - 1)P(\mathbf{G}_1, t). \quad (2.23)$$

5. Пусть в графе \mathbf{G} есть вершина u : $\deg(u) = 2$ и смежные с u вершины смежны между собой (образуют «треугольник»). Обозначим через \mathbf{G}_1 граф, полученный удалением из исходного графа вершины u и обоих инцидентных ей ребер. Тогда

$$P(\mathbf{G}, t) = (t - 2)P(\mathbf{G}_1, t). \quad (2.24)$$

6. Пусть C_1, \dots, C_n — компоненты связности графа \mathbf{G} . Тогда

$$P(\mathbf{G}, t) = \prod_{i=1}^n P(C_i, t). \quad (2.25)$$

7. Знаки коэффициентов хроматического многочлена чередуются.

8. Число ребер в графе равно $|a_{n-1}|$.

9. Хроматическое число графа — наименьшее натуральное число, не являющееся корнем хроматического многочлена.

$$\chi(\mathbf{G}) = \min \{t : P(\mathbf{G}, t) > 0\}.$$

Доказательство. Количество способов правильно раскрасить граф в 0 цветов очевидно равно 0, т. е. $P(\mathbf{G}, 0) = 0$. Пункт 1 установлен.

Для доказательства пункта 2 воспользуемся формулой редукции по полным графам (2.22). Применяя это соотношение рекурсивно, получим

$$\begin{aligned} P(\mathbf{G}, t) &= P(K_n, t) + a_{n-1}P(K_{n-1}, t) + a_{n-2}P(K_{n-2}, t) + \dots = \\ &= t^n + a_{n-1}t^{n-1} + a_{n-2}t^{n-2} + \dots, \end{aligned}$$

что и доказывает требуемое.

Пункт 3 также очевидным образом следует из формулы редукции по полным графам (2.22).

Соотношения (2.23)–(2.25) для пунктов 4–6 легко получаются из определения 2.54 для хроматического многочлена.

Пункт 7 докажем индукцией по числу вершин графа. Для графа \mathbf{G} из одной вершины $P(\mathbf{G}, t) = t$. База индукции установлена.

Пусть утверждение верно для всех n -вершинных графов. Рассмотрим графы из $n + 1$ вершины. Индукционный переход докажем индукцией по числу ребер графа \mathbf{G} .

Если \mathbf{G} не содержит ребер, то $P(\mathbf{G}, t) = P(Z_{n+1}, t) = t^{n+1}$ согласно (2.14) и утверждение справедливо. Рассмотрим $(n + 1, m + 1)$ -граф \mathbf{G}_1 (определение 2.1) и его ребро (u, v) .

Проведем стягивание ребра (u, v) (см. определение 2.50) и воспользуемся для графа \mathbf{G}_1 формулой (2.19). Тогда

$$P(\mathbf{G}_1, t) = P(\mathbf{G}_2, t) - P(\mathbf{G}_3, t), \quad \mathbf{G}_2 = \mathbf{G}_1 \setminus \{(u, v)\}, \quad \mathbf{G}_3 = \mathbf{G}_1 / (uv).$$

В графе \mathbf{G}_2 на одно ребро меньше, а в графе \mathbf{G}_3 на одну вершину меньше, чем в исходном графе \mathbf{G}_1 . Следовательно, для них справедливо индукционное предположение.

$$\begin{aligned} P(\mathbf{G}_2, t) &= t^{n+1} - a_n t^n + a_{n-1} t^{n-1} + \dots, \quad a_i \geq 0, i \in 0 : n + 1, \\ P(\mathbf{G}_3, t) &= t^n - b_{n-1} t^{n-1} + b_{n-2} t^{n-2} + \dots, \quad b_j \geq 0, j \in 0 : n. \end{aligned}$$

Тогда

$$P(\mathbf{G}_1, t) = t^{n+1} - (a_n + 1)t^n + (a_{n-1} + b_{n-1})t^{n-1} + \dots, \quad (2.26)$$

что и доказывает пункт 7.

Из равенства (2.26) следует, что при добавлении одного ребра к графу второй коэффициент его хроматического многочлена увеличивается на 1. Поскольку для пустого графа второй коэффициент равен 0, то утверждение пункта 8 справедливо для любого графа.

Пункт 9 очевидным образом следует из определения 2.51 для хроматического числа и определения 2.54 для хроматического многочлена. ■

Замечание 2.44

Соотношения (2.23) и (2.24) теоремы 2.46 в некоторых случаях существенно упрощают вычисление хроматического многочлена.

Пример 2.56. Используя замечание 2.44, найдем хроматический многочлен графа на рисунке 2.68.

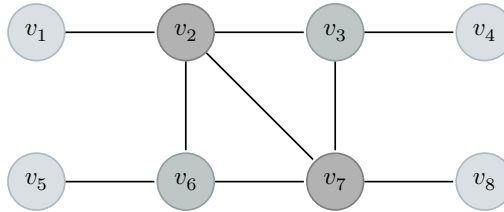


Рис. 2.68.

Используем соотношение (2.23) для висятых вершин v_1, v_4, v_5, v_8 , а затем соотношение (2.24) для «треугольников» v_6, v_2, v_7 и v_3, v_7, v_2 . Получаем граф K_2 из оставшихся вершин v_2, v_7 . Тогда

$$P(\mathbf{G}, t) = P(K_2, t)(t-1)^4(t-2)^2 = t(t-1)(t-1)^4(t-2)^2 = t(t-1)^5(t-2)^2.$$

Пример 2.57. Рассмотрим еще один аналогичный пример. Найдем хроматический многочлен графа на рисунке 2.69.

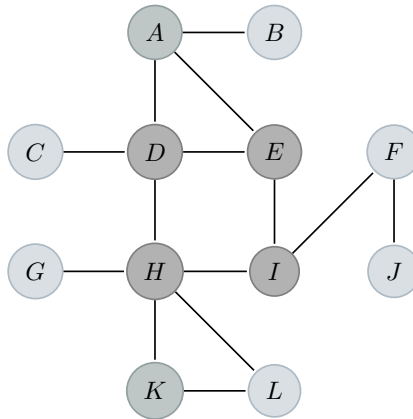


Рис. 2.69.

Используем соотношение (2.23) для висятых вершин B, C, G, J . Затем применим эту же формулу для вершины F , которая станет висятчей после удаления вершины J .

Далее применим формулу (2.24) для «треугольников» A, D, E и K, H, L и соотношение (2.23) для вершины L , которая станет висятчей после удаления вершины треугольника K .

Для оставшегося четырехвершинного цикла $D \rightarrow E \rightarrow I \rightarrow H \rightarrow D$ используем свойство (2.17). Окончательно получаем:

$$P(\mathbf{G}, t) = (t-1)^6(t-2)^2((t-1)^4 + (t-1)) = t(t-1)^7(t-2)^2(t^2 - 3t + 3).$$

Задачи и вопросы

1. Существует ли граф с набором степеней $(2, 2, 3, 3, 3)$?
2. В графе 100 вершин и 800 ребер. Есть ли в этом графе хотя бы одна вершина степени не меньше 16?
3. Докажите, что если число ребер графа с n вершинами ($n > 2$) больше C_{n-1}^2 , то граф связан.
4. Докажите, что если степень каждой вершины графа не меньше двух, то граф содержит цикл.
5. Докажите, что в каждом графе с n вершинами ($n \geq 2$) найдутся две вершины с одинаковыми степенями.
6. Существует ли полный граф с семью ребрами?
7. Определите число ориентированных графов с n вершинами (*без петель*), в которых каждая пара различных вершин соединена:
 - а) не более чем одним ребром;
 - б) точно одним ребром.
8. Справедливо ли утверждение: каждое дерево является двудольным графом?
9. В чем отличие понятий слабой и сильной связности для ориентированных графов?
10. Какое наибольшее число ребер может быть в несвязном графе с p вершинами?
11. Какое наименьшее число ребер может быть в связном графе с p вершинами?
12. Найти число компонент связности леса, имеющего 20 вершин и 10 ребер.
13. В группе 24 студента. Может ли быть так, что 9 из них имеют по 5 друзей среди студентов этой группы, 5 — по 4 друга и 10 — по 3 друга?

14. Можно ли построить 6-вершинный неориентированный граф, имеющий такой набор степеней вершин: $(2, 2, 2, 4, 5, 5)$.
15. Сколько ребер в связном графе с p вершинами, если в нем имеется единственный цикл?
16. Каково минимальное число вершин связного графа с p ребрами?
17. Определите число ориентированных графов с n вершинами (*без петель*), в которых каждая пара различных вершин соединена:
 - а) не более чем одним ребром;
 - б) точно одним ребром.
18. Каково наибольшее число ребер в двудольном графе с p вершинами?
19. Сколько ребер может быть у регулярного графа с 9 вершинами?
20. Сколько ребер имеет регулярный граф с p вершинами степени r ?
21. Сколько ребер в графе $K_{m,n}$?
22. Что такое транспонированный граф?
23. Что такое мост, точка сочленения?
24. Связный граф с p вершинами содержит единственный цикл. Сколько у него остовов?
25. Пусть все ребра связного графа с p вершинами имеют один и тот же вес α . Каков наименьший из весов его остовов?
26. Может ли связный граф иметь ровно два остова?
27. Докажите, что число стягивающих деревьев в графе K_n при $n > 1$ равно n^{n-2} .
28. Найдите хроматическое число связного графа, у которого p вершин и p ребер.
29. Опишите граф с хроматическим числом $\chi(\mathbf{G}) = 1$ или $\chi(\mathbf{G}) = 2$.
30. Что такое граф конденсации (граф Герца)?
31. Используя алгоритм Косарайю и Шарира, определите компоненты сильной связности в ориентированном графе на рисунке 2.70 и постройте соответствующий граф конденсации.
32. Пусть все степени вершин неориентированного графа равны 2. В каком случае этот граф будет эйлеровым?

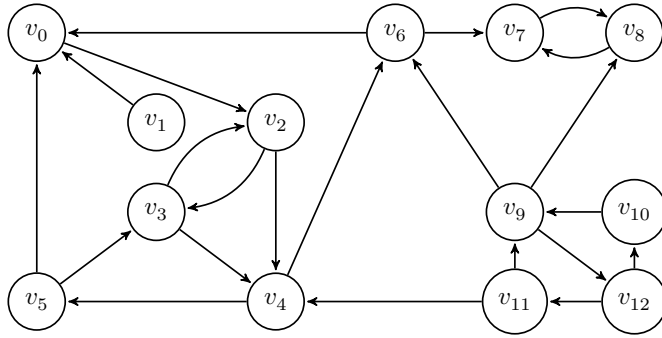


Рис. 2.70.

33. Может ли эйлеров граф содержать мост?

34. Степени валентности вершин связного графа: (2, 3, 6, 4, 3, 2, 2, 8). Существует ли в нем замкнутый (открытый) эйлеров путь?

35. Постройте замкнутую эйлерову цепь для графов на рисунке 2.71а-с.

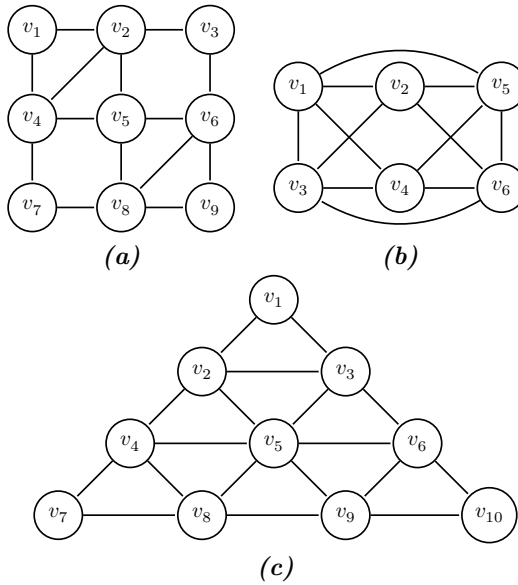


Рис. 2.71.

36. Модифицируйте алгоритм Флери для нахождения *всех* замкнутых эйлеровых цепей графа.

37. В чем разница между эйлеровым и гамильтоновым графом?

38. Постройте главные циклы и коциклы графов K_5 и $K_{3,3}$.

39. В полном двудольном графе 143 ребра. Определите число вершин в каждой доле V_1 и V_2 , если $|V_1| > 1$ и $|V_2| > 1$.

40. Сколько мостов имеет дерево с p ребрами?

41. Можно ли представить неориентированное дерево в виде двудольного графа?

42. Что представляет собой операция «излом ребра»? Для чего она используется?

43. Перечислите все попарно неизоморфные графы с четырьмя вершинами.

44. Определите, изоморфны ли графы на рисунке 2.72а и б.

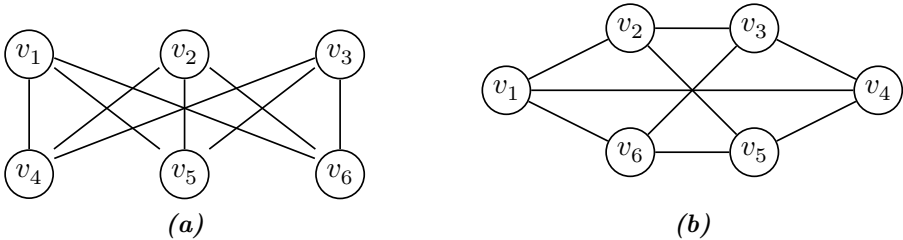


Рис. 2.72.

45. Выясните, какие из графов, изображенных на рисунке 2.73, изоморфны друг другу.

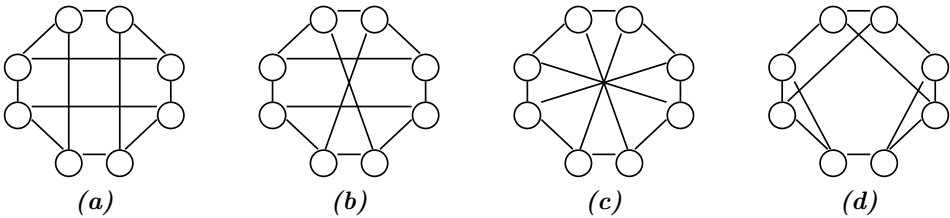


Рис. 2.73.

46. Являются ли графы K_4 и K_5 планарными?

47. Как по коду Прюфера вычислить степень каждой вершины, не восстанавливая само дерево?

48. Связный планарный граф, имеющий 7 ребер, изображен на плоскости так, что они не имеют пересечений. Граф разбивает плоскость на три части. Сколько в нем вершин?

49. Связный планарный граф, имеющий 7 вершин, изображен на плоскости так, что его ребра не имеют пересечений. Он разбивает плоскость на пять частей. Сколько в нем ребер?

50. Покажите, что формула Эйлера (см. теорему 2.37) следующим образом обобщается на случай плоского *несвязного* (n, m) -графа:

$$n - m + f = k + 1,$$

где k — число компонент связности графа.

51. Три поссорившихся соседа имеют три общих колодца. Можно ли провести непересекающиеся дорожки от каждого дома к каждому колодцу?

52. Что такое «правильная» раскраска графа?

2.2. Алгоритмы на графах

2.2.1. Построение наибольшего паросочетания

Рассмотрим задачу, которую называют задачей о супружеских парах. Неформальная постановка задачи такая. Имеется n юношей и m девушек. Про каждого юношу и каждую девушку известно, нравятся ли они друг другу. Нужно переженить как можно больше симпатизирующих друг другу пар.

Перед тем как поставить задачу формально, дадим необходимые определения.

Определение 2.56. *Паросочетанием* двудольного графа называют подмножество его ребер, никакие два из которых не являются смежными (не инциденты одной вершине).

Будем считать, что множество из одного ребра является паросочетанием.

Паросочетание P в графе G называется *наибольшим* или *максимальным по размеру*, если в G нет паросочетаний, число ребер в которых больше, чем в P . Число ребер в наибольшем паросочетании графа G называется его *числом паросочетания*.

Паросочетание P в графе G называется *максимальным по включению* или *максимальным по включению*, если оно не содержится ни в каком другом паросочетании этого графа, т. е. в него нельзя включить ни одного ребра несмежного ко всем ребрам паросочетания. Очевидно, что наибольшее паросочетание является максимальным по включению. Обратное не всегда верно.

Вершина v графа G называется *насыщенной* или *покрытой* в паросочетании P , если в P существует ребро, инцидентное v , в противном случае вершина называется *свободной*.

Паросочетание P называется *совершенным*, если все вершины графа G насыщены в P . Очевидно, что каждое совершенное паросочетание является наибольшим, но обратное не всегда верно.

Замечание 2.45

Определение 2.56 справедливо для любого графа (не обязательно двудольного). Мы будем рассматривать это понятие применительно к двудольным графам.

Пример 2.58. Рассмотрим введенные понятия на примере графа на рисунке 2.74а.

Множество ребер $\{(v_1, v_6), (v_2, v_5)\}$ является паросочетанием, но не максимальным по включению. Оно, например, включается в паросочетание

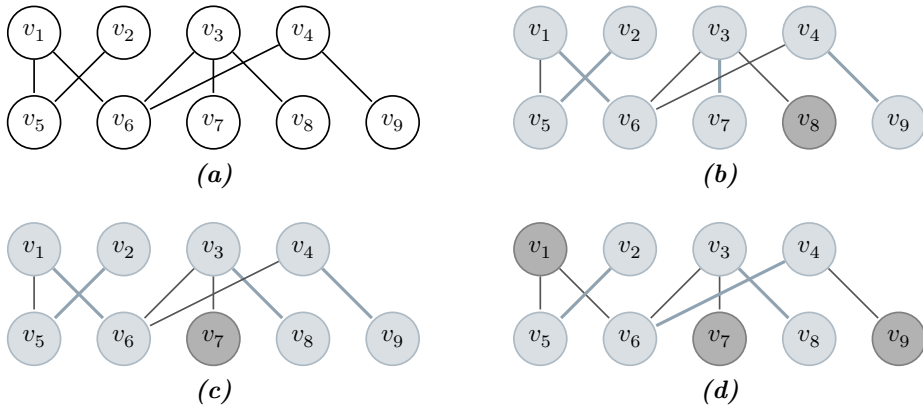


Рис. 2.74.

$\{(v_1, v_6), (v_2, v_5), (v_3, v_7), (v_4, v_9)\}$. Последнее, очевидно, является наибольшим (ребра и насыщенные вершины отмечены светло-серым цветом, оставшиеся ребра и свободная вершина v_8 — темно-серым), рисунок 2.74b.

В данном графе существует еще наибольшее паросочетание, например, $\{(v_1, v_6), (v_2, v_5), (v_3, v_8), (v_4, v_9)\}$, рисунок 2.74c.

Паросочетание $\{(v_2, v_5), (v_3, v_8), (v_4, v_6)\}$, рисунок 2.74d, является максимальным по включению, но не наибольшим.

Число паросочетания равно 4.

Формальная постановка задачи. В заданном двудольном графе найти наибольшее паросочетание (или все наибольшие паросочетания).

Для нахождения наибольшего паросочетания применим идею последовательного улучшения любого паросочетания.

Рассмотрим сначала два произвольных паросочетания X и Y , второе из которых содержит большее число ребер: $|X| < |Y|$ (на рисунке 2.75a и b ребра паросочетания X выделены жирной линией, а паросочетания Y — двойной).

Если исключить общие для обоих паросочетаний ребра (на рисунке 2.75a это ребро GM), то оставшиеся ребра разбиваются на цепочки.

Причем цепочки, имеющие два или более ребер, состоят из чередующихся ребер различных паросочетаний. Действительно, ребра одного паросочетания не могут быть смежными по определению. Так как ребер у паросочетания Y больше, то обязательно найдется цепочка, начинающаяся и заканчивающаяся ребром из паросочетания Y (на рисунке 2.75a это единственное ребро DH , на рисунке 2.75b это цепочка $FLANDN$).

Таким образом, переход от паросочетания X к Y может осуществляться изменением принадлежности ребер вдоль такой цепочки: все ребра, принадлежащие паросочетанию X , из нее надо удалить, заменив оставшимися ребрами этой цепочки.

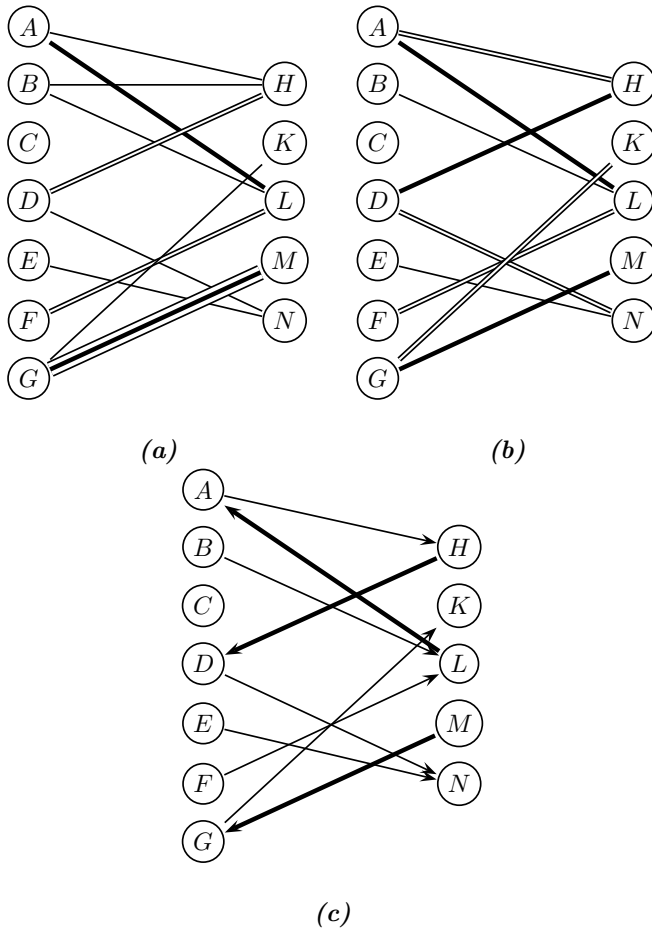


Рис. 2.75.

Чтобы алгоритмизировать процесс нахождения таких цепочек, сделаем следующее: ориентируем ребра паросочетания X справа налево, а ребра, не вошедшие в X , — слева направо. Тогда движение вдоль цепочки соответствует движению по направлениям ребер, начиная с вершины левой части двудольного графа (рисунок 2.75с).

Далее, чтобы применить для нахождения улучшающих цепочек алгоритмы обхода графа, введем две фиктивные вершины: S (начальную) и T (конечную). Соединим вершину S с вершинами левой части L , не входящими в построенное паросочетание X , выбрав направления ребер из вершины S . Аналогично все «свободные» вершины правой части соединим с вершиной T , направив ребра в нее (рисунок 2.76).

Тогда если удастся найти путь из вершины S в вершину T , то вдоль него можно улучшить паросочетание X , поменяв ориентацию вдоль пути (что и означает замену ребер старого паросочетания ребрами улучшающего

паросочетания) и исключив фиктивные ребра. На рисунке 2.76 таким путем является

$$S \rightarrow F \rightarrow L \rightarrow A \rightarrow H \rightarrow D \rightarrow N \rightarrow T.$$

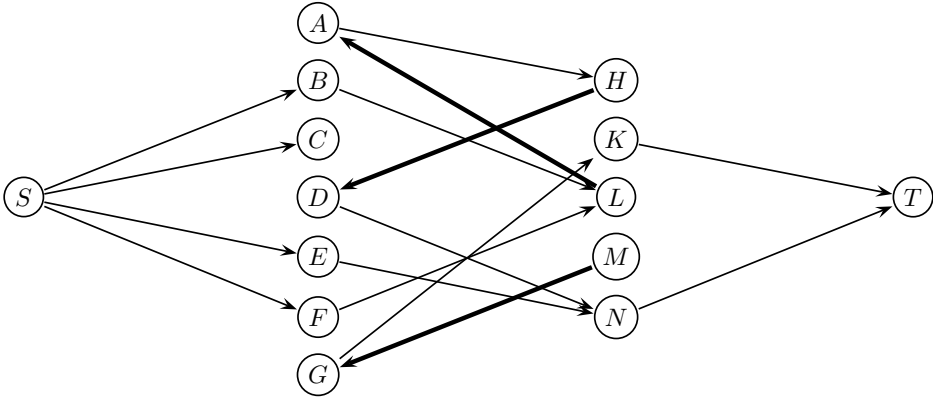


Рис. 2.76.

Проводя указанное действие, получаем следующий граф (рисунок 2.77).

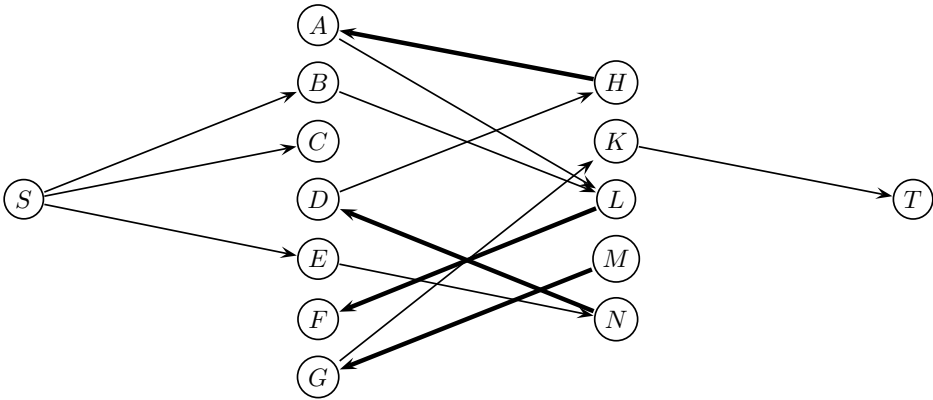


Рис. 2.77.

Если на очередном шаге нет пути из S в T , то максимальное паросочетание найдено.

Обычно существует несколько различных путей из S в T . Если стоит задача отыскания всех паросочетаний, то нужно организовать перебор вариантов, в противном случае достаточно выбора произвольного пути.

В разобранным примере (см. рисунок 2.76) имеется еще один улучшающий путь $S \rightarrow E \rightarrow N \rightarrow T$, который приводит к другому наибольшему паросочетанию.

Сравните: $X_1 = \{AH; DN; FL; GM\}$ и $X_2 = \{AL; DH; EN; GM\}$.

Построенные алгоритмом наибольшие паросочетания могут быть различными (в зависимости от процедуры выбора очередного пути), но число ребер в них всегда будет одинаковым: число паросочетания графа является инвариантом алгоритма.

Сформулируем эти рассуждения в виде алгоритма.

Алгоритм 2.16. Построение наибольшего паросочетания в двудольном графе

Исходные данные: двудольный граф $G = (V_1 \cup V_2, E)$

Результат: Множество ребер графа G , ориентированных справа налево

// Инициализация

Вводим две фиктивные вершины S и T

Соединяем S с вершинами V_1 фиктивными ребрами, ориентированными от вершины S

Соединяем T с вершинами V_2 фиктивными ребрами, ориентированными к вершине T

Ориентируем все ребра двудольного графа слева направо (от V_1 к V_2)

// Основной алгоритм

while существует путь $\{e_1, \dots, e_k\} \mid e_i \in E, 2 \leq i \leq k-1$ от S к T **do**

 Удаляем фиктивные ребра пути: e_1 и e_k

 Изменяем ориентацию ребер графа G вдоль пути: e_2, \dots, e_{k-1}

end while

Пример 2.59. С помощью алгоритма 2.16 построим наибольшее паросочетание для двудольного графа G .

Левая доля V_1 состоит из вершин $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$, правая доля V_2 — из вершин $\{u_1, u_2, u_3, u_4, u_5, u_6, u_7\}$. Ребра в графе заданы следующим списком:

$(v_1, u_5), (v_1, u_6), (v_1, u_7), (v_2, u_3), (v_3, u_2), (v_4, u_2), (v_4, u_3), (v_5, u_1), (v_5, u_3),$
 $(v_5, u_4), (v_5, u_6), (v_6, u_7), (v_7, u_2).$

Проводим инициализацию алгоритма. Получившийся ориентированный граф изображен на рисунке 2.78.

Будем перебирать возможные пути, для определенности соединяя вершины с меньшими номерами. Обозначим искомое паросочетание P . Запишем протокол работы алгоритма в таблице 2.22.

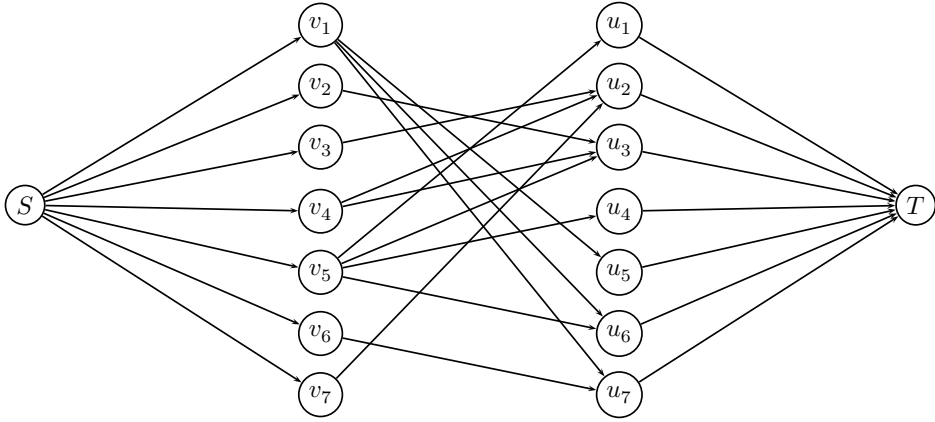


Рис. 2.78.

Таблица 2.22.

Путь	Фиктивные ребра	Добавляем в P
$S \rightarrow v_1 \rightarrow u_5 \rightarrow T$	$(S, v_1), (u_5, T)$	(v_1, u_5)
$S \rightarrow v_2 \rightarrow u_3 \rightarrow T$	$(S, v_2), (u_3, T)$	(v_2, u_3)
$S \rightarrow v_3 \rightarrow u_2 \rightarrow T$	$(S, v_3), (u_2, T)$	(v_3, u_2)
$S \rightarrow v_5 \rightarrow u_1 \rightarrow T$	$(S, v_5), (u_1, T)$	(v_5, u_1)
$S \rightarrow v_6 \rightarrow u_7 \rightarrow T$	$(S, v_6), (u_7, T)$	(v_6, u_7)

Заметим, что на каждом шаге размерность паросочетания увеличивается на единицу.

Таким образом,

$$P = \{(v_1, u_5), (v_2, u_3), (v_3, u_2), (v_5, u_1), (v_6, u_7)\}.$$

Иллюстрация работы алгоритма приведена на рисунке 2.79.

Построенное паросочетание не является единственным для данного графа. Откажемся от условия соединять вершины с минимально возможными номерами. Протокол работы алгоритма запишем в таблице 2.23.

Здесь также на каждом шаге размерность паросочетания увеличивается на единицу: если в паросочетание добавляются два ребра, то одно ребро из паросочетания удаляется.

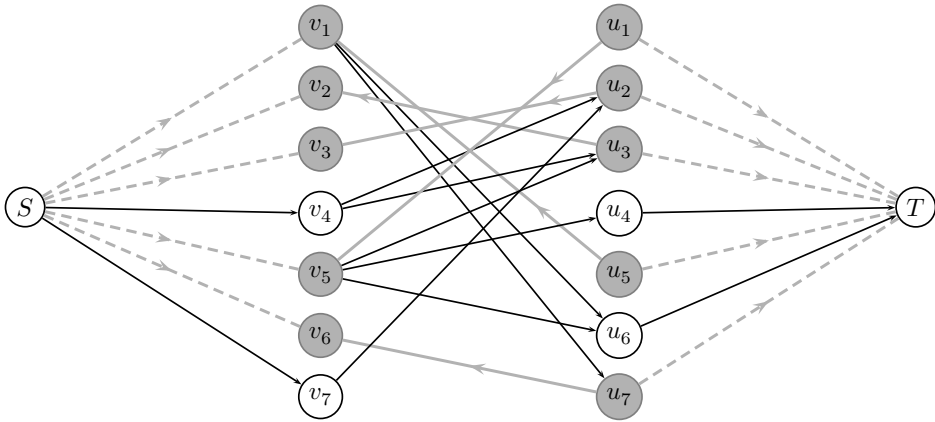


Рис. 2.79.

Таблица 2.23.

Путь	Фиктивные ребра	Добавляем в P	Удаляем из P
$S \rightarrow v_1 \rightarrow u_6 \rightarrow T$	$(S, v_1), (u_6, T)$	(v_1, u_6)	
$S \rightarrow v_5 \rightarrow u_6 \rightarrow v_1 \rightarrow u_5 \rightarrow T$	$(S, v_5), (u_5, T)$	$(v_5, u_6), (v_1, u_5)$	(v_1, u_6)
$S \rightarrow v_4 \rightarrow u_3 \rightarrow T$	$(S, v_4), (u_3, T)$	(v_4, u_3)	
$S \rightarrow v_2 \rightarrow u_3 \rightarrow v_4 \rightarrow u_2 \rightarrow T$	$(S, v_2), (u_2, T)$	$(v_2, u_3), (v_4, u_2)$	(v_4, u_3)
$S \rightarrow v_6 \rightarrow u_7 \rightarrow T$	$(S, v_6), (u_7, T)$	(v_6, u_7)	

Экспериментальная работа. Сравните работу алгоритмов поиска в ширину и глубину для поиска максимальных паросочетаний в графах-решетках размером $n \times t$ (см. пример 2.36, рисунок 2.43b).

2.2.2. Построение минимальных стягивающих деревьев

Определение 2.57. Граф $G = (V, E)$ называют *нагруженным*, или *взвешенным*, если для каждого ребра (v_i, v_j) определена его длина (или вес) w_{ij} . Матрицу $\{w_{ij}\}$ называют *матрицей весов ребер*.

Весы несуществующих ребер полагают равными ∞ или 0 в зависимости от приложений. Заметим, что матрица весов является простым обобщением матрицы смежности.

Пусть G — связный нагруженный граф. Задача построения минимального стягивающего дерева заключается в том, чтобы из множества стягивающих деревьев найти дерево, у которого сумма длин ребер минимальна. Приведем типичные случаи, когда возникает необходимость построения минимального стягивающего дерева графа.

1. Нужно соединить n городов железнодорожными линиями (автомобильными дорогами, линиями электропередачи, сетью трубопроводов и т. д.) так, чтобы суммарная длина линий или стоимость была минимальной.

2. Требуется построить схему электрической сети, в которой клеммы должны быть соединены с помощью проводов наименьшей общей длины.

Решение этой задачи «слепым» перебором вариантов потребовало бы чрезвычайно больших вычислений даже при относительно малых n . Формула Кэли (следствие 2.9) показывает, что полный граф K_n содержит n^{n-2} различных стягивающих деревьев.

Однако для ее решения имеются эффективные алгоритмы. Далее будут описаны два из них — алгоритмы Прима и Краскала, применяемые к произвольному связному графу G с n вершинами.

Историческая справка

Исторически первый алгоритм построения минимального стягивающего дерева был предложен Отакаром Боровкой (Otakar Borůvka) в 1926 г., задолго до появления первых компьютеров и современной теории графов.

Боровка применил его в качестве метода нахождения оптимальной электрической сети в Моравии. Несколько раз он был переоткрыт различными авторами, поэтому его иногда называют алгоритмом Соллина, особенно в литературе по параллельным вычислениям.

Рассматриваемые алгоритмы разработаны практически одновременно Джоозефом Краскалом (Joseph Kruskal) и Робертом Примом (Robert Prim) в 1956 г. Заметим, что еще в 1930 г. алгоритм Прима был открыт чешским математиком Войтеком Ярником, но распространения тогда не получил.

В 1958 г. Э. Дейкстра (Edsger Dijkstra) независимо переоткрыл алгоритм Прима, но название осталось прежним, так как алгоритм Дейкстры уже существовал (см. алгоритм 2.22 поиска кратчайших путей в графе).

Замечание 2.46

Рассматриваемые далее алгоритм Прима и алгоритм Краскала, как и алгоритм Хаффмана (см. замечание 1.51) являются примерами жадных алгоритмов.

Алгоритм 2.17. Прима или ближайшего соседа

// Построение дерева $T = (E_T, V_T)$ начинаем с произвольной вершины s
 $V_T \leftarrow \{s\}$

```

while существуют вершины, не принадлежащие дереву  $T$  do
    // из ребер, соединяющих вершины  $T$  с вершинами графа  $G$ , выбираем ребро
    // минимального веса
     $e \leftarrow \arg \min \{w_{i,j} \mid (v_i, v_j) \in E \mid v_i \in T, v_j \notin T\}$ 
    // включаем его в дерево вместе с его инцидентной вершиной, не входящей в
     $T$ .
     $E_T \leftarrow E_T \cup \{e\}$ ,  $V_T \leftarrow V_T \cup \{v_j\}$ 
end while

```

Проведем обоснование корректности алгоритма Прима.

Лемма 2.7. Пусть веса всех ребер связного графа различны. Разобьем множество V его вершин на два непересекающихся подмножества V_1 и V_2 . Рассмотрим подмножество ребер $\widehat{E} = E(V_1, V_2)$. Выберем в \widehat{E} ребро e_1 минимального веса. Тогда это ребро обязательно войдет в минимальное стягивающее дерево T .

Доказательство. От противного. Пусть это ребро не входит в минимальное стягивающее дерево: $e_1 \notin T$. Тогда должно найтись другое ребро e_2 из множества \widehat{E} , входящее в стягивающее дерево, иначе нарушилась бы связность. По предположению о различности весов ребер вес e_2 больше веса e_1 .

Добавим к дереву T хорду e_1 , тогда в полученном графе появится главный цикл, определяемый хордой e_1 . В этот цикл обязательно должно войти еще одно ребро множества \widehat{E} . Единственно возможное ребро — ребро e_2 (остальные ребра E_1 не входят в $T \cap e_1$).

Сделаем элементарное преобразование дерева T (см. определение 2.38) — включим хорду e_1 и выключим ветвь e_2 . Новое дерево, очевидно, будет также стягивающим, но меньшего веса. Полученное противоречие доказывает лемму. ■

Замечание 2.47

Если среди ребер есть ребра одинакового веса, то формулировка леммы 2.7 несколько изменится. Заключительная часть ее будет звучать так: «найдется минимальное стягивающее дерево, содержащее это ребро». Основная часть доказательства сохранится, только надо обратить внимание на то, что при замене ребер можно перейти к дереву равного веса.

Лемма 2.8. Рассмотрим подграф, содержащий все вершины, построенные на k -м шаге алгоритма и соединяющие их ребра. Тогда дерево, построенное на k -м шаге, является минимальным стягивающим деревом этого подграфа.

Доказательство. Лемма доказывается аналогично лемме 2.7. Обратим внимание на обоснование индукционного перехода.

Пусть мы добавили к текущему остовному дереву минимальное ребро из множества ребер, соединяющих его вершины с остальными. При этом мы расширили множество вершин графа (добавив второй конец ребра). Минимальное остовное дерево нового подграфа обязательно должно включать одно ребро, соединяющее новую вершину с остальными. Таких ребер не было в остовном дереве, построенном ранее. Следовательно, добавление нового ребра, инцидентного данной вершине, не может сказаться на выборе ребер на предыдущих этапах. ■

Справедлива следующая теорема.

Теорема 2.47. Граф T_{n-1} является стягивающим деревом минимального веса в графе G .

Доказательство. Рассмотрим случай, когда все веса ребер различны. Докажем ее по индукции.

База индукции. При инициализации алгоритма выбирается произвольная вершина s связного неориентированного графа. Далее в дерево включается минимальное ребро, соединяющее эту вершину с остальными. Обозначив $V_1 = \{s\}$, $V_2 = V \setminus \{s\}$, по лемме 2.7 можно утверждать, что выбранное ребро обязательно войдет в минимальное остовное дерево.

Индукционный переход. Пусть после k шагов алгоритма получено k ребер, принадлежащих минимальному стягивающему дереву.

Обозначив $V_1 = \{s, v_1, \dots, v_k\}$, $V_2 = V \setminus V_1$, можно применить лемму 2.7 еще раз и утверждать, что очередное выбранное ребро принадлежит стягивающему дереву.

Итак, для случая различных весов ребер теорема доказана.

В общем случае теорема следует из леммы 2.8 очевидным способом, когда k становится равным числу на 1 меньше числа вершин графа. ■

Проиллюстрируем работу алгоритма Прима на примере.

Пример 2.60. Построим стягивающее дерево минимального веса для графа на рисунке 2.80, начиная с вершины H .

Для удобства изложения условимся при наличии на некотором шаге выбора из нескольких ребер включать ребро с уже включенной в дерево вершиной в алфавитном порядке. Если при этом есть выбор, выбираем ребро в алфавитном порядке по второй концевой вершине.

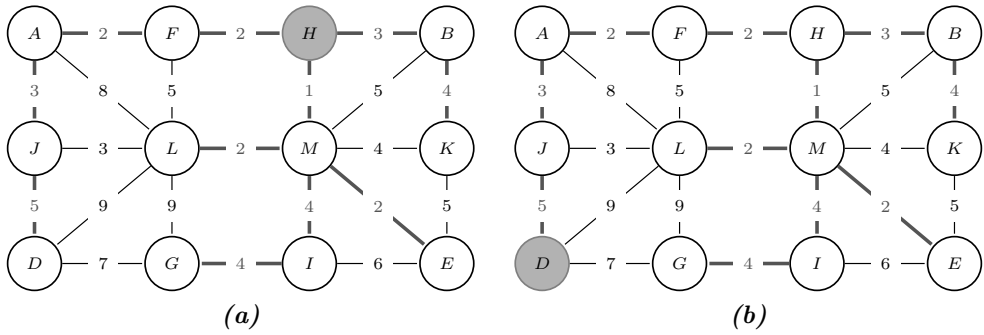


Рис. 2.81.

Последовательность включаемых ребер (в скобках указан вес ребра):

$$GH(6) \rightarrow CH(1) \rightarrow CD(2) \rightarrow BC(3) \rightarrow AB(1) \rightarrow CF(3) \rightarrow FJ(1) \rightarrow \\ \rightarrow FI(3) \rightarrow FK(4) \rightarrow EI(4) \rightarrow KL(4).$$

Таким образом, вес построенного дерева T_{11} равен 32. Полученное дерево представлено на рисунке 2.82b, его ветви имеют темно-серый цвет.

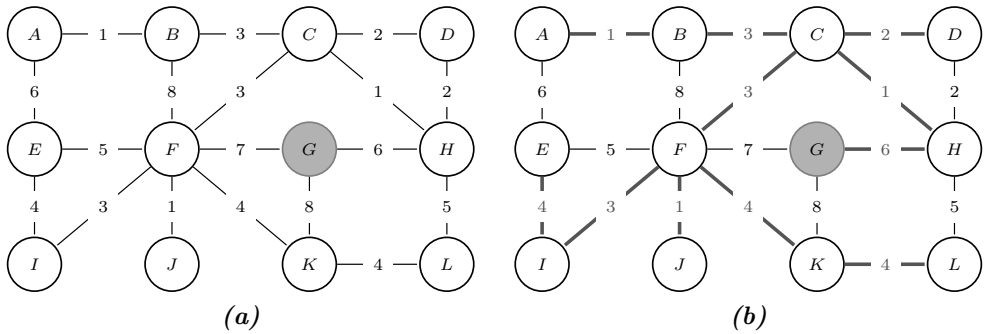


Рис. 2.82.

Замечание 2.48

Приведенные рассуждения нигде не опирались на знаки весов ребер и, следовательно, алгоритм Прима применим для графов с произвольными (положительными, отрицательными или нулевыми) весами ребер.

Отсюда вытекает, что стягивающее дерево максимального веса можно также построить алгоритмом Прима. Следует только изменить знаки весов на противоположные или всюду минимальный вес заменить максимальным.

Все сказанное будет справедливо и для рассматриваемого ниже алгоритма Краскала.

Упражнение 2.8. Используя замечание 2.48, постройте стягивающее дерево максимального веса для графов на рисунках 2.80 и 2.82a.

Сравните результаты для различных стартовых вершин алгоритма.

Алгоритм 2.18. Краскала

```

// Включаем в подграф  $G'$  ребро (с концевыми вершинами) минимального веса в
// графе
 $E_{G'} \leftarrow (u, v) = \arg \min \{w_{i,j} \mid (v_i, v_j) \in E\}$ 
 $V_{G'} \leftarrow \{v, u\}$ 
while существуют ребра из  $G \setminus G'$ , не составляющие цикла с ребрами  $G'$  do
  // добавляем в  $G'$  ребро, имеющее минимальный вес среди ребер графа  $G \setminus G'$ 
  // и не составляющее цикла с ребрами из  $G'$ 
   $(u, v) = \arg \min \{w_{i,j} \mid (v_i, v_j) \in E_G \setminus E_{G'}, (v_i, v_j) \text{ не составляет цикла с } E_{G'}\}$ 
   $E_{G'} \leftarrow E_{G'} \cup \{(u, v)\}$ 
   $V_{G'} \leftarrow V_{G'} \cup \{v, u\}$ 
end while

```

Относительно алгоритма Краскала справедлива теорема, аналогичная теореме 2.47.

Теорема 2.48. Подграф G_{n-1} является минимальным стягивающим деревом графа G .

Доказательство. От противного. Пусть существует дерево \widehat{T} с весом, меньшим чем вес дерева T_K , построенного по алгоритму Краскала. Сравним эти деревья по совпадению ребер в том порядке, как ребра вставлялись в дерево T_K .

Найдем первое из ребер e дерева \widehat{T} , не вошедших в T_K . Добавим это ребро в дерево T_K . В полученном графе появится цикл, содержащий ребро e . В этом цикле должно найтись ребро w , вес которого больше либо равен весу e .

Действительно, ребра меньшего веса вместе с e не могут образовать цикла, так как входят в дерево T_K . Заменяя ребро w на e , получим новое остовное дерево с весом, либо меньшим веса \widehat{T} (что сразу приводит к противоречию), либо равным весу \widehat{T} . В последнем случае после замены ребер можно продолжить процесс сравнения и использовать те же рассуждения.

Так как по предположению вес \widehat{T} меньше веса T_K , то рано или поздно вес нового остовного дерева должен стать меньше веса \widehat{T} , что и приводит к противоречию. ■

 **Замечание 2.49**

Алгоритм Прима можно назвать «растущим минимальным деревом»: на каждом шаге происходит расширение уже построенного минимального остовного дерева за счет соседних вершин.

В отличие от него, алгоритм Краскала можно охарактеризовать словами «растущий лес»: на каждом шаге мы пытаемся достроить одну из его компонент до

некоторого минимального стягивающего дерева. Периодически происходит слияние компонент связности леса.

Пример 2.62. Рассмотрим работу алгоритма 2.18 на примере графа на рисунке 2.83а.

Для определенности, при возможности выбора ребер одинакового веса, будем включать ребра в лексикографическом порядке вершин. Например, ребро (A, D) раньше ребра (C, F) , ребро (B, E) раньше (B, F) .

Применив алгоритм 2.18, получим одну из возможных последовательностей включаемых ребер (2.27). Ребро (D, H) (отмечено в (2.27) серым цветом) на момент рассмотрения образует цикл с уже включенными ребрами.

$$\begin{aligned} AB(1) \rightarrow CF(1) \rightarrow EH(1) \rightarrow HK(1) \rightarrow AD(2) \rightarrow DE(2) \rightarrow \\ \rightarrow DG(2) \rightarrow DH(2) \rightarrow FK(2). \end{aligned} \quad (2.27)$$

Таким образом, вес построенного дерева T_8 равен 12. Полученное дерево выглядит следующим образом (рисунок 2.83б, ветви графа темно-серого цвета, хорды — пунктирные линии).

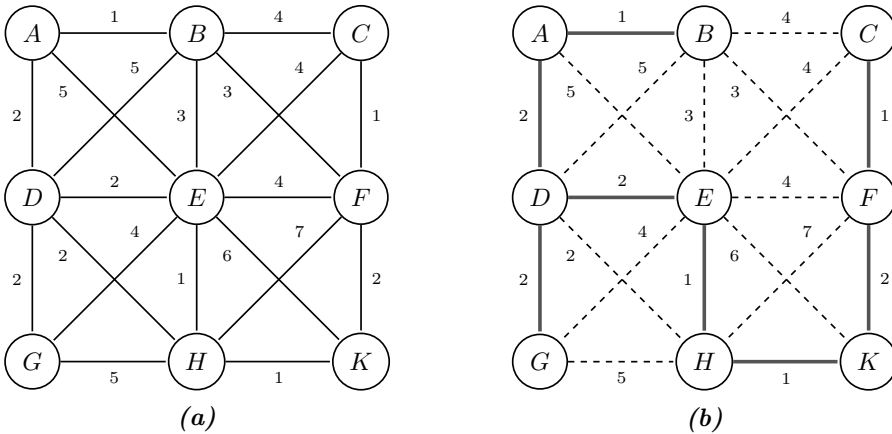


Рис. 2.83.

Замечание 2.50

Построенный алгоритм 2.18 включает процедуру определения наличия цикла в графе, что само по себе трудная задача. Поэтому алгоритм Краскала можно переписать в другой форме, используя идею раскраски вершин.

Алгоритм 2.19. Модифицированный алгоритм Краскала

// Упорядочиваем все ребра графа $G = (V, E)$ по возрастанию (неубыванию) весов

```

L = {e1, ..., ek | ei ∈ E}
T ← ∅

```

```

// Раскрашиваем вершины графа в разные цвета

```

```

clr ← 1

```

```

for each u ∈ V do

```

```

    color(u) ← clr

```

```

    clr ← clr + 1

```

```

end for

```

```

// Основной алгоритм

```

```

for each (v, u) ∈ L do

```

```

    if color(u) ≠ color(v) then

```

```

        T ← G' ∪ {(v, u)}

```

```

        cmax = max {color(u), color(v)}, cmin = min {color(u), color(v)}

```

```

        for each vi ∈ V | color(vi) = cmax do

```

```

            color(vi) = cmin

```

```

        end for

```

```

    end if

```

```

end for

```

Пример 2.63. Рассмотрим протокол работы модифицированного алгоритма Краскала для графа из примера 2.62 рисунок 2.83а.

На рисунке 2.84а граф представлен уже после этапа инициализации алгоритма, все вершины покрашены в разные цвета в соответствии с алфавитным порядком вершин (2.28).

$$A(1), B(2), C(3), D(4), E(5), F(6), G(7), H(8), K(9). \quad (2.28)$$

Замечание 2.51

При соединении двух компонент связности будем перекрашивать вершины в цвет с меньшим номером.

Последовательность рассматриваемых алгоритмом ребер, упорядоченных по неубыванию весов:

$$\begin{aligned}
 AB(1) &\rightarrow CF(1) \rightarrow EH(1) \rightarrow HK(1) \rightarrow AD(2) \rightarrow \\
 &\rightarrow DE(2) \rightarrow DG(2) \rightarrow DH(2) \rightarrow FK(2).
 \end{aligned} \quad (2.29)$$

Как и ранее, ребра одного веса в (2.29) упорядочены в лексикографическом порядке (можно было взять любой другой порядок). Ребра, вершины которых на момент выбора имеют одинаковый цвет, выделены в (2.29) светло-серым.

Рассмотрим последовательно все шаги алгоритма.

- 1 $T \leftarrow \emptyset$, рисунок 2.84a.
- 2 $T \leftarrow T \cup \{(A, B)\}$, рисунок 2.84b.
- 3 $T \leftarrow T \cup \{(C, F)\}$, рисунок 2.84c.

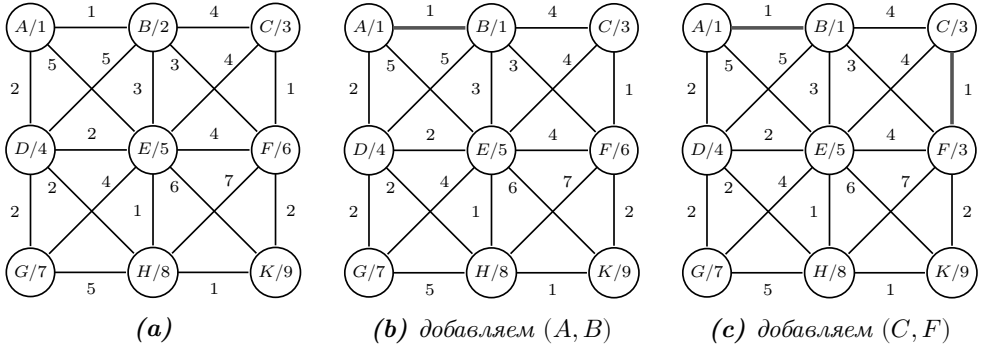


Рис. 2.84.

- 4 $T \leftarrow T \cup \{(E, H)\}$, рисунок 2.85a.
- 5 $T \leftarrow T \cup \{(H, K)\}$, рисунок 2.85b.
- 6 $T \leftarrow T \cup \{(A, D)\}$, рисунок 2.85c.

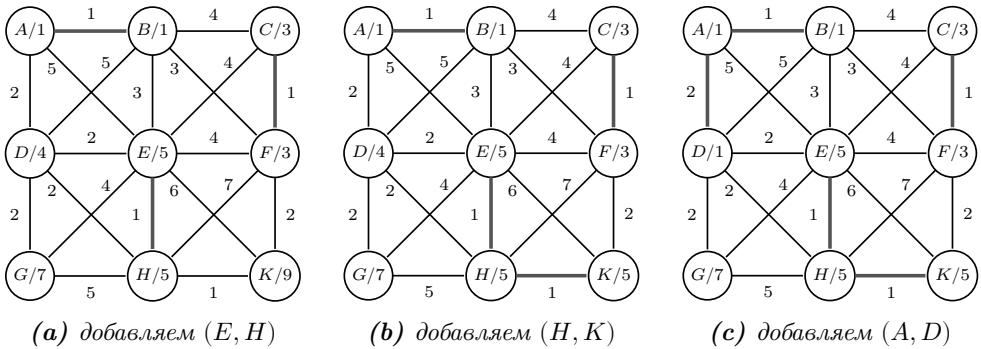


Рис. 2.85.

- 7 $T \leftarrow T \cup \{(D, E)\}$, рисунок 2.86a.
- 8 $T \leftarrow T \cup \{(D, H)\}$, рисунок 2.86b.
- 9 $T \leftarrow T \cup \{(F, K)\}$, рисунок 2.86c.

На последнем рисунке 2.86c хорды графа обозначены штриховыми линиями. Таким образом, вес построенного дерева равен 12.

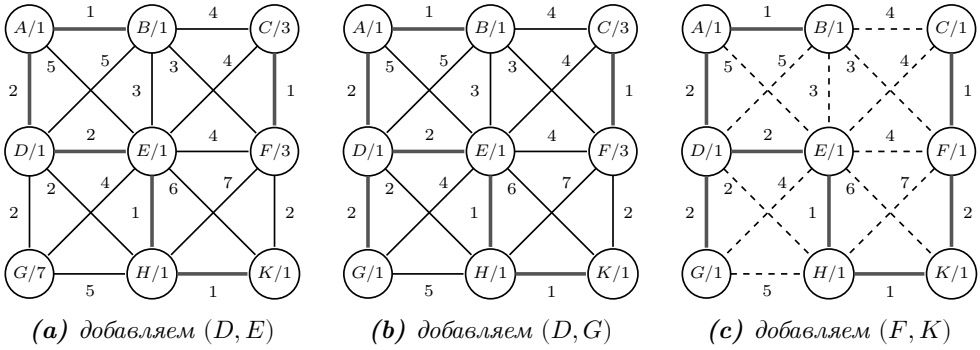


Рис. 2.86.

Замечание 2.52

В примере 2.63 видно, что дерево построено, когда все вершины графа раскрашены в один цвет. Этот признак можно использовать для проверки окончания цикла в модифицированном алгоритме Краскала (например, при числе вершин графа n цикл можно закончить, когда число выбранных ребер равно $n - 1$).

Пример 2.64. Рассмотрим еще один пример работы модифицированного алгоритма Краскала для графа на рисунке 2.87а.

Здесь граф представлен уже после этапа инициализации алгоритма, все вершины покрашены в разные цвета в соответствии с алфавитным порядком вершин (2.30).

$$A(1), B(2), C(3), D(4), E(5), F(6), G(7). \tag{2.30}$$

Перекрашивать вершины будем согласно замечанию 2.51.

Последовательность рассматриваемых алгоритмом ребер, упорядоченных по неубыванию весов:

$$AC(3) \rightarrow DG(3) \rightarrow AB(4) \rightarrow EG(4) \rightarrow GF(4) \rightarrow AD(5). \tag{2.31}$$

Далее последовательно показаны все шаги алгоритма согласно (2.31).

- 1 $T \leftarrow \emptyset$, рисунок 2.87а.
- 2 $T \leftarrow T \cup \{(A, C)\}$, рисунок 2.87b.
- 3 $T \leftarrow T \cup \{(D, G)\}$, рисунок 2.87c.
- 4 $T \leftarrow T \cup \{(A, B)\}$, рисунок 2.88а.
- 5 $T \leftarrow T \cup \{(E, G)\}$, рисунок 2.88b.

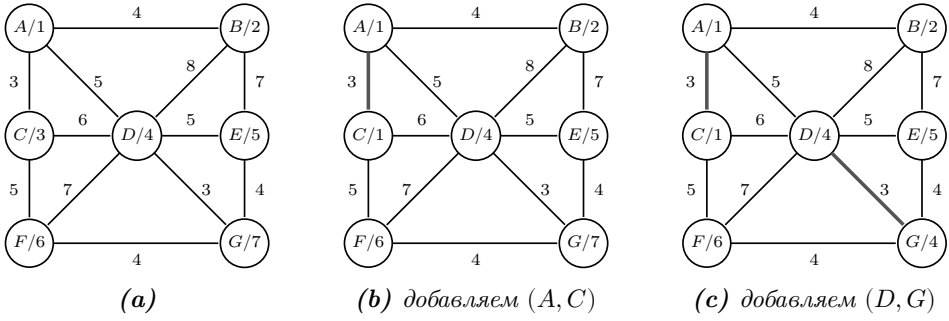


Рис. 2.87.

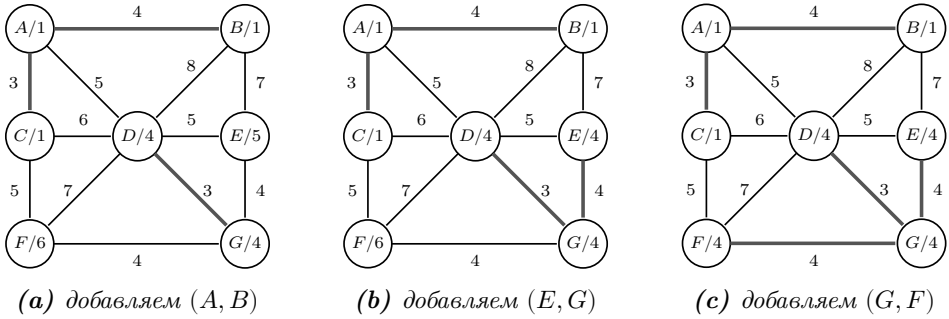


Рис. 2.88.

- 6 $T \leftarrow T \cup \{(G, F)\}$, рисунок 2.88с.
- 7 $T \leftarrow T \cup \{(A, D)\}$, рисунок 2.89.

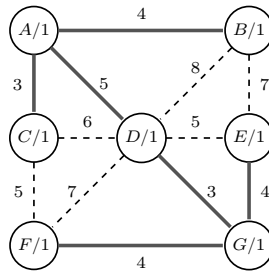


Рис. 2.89.

На рисунке 2.89 показан последний шаг алгоритма после добавления ребра (A, D). Хорды графа обозначены штриховыми линиями.

Вес построенного дерева равен 23.

Задача 2.3. Докажите корректность модифицированного алгоритма Краскала.

2.2.3. Задача нахождения кратчайших путей в графе

Рассмотрим ориентированный или неориентированный взвешенный граф. Сформулируем три различные постановки задачи на нахождение кратчайших путей.

1. Заданы две вершины графа: S и F . Найти длину кратчайшего пути, соединяющего вершину S с вершиной F .

2. Задана вершина графа S . Найти длины кратчайших путей, соединяющих вершину S с остальными вершинами графа.

3. Найти длины кратчайших путей, соединяющих все пары вершин графа.

Попытаемся оценить сложность каждой из задач и упорядочить их для последовательного решения.

Первая мысль — оставить порядок решения таким, как он здесь представлен. Действительно, решив первую задачу, можно перебрать все возможные вершины F графа, сохранив начальную S , и тем самым решить вторую задачу (сделать цикл по вершинам графа, используя в теле цикла процедуру решения первой задачи). Затем можно «освободить» первую вершину и с помощью еще одного цикла решить третью задачу.

Тем самым определена трудоемкость указанных алгоритмов: если трудоемкость алгоритма решения первой задачи обозначить $T(n)$ (n — число вершин графа), то трудоемкость второго будет $nT(n)$, третьего — $n^2T(n)$.

Такой путь возможен, но не рационален. Почему?

Для ответа на вопрос представим, как искать кратчайший путь между двумя фиксированными вершинами S и F : для того чтобы убедиться, что он кратчайший, надо сравнить его длину с длинами других возможных путей. Но для этого надо знать кратчайшие расстояния от вершины S до остальных вершин графа! Получается, что вторая задача по отношению к первой является базовой и решение надо начинать с нее.

Действительно, следующие два алгоритма, которые будут рассмотрены (алгоритм Форда — Беллмана и алгоритм Дейкстры), решают именно вторую задачу. В то же время очевидно, что при решении второй задачи одновременно решается и первая, так что выделять ее особо не требуется.

Из изложенного вытекает и основная идея алгоритма, находящего кратчайшие пути: надо сравнивать уже найденные варианты «кратчайших путей» с теми, которые используют проход через другие вершины.

Это можно записать более формально, если ввести для каждой вершины A ее пометку — вещественное число $d(A)$, которое по окончании работы алгоритма означает кратчайшее расстояние, а на промежуточных шагах

имеет иной смысл (который будет ясен из дальнейшего анализа алгоритмов, так как для каждого алгоритма его смысл на промежуточных шагах свой).

Тогда новое значение пометки можно найти как

$$d(A) := \min\{d(A), d(B) + w(B, A)\}, \quad (2.32)$$

где минимум берется по всем (промежуточным) вершинам графа B , а $w(B, A)$ — это вес ребра (в алгоритме Форда — Беллмана) или вес пути (в алгоритме Флойда), соединяющего вершины B и A .

Формулу пересчета пометок (2.32) иллюстрирует рисунок 2.90.

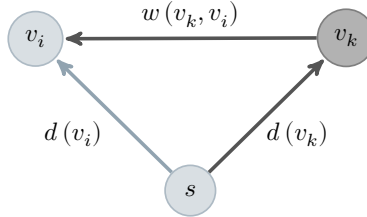


Рис. 2.90.

Для удобства пронумеруем вершины графа, тогда псевдокод описанной операции можно записать так:

```

for  $k \leftarrow 1, n$  do
     $d(i) \leftarrow \min(d(i), d(k) + w(k, i))$ 
end for

```

Назовем внутреннюю часть этого цикла модификацией пометки. Заметим, что эта формула «пересчета расстояний через промежуточную вершину» (правило модификации пометок) будет лежать в основе алгоритмов Форда — Беллмана, Дейкстры и Флойда.

Замечание 2.53

Далее при описании псевдокодов алгоритмов Форда — Беллмана, Дейкстры и Флойда будем считать, что вершины графа $G = (V, E)$ пронумерованы от 1 до $|V|$.

2.2.4. Алгоритм Форда — Беллмана

Постановка задачи. Имеется ориентированный или неориентированный взвешенный граф, веса которого задаются произвольными вещественными числами. Требуется найти длины всех кратчайших путей от вершины S (источника) до остальных вершин.

Первоначально проанализируем корректность постановки задачи. Ответим на два вопроса.

1. Что делать в случае, когда пути вообще нет, в частности если граф оказывается несвязным?

2. Не может ли оказаться, что при наличии отрицательных ребер кратчайших путей нет вовсе?

Ответ на первый вопрос прост. Введем понятие бесконечно большой длины и будем присваивать «бесконечность» пометкам вершин, обозначающим длины кратчайших путей от источника. Ответ на второй вопрос более сложен.

На рисунке 2.91 показан пример графа, в котором не существует кратчайшего пути из вершины S в вершину F . Это связано с наличием цикла отрицательной длины $(v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_1)$ — «бегая» по нему, можно «накрутить» сколь угодно малую (отрицательную, но большую по абсолютному значению) длину, т. е. наименьшего пути между S и F не существует в том смысле, что для любого пути можно найти путь меньшей длины.

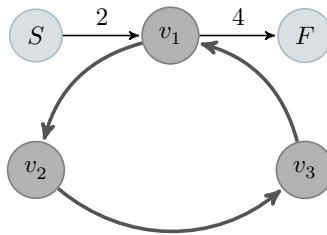


Рис. 2.91.

Нетрудно понять, что отсутствие цикла отрицательной длины и является необходимым и достаточным условием для корректности поставленной задачи. Докажите самостоятельно следующую теорему.

Теорема 2.49. Во взвешенном графе с вещественными весами кратчайшие пути между любыми двумя вершинами существуют тогда и только тогда, когда в графе нет циклов отрицательной длины.

Заметим, что это условие несколько избыточно для исходной задачи, в которой начальная вершина фиксирована. Так появляется интересное задание для исследования.

Задание 2.1. Может ли при наличии цикла отрицательной длины существовать решение задачи о кратчайших путях от фиксированной вершины S до остальных. Найдите поясняющий пример. Сформулируйте необходимые и достаточные условия для этого.

Основная идея алгоритма. Сопоставим вершинам пометки, смысл которых — кратчайшие пути, состоящие из одного ребра, т. е. из матрицы весов возьмем строчку, соответствующую начальной вершине (веса ребер, выходящих из начальной вершины).

Повторим для каждой вершины графа цикл (2.32), модифицирующий пометки одной вершины через все возможные промежуточные вершины.

Нетрудно понять, что при этом к путям из одного ребра добавятся пути из двух ребер и смысл пометки станет иным: «кратчайшее расстояние от источника до всех остальных вершин на множестве путей, состоящих либо из одного ребра, либо из двух». Каждое следующее повторение будет расширять множество путей, добавляя пути, состоящие из большего количества ребер. Но этот процесс не бесконечен.

Простой путь в графе из n вершин не может иметь более $(n - 1)$ ребра (заметим, что здесь как раз и играет роль отсутствие циклов отрицательной длины — ведь рассматривая пути с повторяющимися вершинами, т. е. с циклами, в этом случае можно было бы уменьшать длину пути и далее). Поэтому, повторив цикл $(n-2)$ раза, наверняка получим искомым результат.

После сделанных замечаний нетрудно сформулировать алгоритм, который называется алгоритмом Форда — Беллмана.

Алгоритм 2.20. Форда — Беллмана

// Инициализация

for $i \leftarrow 1, n$ **do**

$d(i) \leftarrow w(s, i)$

// s — вершина-источник

end for

// Здесь в качестве начальных значений пометок берутся веса ребер, соединяющих вершину-источник s с остальными (при отсутствии таких ребер пометка берется равной ∞)

// Основной алгоритм

for $t \leftarrow 1, n - 2$ **do**

for $i \leftarrow 1, n$ **do**

for $k \leftarrow 1, n$ **do**

$d(i) \leftarrow \min(d(i), d(k) + w(k, i))$ *// Правило модификации пометок (2.32)*

end for

end for

end for

// Переменная t — счетчик итераций, i и k — номера вершин

Замечание 2.54

В случае отсутствия циклов отрицательной длины приведенный алгоритм можно ускорить за счет следующих простых соображений.

1. Если на очередной итерации не произошло ни одного изменения пометок, то алгоритм завершает работу. Это не улучшит асимптотику, т. е. на некоторых графах по-прежнему будут нужны все итерации, но ускорит поведение алгоритма «в среднем».
2. На очередной итерации рассматривать не все ребра, а только выходящие из вершин, для которых на предыдущей итерации были улучшены пометки. На первой итерации — только ребра, выходящие из вершины-источника.
3. Предложенный алгоритм позволяет определить, существует ли в графе G цикл отрицательной длины, достижимый из вершины источника s . Для этого нужно провести дополнительную внешнюю итерацию цикла для $m = n - 1$. Если при этом пометка какой-либо вершины уменьшилась, то в графе есть отрицательный цикл, достижимый из s .

Доказательство корректности алгоритма Форда — Беллмана. Доказательство корректности циклических алгоритмов использует общий прием, который называется выделением инварианта цикла.

Инвариант цикла — это утверждение, зависящее от параметра цикла (в данном случае m), которое остается верным (при любом допустимом значении параметра m) после исполнения тела цикла при условии, что оно было верно до этого.

Для данного циклического алгоритма таким инвариантом является содержательный смысл пометки $d(i)$. Сформулируем этот результат в виде леммы.

Лемма 2.9. После выполнения m -го шага алгоритма (точнее, внешнего цикла алгоритма) пометка $d(i)$ обозначает кратчайшее расстояние между источником и вершиной i на множестве путей, состоящих не более чем из m ребер.

Доказательство. Докажем утверждение по индукции.

База индукции. Первым шагом алгоритма является инициализация, которая и обеспечивает верность утверждения для $m = 1$. База доказана.

Индукционный переход. Пусть утверждение верно для $m = t$ перед выполнением тела цикла, докажем, что оно будет верно и для $m = t + 1$ после выполнения тела цикла.

То есть надо доказать, что если до выполнения тела цикла пометка $d(i)$ показывала кратчайшее расстояние от источника до вершины i на множестве путей, состоящих не более чем из t ребер, то после его выполнения она будет показывать кратчайшее расстояние на множестве путей, состоящих не более чем из $(t + 1)$ ребра.

Но любой путь до вершины i из не более чем $(t+1)$ ребра — это путь из не более чем t ребер (для таких путей кратчайшее расстояние уже известно по индукционному предположению) или путь ровно из $(t+1)$ ребра. В последнем случае этот путь можно разбить на два: путь до предпоследней вершины k из t ребер (наименьшее значение которого известно по индукционному предположению) и ребро из вершины k в вершину i . Остается из этих двух вариантов выбрать наилучший:

$$d(i) := \min\{d(i), d(k) + w(k, i)\}.$$

Но это и есть то самое правило модификации, которое лежит в основе алгоритма! ■

Перейдем к доказательству корректности алгоритма. Посмотрим теперь на наш инвариант:

« $d(i)$ — кратчайшее расстояние на множестве путей не более чем из t ($t \leq n$) ребер».

С другой стороны, число t меняется от 1 до $(n-1)$, а утверждение остается верным. Значит, после окончания работы алгоритма получим утверждение:

« $d(i)$ — кратчайшее расстояние на множестве путей не более чем из $(n-1)$ ребра».

Но так как циклов отрицательной длины в графе нет, то последнее утверждение равносильно утверждению:

« $d(i)$ — кратчайшее расстояние на множестве всех путей».

Корректность алгоритма установлена.

Замечание 2.55

Полезно обратить внимание на два содержательных шага в доказательстве:

1) использование метода математической индукции; причем база индукции обеспечивалась соответствующей инициализацией, а индукционный переход — соответствующей модификацией;

2) использование инварианта цикла — утверждения, которое, с одной стороны, отражает содержание введенной переменной, с другой — позволяет формально анализировать корректность циклического алгоритма.

Это общие приемы, которые будут повторяться при выводе и обосновании других алгоритмов.

В доказательстве предполагалось, что на очередном шаге внешнего цикла используются пометки предыдущего шага, на самом деле пометки меняются не только при изменении параметра внешнего цикла t , но и при изменении внутреннего i (при переходе от одной вершины к другой). Разумеется, от этого «скорость» нахождения длин кратчайших путей только увеличивается.

2.2.5. Алгоритм Форда — Беллмана для нахождения дерева кратчайших путей

Рассмотренный алгоритм Форда — Беллмана позволяет находить длины кратчайших путей от источника до остальных вершин, но не сами пути. В то же время нетрудно видеть, что при поиске длин кратчайших путей имеется вся необходимая информация для восстановления путей.

Небольшая доработка алгоритма Форда — Беллмана, предусматривающая введение вторых пометок вершин графа (для хранения тех вершин, через которые осуществлялась модификация длины пути), решает эту задачу.

Заметим, что объединение всех кратчайших путей от источника до остальных вершин обязательно дает дерево, ориентированное естественным образом (от корня к листьям), которое называют *деревом кратчайших путей*. Обоснование этому дает следующая лемма.

Лемма 2.10. Если построен кратчайший путь от вершины S до вершины F , а M — произвольная промежуточная вершина этого пути, то кратчайший путь от S до M будет частью пути от S до F .

Доказательство. Докажем от противного. Пусть существует путь от S до M , не полностью совпадающий с начальной частью пути от S до F (рисунок 2.92). Тогда построим новый путь от S до F , состоящий из нового пути от S до M и старого — от M до F . По предположению он будет короче, что противоречит тому, что старый путь от S до F является кратчайшим. Противоречие доказывает теорему. ■

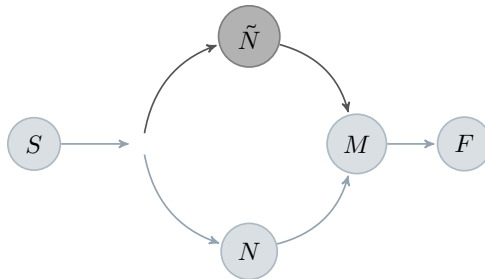


Рис. 2.92.

Задание 2.2. Верно ли, что для связного неориентированного взвешенного графа дерево кратчайших путей будет:

- 1) остовным деревом графа;
- 2) минимальным остовным деревом графа.

Ответ обоснуйте.

Приведем теперь сам алгоритм.

Алгоритм 2.21. Форда — Беллмана: нахождение дерева кратчайших путей

```

// Инициализация
for  $i \leftarrow 1, n$  do
     $d(i) \leftarrow w(s, i)$  //  $s$  — вершина-источник
     $G(i) \leftarrow s$ 
end for

// Вычисление пометок
for  $m \leftarrow 1, n - 2$  do
    for  $i \leftarrow 1, n$  do
        for  $k \leftarrow 1, n$  do
            if  $d(k) + w(k, i) < d(i)$  then
                 $d(i) \leftarrow d(k) + w(k, i)$  // Правило модификации пометок (2.32)
                 $G(i) \leftarrow k$ 
            end if
        end for
    end for
end for

// Построение дерева кратчайших путей  $T$ 
 $T \leftarrow \emptyset$ 
for  $i \leftarrow 1, n$  do
     $T \leftarrow T \cup (G(i), i)$  //  $(G(i), i)$  — очередное ребро дерева кратчайших путей
end for

```

Замечание 2.56

В алгоритме 2.21 в дерево добавляется петля (s, s) , где s — источник. Этого можно легко избежать, организовав последний цикл по всем вершинам, кроме вершины s .

Рассмотрим работу алгоритма 2.21 на примере.

Пример 2.65. Граф представлен на рисунке 2.93а, матрица его весов¹ — на рисунке 2.93б. Источником является вершина A .

¹ Символом ∞ в таблице обозначается отсутствие соответствующей дуги.

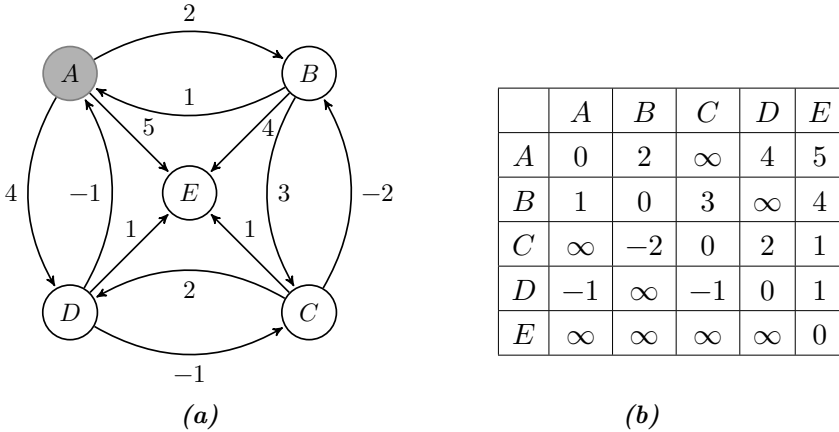


Рис. 2.93.

Методические замечания. При «прогонке» алгоритма полезно не смотреть на рисунок, а использовать более формальное представление графа, например матрицу весов. Результат работы алгоритма, наоборот, рекомендуется проиллюстрировать графически.

Протокол оформим в виде таблицы, где каждая строка — значения пометок на очередном шаге внешнего цикла. Для экономии места будем следить за обеими пометками одновременно.

Выполним инициализацию. В качестве начальной вершины S (источника) возьмем вершину A . Тогда первой строкой протокола будет первая строка матрицы весов, в которую добавлены пометки «предшествующих» вершин. В начале работы алгоритма это источник, в данном случае — вершина A .

Пометку вершины A можно не модифицировать, так как при отсутствии циклов отрицательной длины пометка источника измениться не может.

Номер шага m	B	C	D	E
1	2/ A	∞	4/ A	5/ A

Далее поочередно для всех вершин, начиная с вершины B , проверим возможность улучшить ее пометку через все соседние вершины.

К вершине B можно перейти из вершин A и C (см. столбец B матрицы весов, рисунок 2.93b). Для вершины C суммарный вес пути будет

$$d(C) + w(C, B) = \infty - 2 = \infty,$$

что не улучшает старую пометку, которую, таким образом, следует сохранить.

К вершине C можно перейти из вершин B и D (см. столбец C матрицы весов, рисунок 2.93*b*), соответствующие веса путей будут:

$$d(B) + w(B, C) = 2 + 3 = 5; \quad d(D) + w(D, C) = 4 - 1 = 3.$$

Вторая сумма лучше и первой, и старой пометки (которая равна ∞), поэтому меняем эту пометку и сохраним имя вершины (пометка $G(C)$), через которую пришли в C . Далее аналогично. Заметим, что пометку в вершине E улучшим через C :

$$d(E) = d(C) + w(C, E) = 3 + 1 = 4,$$

которая сама была улучшена на предыдущем шаге. Об этом «ускорении» отмечалось в замечании 2.55.

Номер шага m	B	C	D	E
1	2/ A	∞	4/ A	5/ A
2	2/ A	3/ D	4/ A	4/ C

На следующем шаге внешнего цикла произойдет модификация пометки $d(B)$ через вершину C :

$$d(B) = d(C) + w(C, B) = 3 - 2 = 1.$$

Очередной шаг не принесет изменений, что неудивительно, так как самые короткие пути необязательно должны состоять из наибольшего числа ребер. Итоговое состояние протокола:

Номер шага m	B	C	D	E
1	2/ A	∞	4/ A	5/ A
2	2/ A	3/ D	4/ A	4/ C
3	1/ C	3/ D	4/ A	4/ C
4	1/ C	3/ D	4/ A	4/ C

Для построения дерева кратчайших путей достаточно последней строки протокола:

Номер шага m	B	C	D	E
4	1/ C	3/ D	4/ A	4/ C

Собрав пары «пометка вершины — вершина», получим дерево кратчайших путей:

$$(C, B), (D, C), (A, D), (C, E).$$

Около каждой вершины можно указать кратчайшее расстояние до нее:

$$A - 0, B - 1, C - 3, D - 4, E - 4.$$

На рисунке 2.94 показано дерево кратчайших путей с кратчайшими расстояниями до вершин.

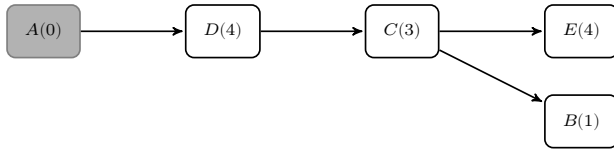
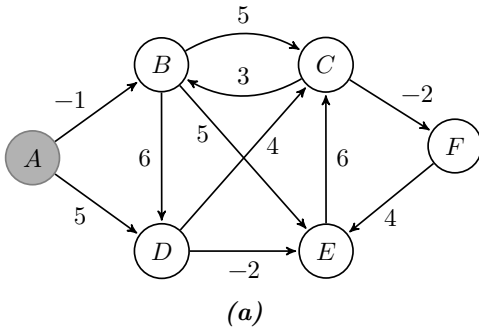


Рис. 2.94.

Задача 2.4. Примените алгоритм Форда — Беллмана для построения дерева кратчайших путей для того же графа, но других начальных вершин: а) B , б) C , в) D .

Рассмотрим еще один пример работы алгоритма Форда — Беллмана.

Пример 2.66. Граф представлен рисунке 2.95а, матрица его весов — на рисунке 2.95б. Источником является вершина A .



	A	B	C	D	E	F
A	0	-1	∞	5	∞	∞
B	∞	0	5	6	5	∞
C	∞	3	0	∞	∞	-2
D	∞	∞	4	0	-2	∞
E	∞	∞	6	∞	0	∞
F	∞	∞	∞	∞	4	0

Рис. 2.95.

Выполним инициализацию аналогично примеру 2.65. Имеем:

Номер шага m	B	C	D	E	F
1	$-1/A$	∞	$5/A$	∞	∞

Согласно замечанию 2.54 на каждой итерации рассматриваем только ребра, выходящие из вершин, для которых на предыдущей итерации были улучшены пометки.

Для второй итерации это ребра из вершин B и D , т. е.

$$(B, C), (B, D), (B, E), (D, C), (D, E).$$

Улучшаем пометки:

$$d(C) = d(B) + w(B, C) = 4, \quad d(E) = d(D) + w(D, E) = 3.$$

Запишем протокол после второго шага алгоритма:

Номер шага m	B	C	D	E	F
1	$-1/A$	∞	$5/A$	∞	∞
2	$-1/A$	$4/B$	$5/A$	$3/D$	∞

Рассматриваем ребра из вершин C и E : $(C, B), (C, F), (E, C)$. Тогда

$$d(F) = d(C) + w(C, F) = 2$$

и протокол после третьего шага алгоритма имеет вид:

Номер шага m	B	C	D	E	F
1	$-1/A$	∞	$5/A$	∞	∞
2	$-1/A$	$4/B$	$5/A$	$3/D$	∞
3	$-1/A$	$4/B$	$5/A$	$3/D$	$2/C$

Ребро (F, E) из вершины F улучшений пометок не дает. Согласно замечанию 2.54 алгоритм заканчивает работу.

На рисунке 2.96 показано соответствующее дерево кратчайших путей с кратчайшими расстояниями до вершин.

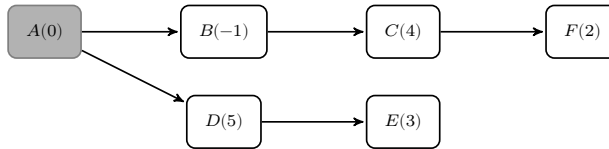


Рис. 2.96.

2.2.6. Алгоритм Дейкстры

Постановка задачи. В отличие от предыдущей, в данной задаче веса ребер неотрицательны. Тем самым циклы отрицательной длины автоматически исключаются и задача всегда будет иметь решение. Кроме того, это

условие позволяет на порядок уменьшить трудоемкость алгоритма (об этом далее).

Основная идея алгоритма. На каждом шаге отыскивается вершина, кратчайшее расстояние до которой уже найдено. Например, на первом шаге это будет вершина — конец наименьшего ребра, инцидентного источнику. Пометки остальных вершин модифицируются через эту вершину, после чего ее можно не рассматривать.

Алгоритм 2.22. Дейкстры

```

// Инициализация (как в алгоритме 2.21)
for  $i \leftarrow 1, n$  do
     $d(i) \leftarrow w(s, i)$  //  $s$  — вершина-источник
     $G(i) \leftarrow s$ 
end for

//  $W$  — множество вершин, кратчайшее расстояние до которых уже найдено
 $W \leftarrow \{s\}$ 

// Основной алгоритм
while  $V \setminus W \neq \emptyset$  do // есть необработанные вершины
     $t \leftarrow u \mid d(u) = \min \{d(v), v \in V \setminus W\}$ 
    //  $t$  — необработанная вершина с минимальной пометкой
     $W \leftarrow W \cup \{t\}$  // переносим найденную вершину  $t$  в множество обработанных

    // модифицируем пометки необработанных вершин, смежных с  $t$ , через вершину  $t$ 
    for  $u \in V \setminus W \mid (t, u) \in E$  do
        if  $d(t) + w(t, u) < d(u)$  then
             $d(u) \leftarrow d(t) + w(t, u)$  // Правило модификации пометок (2.32)
             $G(u) \leftarrow t$ 
        end if
    end for
end while

```



Историческая справка

Эдсгер Дейкстра — нидерландский ученый, один из разработчиков (совместно с Тони Хоаром и Никлаусом Виртом) концепции структурного программирования. Лауреат премии Тьюринга (1972 г.).

В 1956 г. принял участие в разработке ЭВМ X1. Именно для оптимизации разводки плат для X1 был придуман алгоритм поиска кратчайшего пути на графе, известный впоследствии как алгоритм Дейкстры.

Доказательство корректности алгоритма 2.22. Для доказательства выясним смысл пометок вершин.

Лемма 2.11. Пометка $d(t)$ каждой обработанной вершины — кратчайшее расстояние от источника до вершины t . Пометка $d(u)$ каждой необработанной вершины — кратчайшее расстояние от источника до этой вершины на множестве путей, у которых все вершины, кроме последней, обработаны.

Доказательство. Докажем лемму по индукции. База индукции обеспечена инициализацией: источник — обработанная вершина, и расстояние до нее (равное 0) найдено, пометки других вершин определяются длинами ребер из источника в эти вершины, их можно рассматривать как пути из одного ребра, начальная вершина которого обработана, а конец — нет.

Индукционный переход. Предположим, утверждение леммы верно для $m = k$, докажем, что после выполнения одного шага внешнего цикла оно будет верно для $m = k + 1$.

Сначала докажем, что значение пометки необработанной вершины t с минимальной пометкой — это расстояние до нее. Докажем от противного. Пусть существует более короткий путь до t . Тогда часть его вершин обработана (по крайней мере первая вершина), а часть — нет (по крайней мере последняя).

Рассмотрим первую необработанную вершину u (рисунок 2.97). Длина пути от источника до этой вершины должна быть не больше длины пути от источника до вершины t (в силу неотрицательности ребер), но тогда пометка вершины u будет меньше пометки вершины t , что противоречит правилу выбора вершины t ! Противоречие доказывает утверждение.

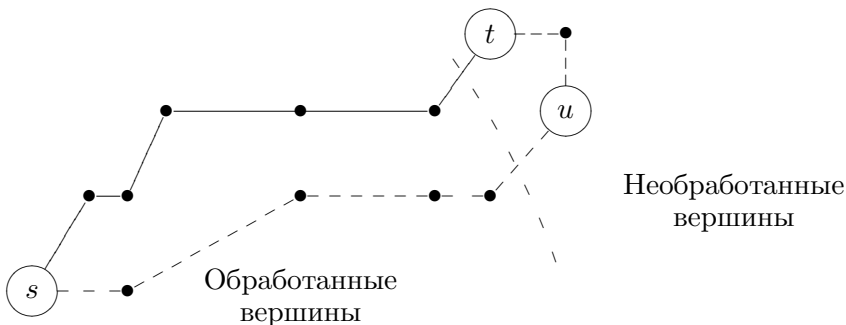


Рис. 2.97.

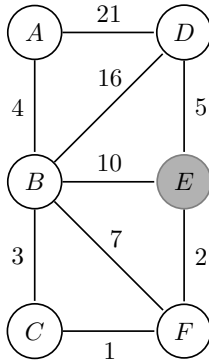
Теперь докажем справедливость утверждения леммы для необработанных вершин. Но оно следует непосредственно из алгоритма модификации — новая пометка строится так, чтобы она была минимальным расстоянием на множестве путей, у которых все вершины, кроме последней, обработаны. ■

Теорема 2.50. Алгоритм Дейкстры корректно решает задачу о нахождении кратчайших длин путей от источника до всех вершин графа.

Доказательство. Согласно лемме 2.11 на каждом шаге алгоритма одна вершина переходит в множество обработанных, следовательно, за конечное число шагов алгоритм свою работу заканчивает, а все вершины при этом оказываются обработанными. Из леммы также следует, что пометки обработанных вершин являются кратчайшими расстояниями до них. ■

Рассмотрим пример протокола работы алгоритма Дейкстры.

Пример 2.67. Найдем кратчайшие расстояния от вершины E (источника) связного неориентированного графа до остальных его вершин. Граф изображен на рисунке 2.98а, его матрица весов — на рисунке 2.98б.



(a)

	A	B	C	D	E	F
A	0	4	∞	21	∞	∞
B	4	0	3	16	10	7
C	∞	3	0	∞	∞	1
D	21	16	∞	0	5	∞
E	∞	10	∞	5	0	2
F	∞	7	1	∞	2	0

(b)

Рис. 2.98.

Протокол работы алгоритма будем вести так же, как при «прогонке» (алгоритма Форда — Беллмана (см. пример 2.65)). Первая строка протокола — строка E матрицы весов, показывающая длины ребер, выходящих из E . Вершина E уже «обработана», поэтому ее пометка не пересчитывается.

	A	B	C	D	F
E	∞	10	∞	5	2

Минимальная из этих пометок длин — пометка вершины F — есть длина кратчайшего пути от E до F . Переведем эту вершину в разряд обработанных (что в протоколе выделим серым цветом) и пересчитаем пометки вершин, смежных с вершиной F .

Для этого обратим внимание на значения весов ребер в строке F :

$$d(F) + w(F, B) = 2 + 7 = 9 < 10,$$

поэтому пометка вершины B модифицируется на 9 через вершину F ;

$$d(F) + w(F, C) = 2 + 1 = 3 < \infty,$$

поэтому пометка вершины C также модифицируется через F . Теперь выбираем минимальную из получившихся пометок — пометку вершины C — и продолжаем до тех пор, пока все вершины не будут обработаны. Полностью протокол показан в таблице 2.24.

Таблица 2.24.

Номер шага m	A	B	C	D	F
1	∞	10/ E	∞	5/ E	2/ E
2	∞	9/ F	3/ F	5/ E	
3	∞	6/ C		5/ E	
4	26/ D	6/ C			
5	10/ B				

Числовые пометки обработанных вершин показывают кратчайшие расстояния до них, а символьные — предшествующие вершины в кратчайших путях (таблица 2.24). Таким образом, дерево кратчайших путей состоит из ребер: (B, A) , (C, B) , (F, C) , (E, D) , (E, F) .

Таблица 2.25.

A	B	C	D	F
10/ B	6/ C	3/ F	5/ E	2/ E

Используя символьные пометки (таблица 2.25), мы можем получить и сами кратчайшие маршруты. Начиная от данной вершины, нужно каждый раз брать символьную пометку от текущей вершины, пока не придем в стартовую вершину — так мы получим искомым кратчайший путь, но записанный в обратном порядке:

$$A \rightarrow B \rightarrow C \rightarrow F \rightarrow E$$

$$B \rightarrow C \rightarrow F \rightarrow E$$

$$C \rightarrow F \rightarrow E$$

$$D \rightarrow E$$

$$F \rightarrow E$$

Задание 2.3. Найдите дерево кратчайших путей для графа из примера 2.67 (рисунок 2.98а) для вершины источника D . Убедитесь, что дерево в данном случае вырождается в путь, проходящий через все вершины графа по одному разу (*гамильтонов путь*).

Рассмотрим еще один пример работы алгоритма Дейкстры для ориентированного графа.

Пример 2.68. Найдем кратчайшие расстояния от вершины v_1 (источника) ориентированного графа до остальных его вершин. Граф изображен на рисунке 2.99а, его матрица весов — на рисунке 2.99b.

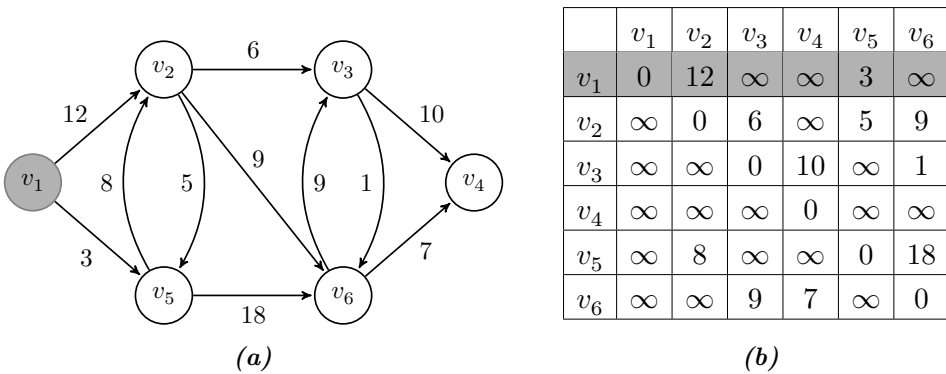


Рис. 2.99.

Протокол работы алгоритма будем вести так же, как в предыдущем примере 2.67. Окончательный протокол показан в таблице 2.26.

Таблица 2.26.

Номер шага m	v_2	v_3	v_4	v_5	v_6
1	$12/v_1$	∞	∞	$3/v_1$	∞
2	$11/v_5$	∞	∞		$21/v_5$
3		$17/v_2$	∞		$20/v_2$
4			$27/v_3$		$18/v_3$
5			$25/v_6$		

Аналогично предыдущему примеру 2.67, используя символьные пометки в таблице 2.26, получаем кратчайшие маршруты:

$$v_1 \rightarrow v_5 \rightarrow v_2,$$

$$v_1 \rightarrow v_5 \rightarrow v_2 \rightarrow v_3,$$

$$\begin{aligned}v_1 &\rightarrow v_5 \rightarrow v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_4, \\v_1 &\rightarrow v_5, \\v_1 &\rightarrow v_5 \rightarrow v_2 \rightarrow v_3 \rightarrow v_6.\end{aligned}$$

2.2.7. Сравнительный анализ трудоемкости алгоритмов Форда — Беллмана и Дейкстры

При нахождении длин кратчайших путей каждый из описанных алгоритмов многократно проверяет возможность модификации вершины и, если надо, осуществляет ее. Посчитаем количество таких проверок в зависимости от n — числа вершин графа.

Алгоритм Форда — Беллмана повторяет $(n - 2)$ раза операцию, которая включает в себя попытку модификации каждой из вершин через все (соседние) вершины графа. (При задании графа матрицей весов отсутствуют специальные указатели на соседние вершины и приходится делать проверку для всех вершин.) Таким образом, трудоемкость алгоритма Форда — Беллмана:

$$T(n) = (n - 2)n \cdot n \sim n^3.$$

Здесь знак \sim означает асимптотическую эквивалентность, т. е. предел отношения соединенных этих знаком функций на бесконечности равен константе, отличной от нуля. Следовательно, главное в оценке — показатель степени n , или в общем случае характер роста функции трудоемкости.

Использование асимптотических оценок трудоемкости оправдано, поскольку небольшие вариации алгоритма (в данном случае можно было бы исключить операции с источником и число сравнений сразу бы изменилось на $(n - 2)(n - 1)(n - 1)$) или подсчет иного множества операций (в данном случае, например, можно считать не только операции сравнения, но и арифметические операции) привели бы к изменению формулы трудоемкости, но не изменили бы порядок ее роста.

Итак, трудоемкость алгоритма Форда — Беллмана имеет порядок n^3 .

Алгоритм Дейкстры делает n шагов, связанных с обработкой вершин. В рамках каждого шага ищется минимальная пометка и через нее пересчитываются пометки соседних вершин.

Поиск минимальной пометки в простейшем случае осуществляется перебором всех n вершин (на самом деле на каждом новом шаге просматривать надо на одну вершину меньше), после чего просматриваются все (необработанные) вершины и их пометки (если надо) модифицируются. Нетрудно видеть, что порядок трудоемкости для алгоритма Дейкстры:

$$T(n) \sim n^2.$$

Замечание 2.57

Поиск минимальной пометки можно ускорить за счет иного способа структурирования данных — на этом основаны различные модификации алгоритма Дейкстры. Понятно, что для таких алгоритмов и пересчет пометок надо организовать по-другому, храня информацию о соседях каждой вершины, иначе просмотр всех вершин при модификации сведет на нет все выгоды, полученные за счет быстрого нахождения минимума.

Итак, какой выигрыш во времени дает замена алгоритма Форда — Беллмана алгоритмом Дейкстры?

Для $n = 100$ получим стократный выигрыш, т. е. если программа по второму алгоритму решит задачу за 1 с, то по первому только за 100 с.

Еще более заметен выигрыш для $n = 1000$, что для задач на графах — реальная ситуация. В этом случае на компьютере, который решит задачу с помощью алгоритма Дейкстры за 1 с, с помощью алгоритма Форда — Беллмана ее придется решать более 15 мин.

2.2.8. Алгоритм Флойда

Рассмотрим более общую задачу — нахождение всех кратчайших расстояний между всеми парами вершин графа. Можно решить эту задачу, последовательно применяя алгоритм Дейкстры для каждой вершины, объявляемой в качестве источника. Но существует прямой способ решения данной задачи — алгоритм Флойда.

Пожалуй главное достоинство этого алгоритма — простота. Как известно, с позиций надежности программирования «прозрачность» алгоритма часто имеет большее значение, чем его трудоемкость.

Постановка задачи. Найти кратчайшие расстояния между всеми парами вершин ориентированного или неориентированного взвешенного графа, не имеющего циклов отрицательной длины, а в случае, когда такой цикл есть, необходимо найти хотя бы один.

Историческая справка

Рассматриваемый алгоритм был одновременно опубликован в статьях Роберта Флойда и Стивена Уоршелла в 1962 г. Обычно его называют алгоритмом Флойда, реже — алгоритмом Флойда — Уоршелла. Впервые алгоритм был опубликован в 1959 г. Бернардом Роем, однако его публикация прошла незамеченной.

Идея алгоритма. Проводится расширение множества промежуточных вершин для путей, по которым ищутся кратчайшие расстояния.

Алгоритм начинает работу с матрицы весов (матрицы пометок $d(i, j)$), рассматриваемых как длины кратчайших путей от вершины v_i к вершине

v_j , не имеющих ни одной промежуточной вершины, т. е. вес ребра (дуги) (v_i, v_j) .

Далее поочередно каждую вершину представляют в роли промежуточной и через нее осуществляют модификацию пометки, т. е. на каждом шаге одну из вершин добавляют к множеству промежуточных вершин.

Замечание 2.58

Аналогично рассмотренным ранее алгоритму Форда — Беллмана и алгоритму Дейкстры алгоритм Флойда легко модифицировать таким образом, чтобы он возвращал не только длину кратчайшего пути, но и сам путь.

Для этого, сохранив преемственность, введем дополнительную матрицу G , в которой для каждой пары вершин u и v будем хранить номер первой промежуточной вершины кратчайшего пути из u в v .

Опишем сам алгоритм с учетом замечания 2.58.

Алгоритм 2.23. Флойда

// Инициализация

for $i \leftarrow 1, n$ **do**

for $j \leftarrow 1, n$ **do**

$d(i, j) \leftarrow w(i, j)$

// Матрицу пометок заполняем весами ребер

$G(i, j) \leftarrow i$

// v_i — первая промежуточная вершина пути $v_i \rightarrow v_j$

end for

end for

// Основной алгоритм

// Внешний цикл идет по промежуточным вершинам k , внутри него для каждой

// пары вершин проверяется возможность улучшения их пометки через k

for $k \leftarrow 1, n$ **do**

for $i \leftarrow 1, n$ **do**

for $j \leftarrow 1, n$ **do**

if $d(i, k) + d(k, j) < d(i, j)$ **then**

$d(i, j) \leftarrow d(i, k) + d(k, j)$

$G(i, j) \leftarrow G(k, j)$

end if

end for

end for

end for

// Нахождение кратчайших путей для каждой пары вершин v_i, v_j

// $L(i, j)$ — список вершин кратчайшего пути $v_i \rightarrow v_j$

for $i \leftarrow 1, n$ **do**

for $j \leftarrow 1, n$ **do**

```

     $L(i, j) \leftarrow \emptyset$ 
    if  $d(i, k) \neq \infty$  then
         $k = i$  //  $v_i$  — первая вершина пути  $L(i, j)$ 
        do
             $L(i, j) \leftarrow L(i, j) \cup \{k\}$ 
             $k \leftarrow G(k, j)$ 
        while  $k \neq G(k, j)$ 
        // Если промежуточная вершина подпути  $v_k \rightarrow v_j$  совпала с его
        // началом, то других промежуточных вершин на этом подпути нет
         $L(i, j) \leftarrow L(i, j) \cup \{j\}$ 
        // Добавляем в список  $L(i, j)$  последнюю вершину пути  $v_i \rightarrow v_j$ 
    end if
end for
end for

```

Обоснуем корректность приведенного алгоритма.

Лемма 2.12. После выполнения k -го шага (внешнего цикла) алгоритма Флойда пометки $d(i, j)$ суть длины кратчайших путей, все промежуточные вершины которых принадлежат множеству $\{1, \dots, k\}$.

Доказательство. Докажем по индукции. Как уже отмечалось, база индукции обеспечена инициализацией пометок.

Индукционный переход. Предположим, что после шага k пометки $d(i, j)$ равны кратчайшим длинам путей между парами вершин i и j на множестве путей, у которых все промежуточные вершины принадлежат множеству $\{1, \dots, k\}$.

Докажем, что после выполнения $(k + 1)$ -го шага пометки будут равны длинам кратчайших путей с промежуточными вершинами из множества $\{1, \dots, k, k + 1\}$.

Действительно, каждый из таких путей либо не проходит через вершину $(k + 1)$, и тогда его длина $d(i, j)$ нам известна, либо проходит, и тогда его длина будет равна

$$d(i, k + 1) + d(k + 1, j).$$

Заметим, что каждая вершина может входить в кратчайший путь не более одного раза, так как граф не имеет циклов отрицательной длины. Модификация пометки организована так, чтобы из этих чисел выбрать меньшее. Таким образом, индукционный переход доказан. ■

Теорема 2.51. Алгоритм Флойда корректно решает задачу нахождения всех кратчайших расстояний между всеми парами вершин графа.

Доказательство. После окончания работы внешнего цикла все вершины попадут во множество промежуточных и по лемме 2.12 пометки $d(i, j)$ будут равны длинам кратчайших путей, у которых промежуточные вершины принадлежат множеству $\{1, \dots, n\}$, т. е. множеству всех вершин графа. ■

Замечание 2.59

У алгоритма Флойда есть еще одна замечательная особенность — проверять собственную применимость: алгоритм позволяет определить наличие циклов отрицательной длины в графе.

Лемма 2.13. При наличии цикла отрицательной длины в матрице пометок $d(i, j)$ появятся отрицательные числа на главной диагонали.

Доказательство. Алгоритм Флойда последовательно уменьшает расстояния между всеми парами вершин, в том числе и петлями, а начальное значение для петли $d(i, i) = 0$.

Таким образом, уменьшение $d(i, i)$ может произойти только при наличии вершины k такой, что $d(i, k) + d(k, i) < 0$, что эквивалентно наличию отрицательного цикла, проходящего через вершину i . ■

Из леммы 2.13 следует, что для поиска цикла отрицательной длины нужно (после завершения работы алгоритма Флойда) найти вершину i , для которой $d(i, i) < 0$, и определить кратчайший путь между парой вершин (i, i) .

Рассмотрим пример работы алгоритма Флойда.

Пример 2.69. Применим алгоритм Флойда к графу, который рассматривался при обсуждении алгоритма Форда — Беллмана (см. рисунок 2.93, пример 2.65).

Как уже отмечалось, полезно следить за работой алгоритма, не глядя на граф. Поэтому не будем повторять рисунок графа, а протокол работы представим последовательными состояниями матриц пометок, которые получаются после выполнения очередного шага внешнего цикла (добавление новой промежуточной вершины).

Замечание 2.60

При работе внешнего цикла алгоритма через вершину v соответствующие ей строка и столбец матрицы пометок $d(v, i)$ и $d(i, v)$ для $i \in 1 : n$, очевидно, не меняются. Далее в протоколе они будут выделяться темно-серым цветом. Модифицированные на данном шаге пометки, как и ранее, будут отмечаться светло-серым цветом.

Начальное состояние матрицы пометок — матрица весов графа. Как и в случае алгоритма Форда — Беллмана и алгоритма Дейкстры, будем добавлять в нее символичные пометки, обозначающие вершину, через которую произошла модификация пометки.

Исходная матрица пометок имеет следующий вид:

	A	B	C	D	E
A	0	$2/A$	∞	$4/A$	$5/A$
B	$1/B$	0	$3/B$	∞	$4/B$
C	∞	$-2/C$	0	$2/C$	$1/C$
D	$-1/D$	∞	$-1/D$	0	$1/D$
E	∞	∞	∞	∞	0

A На первом шаге рассмотрим A в качестве промежуточной вершины.

Обратим внимание на начальные вершины ребер, входящих в A (столбец A), в дальнейшем назовем их входящими вершинами, и на конечные вершины ребер, выходящих из A (строка A). Через ведущую вершину A можно улучшить пометки:

$$d(B, D) > d(B, A) + d(A, D) = 1 + 4 \Rightarrow d(B, D) = 5,$$

$$d(D, B) > d(D, A) + d(A, B) = -1 + 2 \Rightarrow d(D, B) = 1.$$

Протокол первого шага представлен в следующей таблице:

	A	B	C	D	E
A	0	$2/A$	∞	$4/A$	$5/A$
B	$1/B$	0	$3/B$	$5/A$	$4/B$
C	∞	$-2/C$	0	$2/C$	$1/C$
D	$-1/D$	$1/A$	$-1/D$	0	$1/D$
E	∞	∞	∞	∞	0

B Далее в качестве промежуточной вершины рассматриваем вершину B . Через вершину B улучшаем пометки:

$$d(A, C) > d(A, B) + d(B, C) = 2 + 3 \Rightarrow d(A, C) = 5,$$

$$d(C, A) > d(C, B) + d(B, A) = -2 + 1 \Rightarrow d(C, A) = -1.$$

	A	B	C	D	E
A	0	2/A	5/B	4/A	5/A
B	1/B	0	3/B	5/A	4/B
C	-1/B	-2/C	0	2/C	1/C
D	-1/D	1/A	-1/D	0	1/D
E	∞	∞	∞	∞	0

С Следующей промежуточной вершиной будет вершина C . На этом шаге модифицируются пометки:

$$d(D, A) > d(D, C) + d(C, A) = -1 - 1 \Rightarrow d(D, A) = -2,$$

$$d(D, B) > d(D, C) + d(C, B) = -1 - 2 \Rightarrow d(D, B) = -3,$$

$$d(D, E) > d(D, C) + d(C, E) = -1 + 1 \Rightarrow d(D, E) = 0.$$

	A	B	C	D	E
A	0	2/A	5/B	4/A	5/A
B	1/B	0	3/B	5/A	4/B
C	-1/B	-2/C	0	2/C	1/C
D	-2/C	-3/C	-1/D	0	0/C
E	∞	∞	∞	∞	0

D Следующий шаг — модификация через вершину D .

	A	B	C	D	E
A	0	1/D	5/B	4/A	4/D
B	1/B	0	3/B	5/A	4/B
C	-1/B	-2/C	0	2/C	1/C
D	-1/D	-3/C	-1/D	0	1/D
E	∞	∞	∞	∞	0

На этом шаге модифицируются две пометки:

$$d(A, B) > d(A, D) + d(D, B) = 4 - 3 \Rightarrow d(A, B) = 1,$$

$$d(A, E) > d(A, D) + d(D, E) = 4 + 0 \Rightarrow d(A, E) = 4.$$

E На последнем шаге с промежуточной вершиной E модификаций пометок нет.

Рассмотрим работу алгоритма Флойда для построения кратчайших путей по символьным пометкам.

Как в примере 2.65, найдем кратчайший путь от A до B . Согласно алгоритму Флойда имеем:

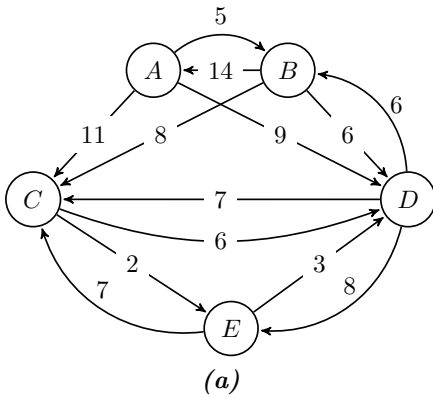
$$G(A, B) = D, G(D, B) = C, G(C, B) = C.$$

Таким образом кратчайший путь $A \rightarrow B$:

$$A \rightarrow D \rightarrow C \rightarrow B.$$

Рассмотрим еще один пример на работу алгоритма Флойда.

Пример 2.70. Исходный граф представлен на рисунке 2.100a, матрица его весов — на рисунке 2.100b.



	A	B	C	D	E
A	0	5/A	11/A	9/A	∞
B	14/B	0	8/B	6/B	∞
C	∞	∞	0	6/C	2/C
D	∞	6/D	7/D	0	8/D
E	∞	∞	7/E	3/E	0

(a)

(b)

Рис. 2.100.

А Начинаем работу с вершины A в качестве промежуточной. Из вершин C, D и E в вершину A не попасть, пути из вершины B через вершину A тоже не улучшить.

В Переходим к вершине B . На этом шаге модифицируются одна пометка:

$$d(D, A) > d(D, B) + d(B, A) = 6 + 14 \Rightarrow d(D, A) = 20.$$

	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
<i>A</i>	0	5/ <i>A</i>	11/ <i>A</i>	9/ <i>A</i>	∞
<i>B</i>	14/ <i>B</i>	0	8/ <i>B</i>	6/ <i>B</i>	∞
<i>C</i>	∞	∞	0	6/ <i>C</i>	2/ <i>C</i>
<i>D</i>	20/ <i>B</i>	6/ <i>D</i>	7/ <i>D</i>	0	8/ <i>D</i>
<i>E</i>	∞	∞	7/ <i>E</i>	3/ <i>E</i>	0

С Следующая промежуточная вершина — *C*. На этом шаге модифицируются пометки:

$$d(A, E) > d(A, C) + d(C, E) = 11 + 2 \Rightarrow d(A, E) = 13,$$

$$d(B, E) > d(B, C) + d(C, E) = 8 + 2 \Rightarrow d(D, E) = 10.$$

	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
<i>A</i>	0	5/ <i>A</i>	11/ <i>A</i>	9/ <i>A</i>	13/ <i>C</i>
<i>B</i>	14/ <i>B</i>	0	8/ <i>B</i>	6/ <i>B</i>	10/ <i>C</i>
<i>C</i>	∞	∞	0	6/ <i>C</i>	2/ <i>C</i>
<i>D</i>	20/ <i>B</i>	6/ <i>D</i>	7/ <i>D</i>	0	8/ <i>D</i>
<i>E</i>	∞	∞	7/ <i>E</i>	3/ <i>E</i>	0

D Переходим к вершине *D*. На этом шаге модифицируются следующие пометки:

$$d(C, A) > d(C, D) + d(D, A) = 6 + 20 \Rightarrow d(C, A) = 26,$$

$$d(C, B) > d(C, D) + d(D, B) = 6 + 6 \Rightarrow d(C, B) = 12,$$

$$d(E, A) > d(E, D) + d(D, A) = 3 + 20 \Rightarrow d(E, A) = 23,$$

$$d(E, B) > d(E, D) + d(D, B) = 3 + 6 \Rightarrow d(E, B) = 9.$$

	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
<i>A</i>	0	5/ <i>A</i>	11/ <i>A</i>	9/ <i>A</i>	13/ <i>C</i>
<i>B</i>	14/ <i>B</i>	0	8/ <i>B</i>	6/ <i>B</i>	10/ <i>C</i>
<i>C</i>	26/ <i>D</i>	12/ <i>D</i>	0	6/ <i>C</i>	2/ <i>C</i>
<i>D</i>	20/ <i>B</i>	6/ <i>D</i>	7/ <i>D</i>	0	8/ <i>D</i>
<i>E</i>	23/ <i>D</i>	9/ <i>D</i>	7/ <i>E</i>	3/ <i>E</i>	0

Ⓔ Последняя промежуточная вершина — E . На этом шаге модифицируются пометки:

$$d(C, A) > d(C, E) + d(E, A) = 2 + 23 \Rightarrow d(C, A) = 25,$$

$$d(C, B) > d(C, E) + d(E, B) = 2 + 9 \Rightarrow d(C, B) = 11,$$

$$d(C, D) > d(C, E) + d(E, D) = 2 + 3 \Rightarrow d(C, D) = 5.$$

	A	B	C	D	E
A	0	$5/A$	$11/A$	$9/A$	$13/C$
B	$14/B$	0	$8/B$	$6/B$	$10/C$
C	$25/E$	$11/E$	0	$5/E$	$2/C$
D	$20/B$	$6/D$	$7/D$	0	$8/D$
E	$23/D$	$9/D$	$7/E$	$3/E$	0

Найдем для примера кратчайший путь от C до A . Согласно алгоритму Флойда имеем:

$$G(C, A) = E, G(E, A) = D, G(D, A) = B, G(B, A) = B.$$

Таким образом кратчайший путь $C \rightarrow A$:

$$C \rightarrow E \rightarrow D \rightarrow B \rightarrow A.$$

Задание 2.4. Сравните алгоритм Уоршелла для построения транзитивного замыкания и алгоритм Флойда. При каких значениях весов алгоритм Флойда превращается в алгоритм Уоршелла?

2.2.9. Волновой алгоритм

Рассмотрим еще одну специальную задачу поиска пути в графе — поиск кратчайшего пути в лабиринте. Для ее решения был предложен *волновой алгоритм* (*алгоритм Ли*). Алгоритм основан на методе поиска в ширину. В основном алгоритм используется при разводке печатных плат, поиске кратчайшего расстояния на карте в компьютерных играх.



Историческая справка

Волновой алгоритм для поиска пути в лабиринте был предложен Э. Ф. Муром. Ли независимо открыл этот же алгоритм при формализации задачи разводки печатных плат в 1961 г.

Алгоритм работает на ограниченной замкнутой линией фигуре (не обязательно прямоугольной), разбитой на прямоугольные ячейки, в частном случае — квадратные. Множество всех ячеек разбивается на подмножества: *проходимые* (свободные), т. е. при поиске пути их можно проходить, *непроходимые* (препятствия), путь через эту ячейку запрещен, стартовая ячейка (источник) и финишная (приемник), между которыми нужно найти кратчайший путь, если это возможно, либо при отсутствии пути выдать сообщение о непроходимости.

Определение 2.58. *Соседними* ячейками для данной ячейки по фон Нейману считаются только 4 ячейки по вертикали и горизонтали, по Муру — все 8 ячеек, включая диагональные.

Работа алгоритма включает в себя два этапа: распространение волны и восстановление пути.

Алгоритм 2.24. Алгоритм Ли: распространение волны

Исходные данные: G — множество ячеек, $s \in G$ — стартовая, $f \in G$ — финишная

// Инициализация: помечаем стартовую ячейку, остальные — не помечены

$d \leftarrow 0$, $m(s) = d$

$m(u) = -1 \mid u \in G \setminus \{s\}$

// Основной алгоритм

while $m(f) = -1$ и можно помечать новые ячейки **do**

for each $u \in G \mid m(u) = d$ **do**

 // рассматриваем соседние по Муру или по фон Нейману — определение 2.58

 помечаем все соседние с u свободные непомяченные ячейки числом $d + 1$

end for

$d \leftarrow d + 1$

end while

Если финишная ячейка не помечена, то пути не существует.

Восстановление кратчайшего пути происходит в обратном направлении: при выборе ячейки от финишной ячейки к стартовой на каждом шаге выбирается ячейка, имеющая расстояние от стартовой (d) на единицу меньше текущей ячейки. Очевидно, что таким образом находится кратчайший путь.

Некоторые модификации волнового алгоритма предполагают сохранение информации о том, из какой вершины волна перешла в данную, что ускоряет генерацию пути, но занимает больше памяти.

Преимущества волнового алгоритма в том, что с его помощью можно найти трассу в любом лабиринте и с любым количеством препятствий. Единственным недостатком волнового алгоритма является то, что при построении пути требуется большой объем памяти.

Замечание 2.61

Путей с минимальной числовой длиной пути, как при поиске пути по Муру, так и по фон Нейману, может существовать несколько. Выбор окончательного пути в приложениях диктуется другими соображениями, находящимися вне этого алгоритма. Однако длина пути в любом случае остается одной, вне зависимости от выбранной стратегии.

Пример 2.71. Рассмотрим работу алгоритма на примере лабиринта на рисунке 2.101.

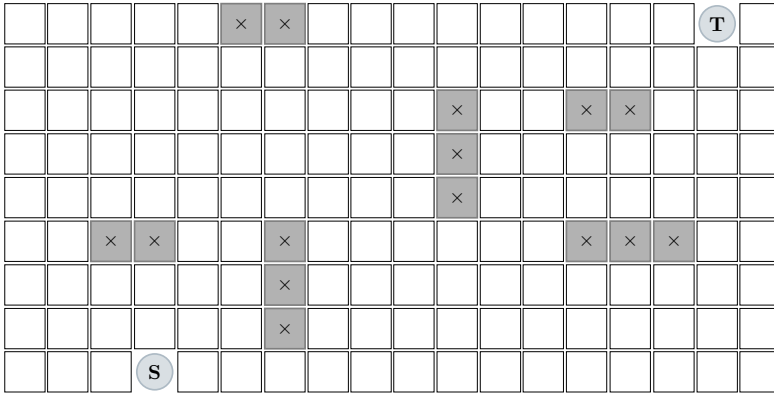


Рис. 2.101.

На рисунке 2.102 показан этап распространения волны. Заметим, что в данном примере существует несколько кратчайших путей.

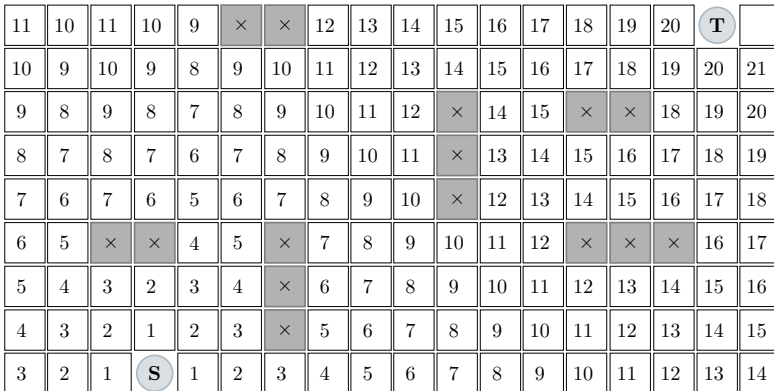


Рис. 2.102.

Пример 2.72. В этом примере (рисунок 2.103a и b) такой путь единственный. Отмечен светло-серым цветом.

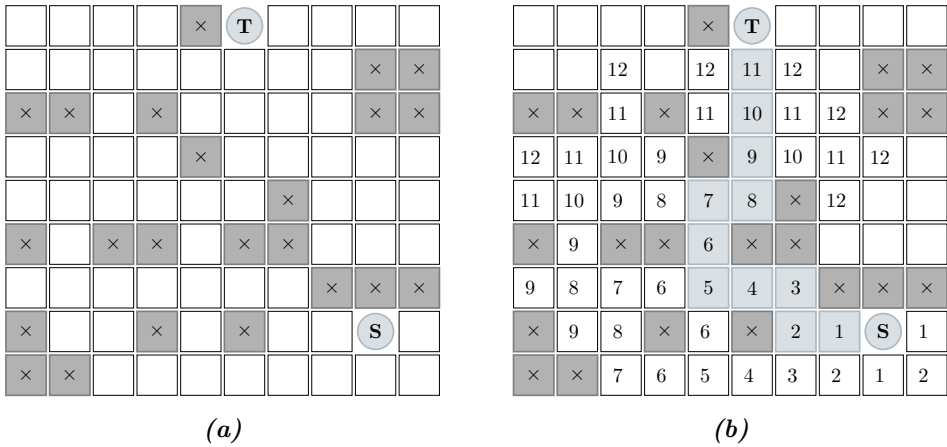


Рис. 2.103.

2.2.10. Алгоритм Форда — Фалкерсона

Историческая справка

Впервые задача о максимальном потоке была решена Джорджем Данцигом в ходе подготовки воздушного моста во время блокады Западного Берлина, происходившей в 1948–1949 гг. Позднее, в 1951 г., он сформулировал задачу в общем виде. В 1955 г., Лестер Форд и Делберт Фалкерсон первыми разработали алгоритм, специально предназначенный для решения этой задачи. Он получил название алгоритм Форда — Фалкерсона. В дальнейшем решение задачи многократно улучшалось различными авторами.

Определение 2.59. *Транспортной сетью* (или просто *сетью*) S называют ориентированный взвешенный граф G с положительными весами дуг, в котором существует ровно один узел s (*источник*), в который не входит ни одна дуга (его отрицательная валентность равна 0), и ровно один узел t (*сток*), из которого не выходит ни одной дуги (его положительная валентность равна 0):

$$|\delta^-(s)| = 0, \quad |\delta^+(t)| = 0.$$

Вес дуги $[u, v]$ будем обозначать $c(u, v)$.

Смысл весов дуг в данной задаче — пропускные способности дуг.

Пример 2.73. Для сети автомобильных дорог пропускная способность дуги — количество автомобилей, проезжающих в единицу времени, для электрической сети — проводимость (величина, обратная сопротивлению) участка цепи.

Определение 2.60. *Потоком* в сети называют функцию, сопоставляющую каждой дуге неотрицательное число (*величину потока через дугу*) так, чтобы:

- 1) оно не превышало пропускной способности дуги;
- 2) для каждого промежуточного узла (т. е. любого узла, кроме s и t) выполнялось следующее условие: сумма потоков по входящим в узел дугам была равна сумме потоков по дугам, исходящим из нее.

Более формально, функция $f : \mathbf{E} \rightarrow \mathbb{R}_+$, определенная на множестве дуг графа $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, называется потоком в сети S , если она удовлетворяет следующим условиям:

- 1) для любой дуги $[u, v] \in \mathbf{E} : 0 \leq f(u, v) \leq c(u, v)$;
- 2) для любого узла $v \in \mathbf{V} \setminus \{s, t\}$:

$$\sum_{u \in \mathbf{V} | [v, u] \in \mathbf{E}} f(v, u) = \sum_{u \in \mathbf{V} | [u, v] \in \mathbf{E}} f(u, v). \quad (2.33)$$

Пример 2.74. На рисунке 2.104а показан пример транспортной сети. Потоки на ней представлены на рисунке 2.104b и c.

Весы дуг — их пропускные способности, а величины в скобках — значения потоков через дуги. «Активные» дуги (с положительными значениями потоков) отмечены светло-серым цветом.

Заметим, что ребро (u, v) неориентированное. Его можно заменить двумя по-разному ориентированными дугами с одинаковыми пропускными способностями.

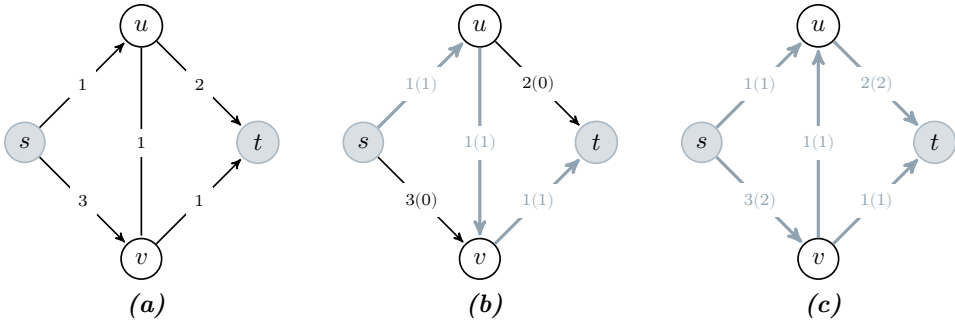


Рис. 2.104.

Определение 2.61. Разность (2.33) суммарных потоков через входящие и выходящие в узел v дуги называют *дивергенцией потока* в данном узле и обозначают $\text{div}(f, v)$.

Равенство дивергенции нулю в данном узле означает, что он не является ни источником, ни стоком.

Пример 2.75. Равенство дивергенции нулю в промежуточных узлах транспортной сети для автомобильных потоков означает невозможность

приостановления движения и отправки новых автомобилей с промежуточных пунктов маршрута.

Упражнение 2.9. Вычислите $\operatorname{div}(s)$ и $\operatorname{div}(t)$.

Определение 2.62. $s - t$ разрезом транспортной сети P называется множество дуг, соединяющих узлы множеств V_1 и V_2 , где V_1, V_2 — разбиение множества узлов V графа на два непересекающихся подмножества, обладающих свойствами: $s \in V_1$ и $t \in V_2$.

Более формально множество дуг P графа G называется $s - t$ разрезом, если оно состоит из всех дуг $[u, v]$ таких, что:

$u \in V_1, v \in V_2$, или $u \in V_2, v \in V_1$, где $V = V_1 \cup V_2, V_1 \cap V_2 = \emptyset, s \in V_1, t \in V_2$.

В дальнейшем, где это не приводит к неоднозначности, $s - t$ разрез будем называть разрезом. Ребра разреза P с началом в V_1 обозначим P^+ , а с началом в V_2 — соответственно, P^- .

Определение 2.63. Пропускной способностью $C(P)$ разреза P назовем сумму пропускных способностей дуг, образующих разрез:

$$C(P) = \sum_{[u,v] \in P} c(u, v).$$

Пусть $E_1 \subseteq E$ — подмножество дуг транспортной сети. Обозначим $F(E_1)$ сумму потоков его дуг:

$$F(E_1) = \sum_{[u,v] \in E_1} f(u, v).$$

Определение 2.64. Величиной потока $w(P)$ через разрез P назовем разность $F(P^+) - F(P^-)$.

Пример 2.76. Рассмотрим разрез $V_1 = \{s\}, V_2 = V \setminus \{s\}$. Величина потока через этот разрез равна сумме потоков через исходящие из начального узла дуги.

Упражнение 2.10. Вычислите величину потока через разрез

$$V_1 = V \setminus \{t\}, V_2 = \{t\}.$$

Теорема 2.52. Величина потока через разрез $w(P)$ не превышает пропускной способности разреза $C(P)$.

Доказательство. Запишем последовательность неравенств:

$$\begin{aligned} w(P) = F(P^+) - F(P^-) &\leq F(P^+) = \sum_{[u,v] \in P^+} f(u,v) \leq \sum_{[u,v] \in P} f(u,v) \leq \\ &\leq \sum_{[u,v] \in P} c(u,v) = C(P). \end{aligned}$$

■

Теорема 2.53. Величина потока через разрез не зависит от выбора разреза.

Доказательство. Рассмотрим произвольный разрез графа P , определяемый разбиением V на V_1 и V_2 , и просуммируем дивергенции по узлам V_1 . По определению потока дивергенции в промежуточных узлах равны нулю, поэтому

$$\sum_{v \in V_1} \operatorname{div}(v) = \operatorname{div}(s).$$

Вычислим эту сумму по-другому, используя определение дивергенции.

$$\begin{aligned} \sum_{v \in V_1} \operatorname{div}(v) &= \sum_{v \in V_1} \left(\sum_{u|[u,v] \in E} f(u,v) - \sum_{u|[v,u] \in E} f(v,u) \right) = \\ &= \sum_{v \in V_1} \sum_{u|[u,v] \in E} f(u,v) - \sum_{v \in V_1} \sum_{u|[v,u] \in E} f(v,u). \end{aligned}$$

Заметим, что двойное суммирование можно рассматривать как суммирование по дугам, а каждая дуга, концом которой является промежуточный узел, входит один раз в первую сумму и один раз во вторую. Таким образом, после упрощения в сумме остаются только члены, соответствующие дугам разреза:

$$\sum_{v \in V_1} \sum_{u \in V_2} f(u,v) - \sum_{v \in V_1} \sum_{u \in V_2} f(v,u) = F(P^+) - F(P^-) = F(P).$$

Следовательно, для любого разреза P величина $F(P)$ потока через разрез равна $\operatorname{div}(s)$. ■

Определение 2.65. Величину $\operatorname{div}(s)$ называют величиной потока P .

Пример 2.77. В терминах сети автомобильных дорог это определение означает, что величина потока определяется как количество автомобилей, выезжающих из начального пункта транспортной сети в единицу времени.

Лемма 2.14. Максимальная величина потока не превышает минимальной пропускной способности разреза.

Доказательство. Величина потока равна величине потока через любой разрез, значит, величина любого потока не превышает минимальной пропускной способности разреза, это верно и для максимального потока. ■

Можно предположить, что верно и более сильное утверждение.

Теорема 2.54 (Форд — Фалкерсон). Максимальная величина потока равна минимальной пропускной способности разреза.

Замечание 2.62

Теорему 2.54 будем доказывать для сетей с целочисленными пропускными способностями дуг.

Лемма 2.15 (об увеличивающей цепи). Если величина потока меньше минимальной пропускной способности разреза, то существует цепь от начального узла до конечного, обладающая свойством: величина потока через каждую дугу этой цепи:

1) меньше пропускной способности этой дуги (такие дуги называются *ненасыщенными*), если направление дуги согласуется с направлением движения от начального узла к конечному (такие дуги называются *согласованными*) и

2) ненулевая, если направление дуги противоположное.

Такую цепь будем называть *увеличивающей цепью*.

Доказательство. Докажем от противного. Пусть такой цепи не существует. Введем пометки дуг. Сначала пометим узел s . Далее будем помечать узлы графа следующим образом: если выходящие из помеченных узлов дуги не насыщены, то концы этих дуг помечаем, также помечаем начальные узлы всех входящих дуг с ненулевым потоком.

Узел t по предположению не должен оказаться помеченным. Тогда если обозначить V_1 — множество помеченных узлов, а $V_2 = V \setminus V_1$ — множество не помеченных, то дуги, соединяющие V_1 с V_2 , определяют разрез сети.

Этот разрез не содержит ни одной ненасыщенной дуги, идущей от V_1 в V_2 , и ни одной дуги, ведущей из V_2 в V_1 , с положительной величиной потока, значит, поток через этот разрез совпадает с его пропускной способностью, что противоречит условию и доказывает лемму. ■

Приведем теперь конструктивное доказательство теоремы 2.54, которое называют *алгоритмом Форда — Фалкерсона*.

В качестве начального потока возьмем нулевой. Если этот поток меньше минимальной пропускной способности разреза, то по лемме 2.15 найдем

увеличивающую цепь. Вдоль этой цепи величину потока можно увеличить на 1, добавляя к величинам потока по согласованным дугам 1 и вычитая по 1 из остальных дуг.

Так будем повторять до тех пор, пока величина потока не достигнет минимальной пропускной способности разреза. Процесс обязательно остановится, так как величина потока на каждом шаге увеличивается на 1.

Замечание 2.63

Перенос доказательства на случай вещественных пропускных способностей предполагает увеличение потока вдоль улучшающего пути на величину, равную минимальной разности между пропускными способностями дуг и величинами потока через дуги этого пути. Эта величина на каждом шаге может уменьшаться, что не гарантирует в общем случае сходимость процесса.

Алгоритм 2.25. Форда — Фалкерсона

```
// Обозначения:
//  $s$  — начальный узел,  $t$  — конечный узел,  $m(v)$  — пометка узла  $v$ 
// пометки узлов будут использоваться для нахождения увеличивающей цепи
//  $F$  — величина максимального потока

// Инициализация потока нулевыми значениями
for each  $(u, v) \in V \mid [u, v] \in E$  do
     $f(u, v) \leftarrow 0$ 
end for

// Основной цикл
repeat
    // Расставляем пометки узлов поиском в ширину
    // Рассматриваем дуги согласно лемме 2.15
     $m(s) \leftarrow s$ 
     $V_1 \leftarrow \{s\}$ 
     $V_2 \leftarrow V \setminus \{s\}$ 
    for each  $u \in V_1$  do
        for each  $w \in V_2 \mid [w, u] \in E \vee [u, w] \in E$  do
            if  $(f(u, w) < c(u, w)) \vee (f(w, u) > 0)$  then
                 $m(w) \leftarrow u$ 
                // Переносим  $w$  из  $V_2$  в  $V_1$ 
                 $V_1 \leftarrow V_1 \cup \{w\}$ 
                 $V_2 \leftarrow V_2 \setminus \{w\}$ 
            end if
        end for
    end for
end repeat
```

```

// Увеличение потока на 1 вдоль увеличивающей цепи.
// Цепь строим по пометкам узлов  $m(v)$  от стока  $t$  к источнику  $s$ 
if  $t \in V_1$  then
   $v \leftarrow t$ 
  while  $v \neq s$  do
     $z \leftarrow v$ 
     $v \leftarrow m(v)$ 
    if  $f(v, z) < c(v, z)$  then
      // На согласованной дуге увеличиваем поток
       $f(v, z) \leftarrow f(v, z) + 1$ 
    else
      if  $f(z, v) > 0$  then
        // На несогласованной дуге с ненулевым потоком уменьшаем
        // поток
         $f(z, v) \leftarrow f(z, v) - 1$ 
      end if
    end if
  end while
end if
until  $t \in V_1$ 

// Вычисление максимального потока. Согласно теореме 2.53 вычисляем  $\text{div}(s)$ .
 $F \leftarrow 0$ 
for each  $w \in V \mid [w, s] \in E \vee [s, w] \in E$  do
   $F \leftarrow F + f(s, w)$ 
end for

```

Замечание 2.64

После завершения алгоритма согласно лемме 2.15 множества дуг, соединяющих V_1 с V_2 , определяют разрез сети.

Пример 2.78. На рисунке 2.105 показана работа алгоритма Форда — Фалкерсона для транспортной сети с источником s и стоком t .

Неориентированное ребро (u, v) на рисунке 2.105а заменяем двумя ориентированными дугами (см. рисунок 2.105b).

Поток F инициализируется нулевыми значениями (рисунок 2.105с), далее осуществляется постановка пометок и выбор увеличивающей цепи.

На первом шаге построено множество V_1 (в скобках, значения пометок), увеличивающаяся цепь P и поток увеличен вдоль цепи.

$$V_1 = \{u(s), v(s), t(u)\}, \quad P = s \xrightarrow{f(s,u)=1} u \xrightarrow{f(u,t)=1} t.$$

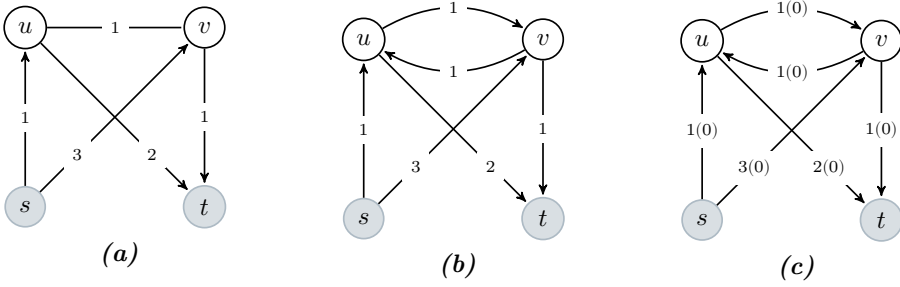


Рис. 2.105.

На рисунке 2.106a дуги увеличивающей цепи отмечены серым цветом.

На втором шаге построены множество V_1 , увеличивающаяся цепь P и поток увеличен вдоль цепи (рисунок 2.106b).

$$V_1 = \{v(s), u(v), t(v)\}, \quad P = s \xrightarrow{f(s,v)=1} v \xrightarrow{f(v,t)=1} t.$$

Результаты третьего шага (рисунок 2.106c).

$$V_1 = \{v(s), u(v), t(u)\}, \quad P = s \xrightarrow{f(s,v)=2} v \xrightarrow{f(v,u)=1} u \xrightarrow{f(u,t)=2} t.$$

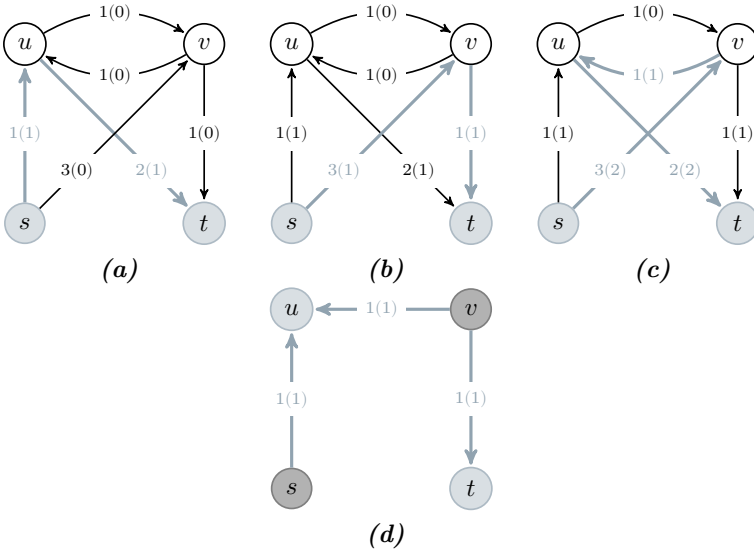


Рис. 2.106.

На следующем шаге $V_1 = \{s, v\}$ и конечный узел $t \notin V_1$. Значит, найден максимальный поток. Непомеченные узлы u и t образуют множество V_2 .

По замечанию 2.64 дуги $[s, u]$, $[v, u]$ и $[v, t]$ образуют минимальный разрез (рисунок 2.106d). Его пропускная способность равна 3.

Согласно алгоритму Форда — Фалкерсона можно вычислить

$$\text{div}(s) = f(s, u) + f(v, u) + f(v, t) = 3.$$

Если транспортную сеть можно представить плоским графом, то алгоритм Форда — Фалкерсона можно упростить. Идея в том, чтобы на каждом шаге находить самый «верхний» ориентированный путь из возможных путей и загружать его как можно больше.

Алгоритм 2.26. Максимальный поток в плоской транспортной сети

```

// Обозначения:
//  $s$  — начальный узел,  $t$  — конечный узел
//  $F$  — величина максимального потока

// Инициализация
 $F \leftarrow 0$ 

// Основной цикл
while существует ориентированный путь из  $s$  в  $t$  do
  // находим самый верхний ориентированный путь в сети  $W$ 
   $W : s = v_1 \rightarrow \dots \rightarrow v_k = t$ 

  // находим минимальную пропускную способность дуг  $W$ 
   $c_{min} = \min \{c(v_i, v_{i+1}) \mid v_i, v_{i+1} \in W\}$ 

  // уменьшаем пропускные способности дуг пути  $W$  на величину  $c_{min}$ 
  for each  $[v_i, v_{i+1}] \in E, v_i, v_{i+1} \in W$  do
     $c(v_i, v_{i+1}) \leftarrow c(v_i, v_{i+1}) - c_{min}$ 

    // удаляем из сети дуги с нулевой пропускной способностью
    if  $c(v_i, v_{i+1}) = 0$  then
       $E \leftarrow E \setminus [v_i, v_{i+1}]$ 
    end if
  end for

   $F \leftarrow F + c_{min}$ 
end while

```

Пример 2.79. Применим данный алгоритм к транспортной сети из примера 2.78, изобразив ее в виде плоского графа.

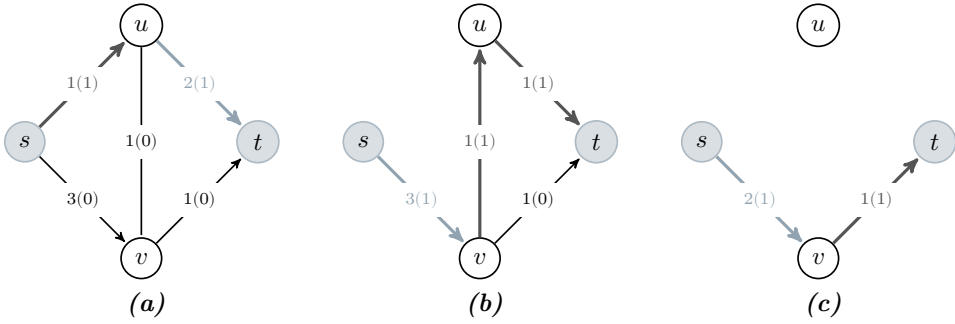


Рис. 2.107.

Таблица 2.27.

Шаг	Иллюстрация	Самый верхний путь	Значение c_{min}	Значение F
1	рисунок 2.107a	$s \rightarrow u \rightarrow t$	1	1
2	рисунок 2.107b	$s \rightarrow v \rightarrow u \rightarrow t$	1	2
3	рисунок 2.107c	$s \rightarrow v \rightarrow t$	1	3

На рисунке 2.107 показан протокол работы алгоритма для плоской транспортной сети с источником s и стоком t .

Комментарии к работе алгоритма приведены в таблице 2.27.

Задача 2.5. Докажите, что алгоритм 2.26 решает задачу нахождения максимального потока плоской транспортной сети.

Пример 2.80. Рассмотрим еще один пример работы алгоритма 2.26 для транспортной сети с источником s и стоком t на рисунке 2.108a.

Протокол работы алгоритма приведен в таблице 2.28.

Таблица 2.28.

Шаг	Иллюстрация	Самый верхний путь	Значение c_{min}	Значение F
1	рисунок 2.108b	$s \rightarrow v_1 \rightarrow v_2 \rightarrow t$	3	3
2	рисунок 2.108c	$s \rightarrow v_3 \rightarrow v_1 \rightarrow v_2 \rightarrow t$	2	5
3	рисунок 2.108d	$s \rightarrow v_3 \rightarrow v_4 \rightarrow t$	2	7

Для целочисленных значений пропускных способностей алгоритм 2.25 можно ускорить.

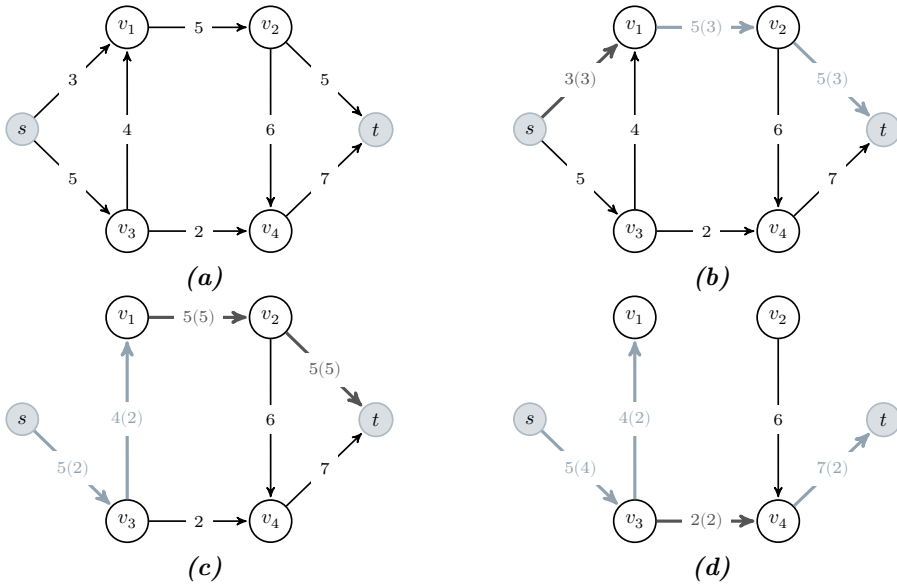


Рис. 2.108.

Будем увеличивать поток по согласованным дугам и уменьшать его по обратным дугам улучшающего пути на величину, равную минимальной пропускной способности дуг (c_{min} в обозначениях алгоритма 2.26).

Пусть алгоритмом 2.25 по пометкам узлов $m(v)$ построена улучшающая цепь P от источника s к стоку t :

$$P = e_1 \rightarrow e_2 \rightarrow \dots \rightarrow e_k.$$

Вычисляем

$$c_{min} = \min \{c(u_i, v_i) - f(u_i, v_i) \mid [u_i, v_i] \in P\}.$$

Далее изменяем поток на величину c_{min} вдоль увеличивающей цепи.

$$f(u_i, v_i) = \begin{cases} f(u_i, v_i) + c_{min}, & \text{если } [u_i, v_i] \text{ — согласованная дуга, и} \\ f(u_i, v_i) - c_{min}, & \text{в противном случае.} \end{cases}$$

Замечание 2.65

Основной цикл в алгоритме 2.25 можно не проводить до конца, а останавливать, как только пометку получает вершина сток t .

Пример 2.81. С помощью модифицированного алгоритма 2.25 построим максимальный поток алгоритмом Форда — Фалкерсона для транс-

портной сети с источником s и стоком t на рисунке 2.109. Поток уже инициализирован нулевыми значениями.

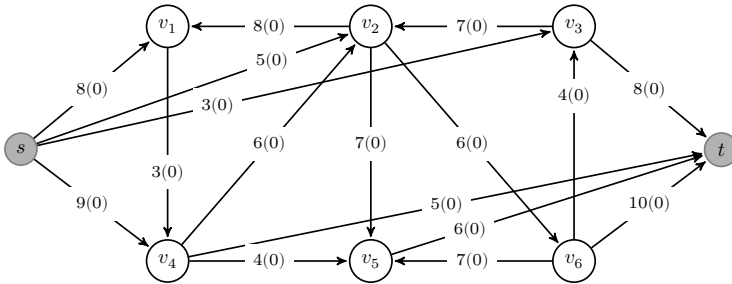


Рис. 2.109.

Протокол работы алгоритма приведен в таблице 2.29. В увеличивающем пути сверху проставлены текущие значения потока.

Таблица 2.29.

V_1	P	c_{min}
$v_1(s), v_2(s), v_3(s), v_4(s), v_5(v_2), v_6(v_2), t(v_3)$	$s \xrightarrow{3} v_3 \xrightarrow{3} t$	3
$v_1(s), v_2(s), v_4(s), v_5(v_2), v_6(v_2), t(v_5), v_3(v_6)$	$s \xrightarrow{5} v_2 \xrightarrow{5} v_5 \xrightarrow{5} t$	5
$v_1(s), v_4(s), v_2(v_4), v_5(v_4), t(v_4), v_6(v_2), v_3(v_6)$	$s \xrightarrow{5} v_4 \xrightarrow{5} t$	5
$v_1(s), v_4(s), v_2(v_4), v_5(v_4), v_6(v_2), t(v_5), v_3(v_6)$	$s \xrightarrow{6} v_4 \xrightarrow{1} v_5 \xrightarrow{6} t$	1
$v_1(s), v_4(s), v_2(v_4), v_5(v_4), v_6(v_2), v_3(v_6), t(v_6)$	$s \xrightarrow{9} v_4 \xrightarrow{3} v_2 \xrightarrow{3} v_6 \xrightarrow{3} t$	3
$v_1(s), v_4(v_1), v_2(v_4), v_5(v_4), v_6(v_2), v_3(v_6), t(v_6)$	$s \xrightarrow{3} v_1 \xrightarrow{3} v_4 \xrightarrow{6} v_2 \xrightarrow{6} v_6 \xrightarrow{6} t$	3

На очередном шаге в множество V_1 попадают только узлы s, v_1 . Все остальные узлы образуют множество V_2 .

Алгоритм заканчивает работу. Результирующий поток представлен на рисунке 2.110.

Дуги с положительной пропускной способностью, вошедшие в разрез: $[s, v_2], [s, v_3], [s, v_4], [v_1, v_4]$ (обозначены на рисунке 2.110 темно-серым цветом). Максимальный поток F :

$$F = \text{div}(s) = f(s, v_2) + f(s, v_3) + f(s, v_4) + f(v_1, v_4) = 20.$$

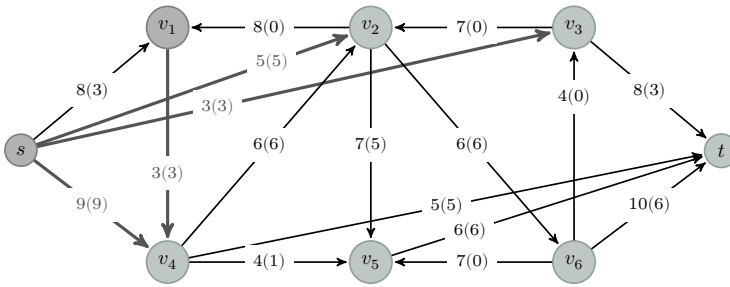


Рис. 2.110.

2.2.11. Перебор с возвратом

Для множества задач дискретной математики эффективных алгоритмов не найдено и, скорее всего, не существует.

Рассмотрим структуру переборных алгоритмов с возвратом, осуществляющих последовательный перебор всех допустимых комбинаций. Аналогичные перечислительные задачи были разобраны в комбинаторике. Однако если там структура перебираемого множества была определена заранее, то здесь она определяется динамически с помощью функций расширения частичных решений.

Приведем основные понятия, связанные с алгоритмом перебора с возвратом, на примере известной задачи о восьми ферзях (шахматного композитора Макса Беззеля). Как уже отмечалось (см. замечание 2.37), эта задача сводится к отысканию наибольшего независимого множества вершин в графе.

Задача 2.6. Расставить на шахматной доске восемь ферзей так, чтобы они не били друг друга.

Для простоты рассмотрим доску размерами 4×4 и соответственно с четырьмя ферзями.

Для осуществления перебора заметим, что каждый ферзь должен стоять на отдельной вертикали. Поставим 1-го ферзя на первую клетку первой вертикали и запомним частичное решение как $(A1)$.

Теперь будем пытаться поставить 2-го ферзя на вторую вертикаль так, чтобы он и 1-й не били друг друга. Первая и вторая позиции вертикали не подходят, поэтому ферзя ставим на $B3$. Полученное частичное решение запомним как $(A1 \rightarrow B3)$. Попробуем расширить полученное частичное решение, но все клетки третьей вертикали находятся под боем поставленных ферзей. Таким образом, мы зашли в тупик.

В этом случае, используя хранящееся частичное решение, делаем шаг назад и пробуем следующее расширение первого частичного решения:

$(A1 \rightarrow B4)$. Можно сделать еще один шаг: $(A1 \rightarrow B4 \rightarrow C2)$, однако дальнейшее расширение невозможно (рисунок 2.111).

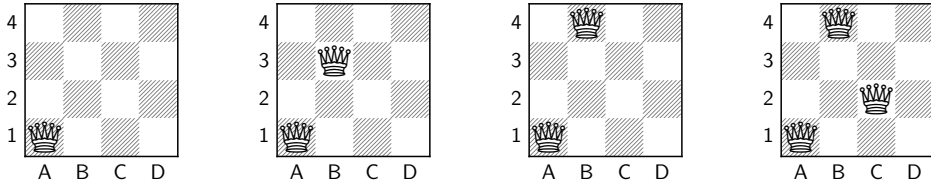


Рис. 2.111.

Возвращаемся на шаг назад, но других расширений для $(A1 \rightarrow B4)$ нет. Делаем еще шаг назад, но у $(A1)$ также нет других расширений, поэтому делаем еще шаг назад и меняем первый ход на $(A2)$.

Ход успешного решения из позиции $(A2)$ представлен на рисунке 2.112.

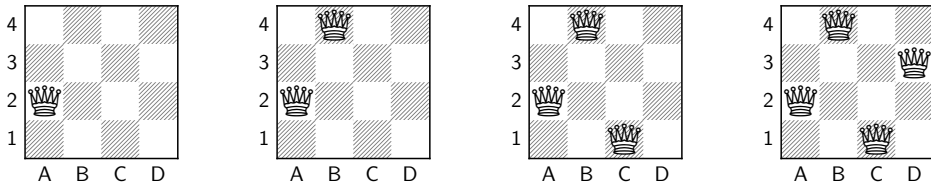


Рис. 2.112.

Возвращаемся назад и делаем первый ход на $(A3)$. Последовательность нахождения решения для этого случая показана на рисунке 2.113.

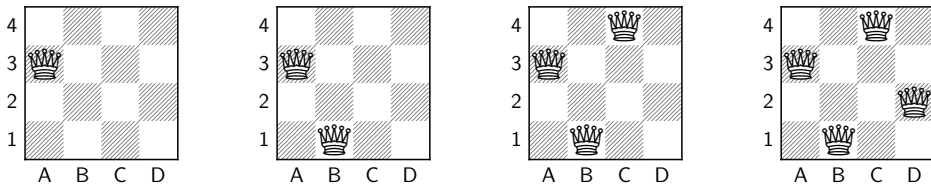


Рис. 2.113.

Поиск решения из начальной позиции $(A4)$ показан на рисунке 2.114. Здесь мы заходим в тупик, решения в этом случае нет.

Весь перебор представлен в таблице 2.30.

На рисунке 2.115 изображено дерево решений, которое строится динамически в процессе перебора с возвратом. Ход решения можно представить как обход этого дерева в глубину.

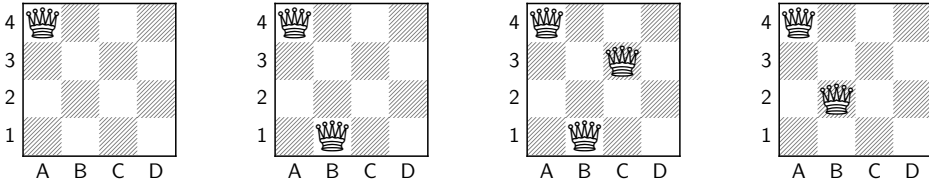


Рис. 2.114.

Таблица 2.30.

(A1)	(A1 → B3)	(A1 → B4)	(A1 → B4 → C2)
(A2)	(A2 → B4)	(A2 → B4 → C1)	(A2 → B4 → C1 → D3)
(A3)	(A3 → B1)	(A3 → B1 → C4)	(A3 → B1 → C4 → D2)
(A4)	(A4 → B1)	(A4 → B1 → C3)	(A4 → B2)

Сформулируем алгоритм в общем виде.

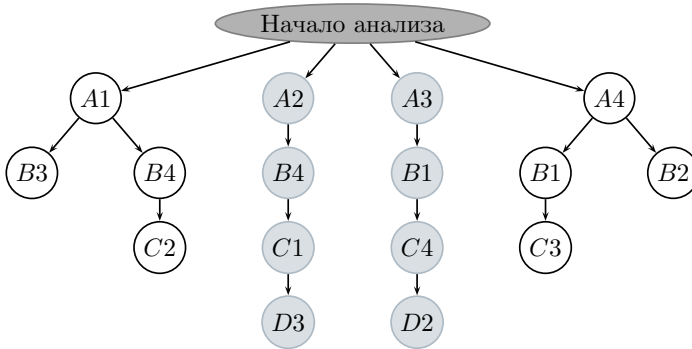


Рис. 2.115.

Пусть множество вариантов, которые надо перебрать, описывается множеством упорядоченных наборов $x = (a_1, a_2, \dots, a_k)$. Каждый следующий элемент набора a_{k+1} определяется выбором из множества $\text{ext}(x)$, которое динамически вычисляется с помощью заданной функции расширения (ext).

Расширение набора будем обозначать операцией конкатенации (приписывание символа к строке):

$$x = (a_1, a_2, \dots, a_k), \quad a_{k+1} \in \text{ext}(x) \Rightarrow x \circ a_{k+1} = (a_1, a_2, \dots, a_k, a_{k+1}).$$

Такие упорядоченные наборы будем называть *частичными решениями* задачи. Для всех частичных решений $x \in X$ определена функция $\text{sol}(x)$, которая устанавливает, является ли данное частичное решение решением

всей задачи. Множество всех решений обозначим P .

Алгоритм 2.27. Рекурсивный алгоритм «перебор с возвратом»

```

 $x \leftarrow \emptyset$  // начальное значение частичного решения — пустой набор
 $P \leftarrow \emptyset$  //  $P$  — множество решений
function backtrack( $x, P$ )
  if sol( $x$ ) then
     $P \leftarrow P \cup \{x\}$  //  $x$  — полное решение, добавляем его в множество решений
  else
    // цикл по всем возможным расширениям частичного решения  $x$ 
    while  $y \in \text{ext}(x)$  do
      backtrack( $x \circ y, P$ ) // расширение частичного решения  $x$ 
    end while
  end if
end function

```

 **Замечание 2.66**

Частичное решение имеет структуру стека: расширение его происходит дописыванием нового элемента справа, а возврат к предыдущим частичным решениям — удалением элементов справа.

При выполнении рекурсивного алгоритма создается стек вызовов, который взаимно однозначно соответствует частичному решению.

Задача 2.7. Используя перебор с возвратом, найдите гамильтонов путь (путь, проходящий через все вершины графа ровно по одному разу, см. определение 2.23) с началом в вершине A для графа на рисунке 2.116.

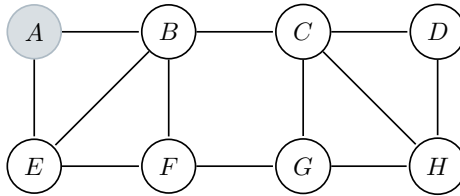



Рис. 2.116.

 **Замечание 2.67**

Алгоритм 2.27 на самом деле описывает группу алгоритмов. Действительно, функция $\text{ext}(x)$, определяющая порядок перебора частичных решений, может быть выбрана по-разному.

Особенно важен правильный выбор функции $\text{ext}(x)$, если нас интересует любое (первое встретившееся) решение. В этом случае полезны различные эмпирические соображения для выбора очередного расширения. В частности, здесь могут использоваться «жадные» алгоритмы.

Пример 2.82. Рассмотрим популярную задачу, связанную с поиском гамильтонова пути в графе. Требуется обойти конем все клетки шахматной доски заданной формы, пройдя каждую ровно по одному разу.

Существует следующий алгоритм для выбора расширения частичного решения: нужно всегда первой идти на ту клетку, из которой существует минимум ходов (стратегия «идти в тупик»). Известно, что для многих конфигураций шахматных досок этот алгоритм дает решение. В общем же случае, очевидно, это оправданный способ упорядочивания частичных решений.

2.2.12. Метод ветвей и границ

Пусть на перебираемом множестве (множестве решений) P задана целевая вещественная функция f . Рассмотрим задачу нахождения элемента множества, для которого значение функции f минимально.

В этом случае объем перебора можно существенно уменьшить, если ввести на множестве частичных решений оценочную функцию $b(x)$.

Функция $b(x)$ подбирается так, чтобы давать нижнюю оценку для значения целевой функции на решении, получающемся последовательным расширением данного частичного решения.

Определение 2.66. Функцию b называют *оценочной* для целевой функции f , если $b(x) \leq f(p)$ для любого решения p и любого частичного решения x , приводящего к p , т. е. если существует последовательность расширений y_1, y_2, \dots, y_k такая что

$$p = x \circ y_1 \circ y_2 \circ \dots \circ y_k, \text{ где } y_i \in \text{ext}(x \circ y_1 \circ y_2 \circ \dots \circ y_{i-1}).$$

Пример 2.83. Рассмотрим задачу нахождения минимального гамильтонова пути в неориентированном графе с неотрицательными весами ребер. Определим целевую функцию f как сумму весов ребер, входящих в гамильтонов путь.

Рассмотрим один из вариантов (не лучший) выбора частичного решения и оценочной функции.

В качестве частичного решения возьмем простой путь в графе, расширения частичного решения — ребра, которыми можно дополнить путь, а в качестве функции b для частичного решения — сумму весов всех ребер этого пути.

Очевидно, функция будет оценочной, так как для построения гамильтонова пути по предложенной схеме на каждом шаге добавляются ребра неотрицательной длины.

Модернизируем алгоритм перебора с возвратом для получения алгоритма, реализующего метод ветвей и границ (*branches and bounds*).

Алгоритм 2.28. Метод ветвей и границ

```

 $x \leftarrow \emptyset$            // начальное значение частичного решения — пустой набор
 $P \leftarrow \emptyset$        //  $P$  — множество решений — пустой набор
//  $r$  — рекорд, минимальное значение целевой функции на найденных решениях
 $r \leftarrow \infty$ 
function bandb( $x, P, r$ )
  if sol( $x$ ) then
    if  $f(x) < r$  then
       $r \leftarrow f(x)$            // сохраняем новый рекорд
       $P \leftarrow \{x\}$          // обновляем множество решений
    else
      if  $f(x) = r$  then
         $P \leftarrow P \cup \{x\}$  // сохранение нового решения
      end if
    end if
  return
else
  if  $b(x) \leq r$  then           // отсечение ветвей дерева перебора
    while  $y \in \text{ext}(x)$  do   // цикл по всем расширениям  $x$ 
      bandb( $x \circ y, P, r$ )
    end while
  end if
end if
end function

```

Пример 2.84. Найдем по предложенной выше схеме кратчайший гамильтонов путь в графе на рисунке 2.117.

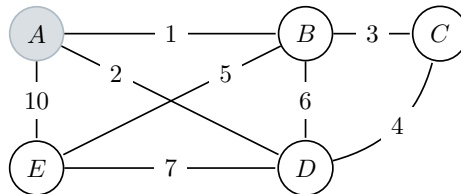


Рис. 2.117.

Заметим, что граф на рисунке 2.117 гамильтонов, так как на нем выполняется условие (2.5) теоремы Оре. Однако предложенный ранее алгоритм 2.9 не обязательно строит кратчайший гамильтонов путь.

Построение начнем с вершины A . Порядок в каждом множестве $\text{ext}(x)$ определим как лексикографический (по вершинам ребер, выходящих из конца построенного пути).

Первым будет построен гамильтонов путь $ABCDE$ (четырьмя последовательными расширениями пустого решения: AB, BC, CD, DE). При этом значение рекорда будет $r = 15$.

Следующие частичные пути будут $ABDC, ABDE$, которые дальше не удастся продолжить, не проходя повторно через некоторые вершины. Затем будет найден второй гамильтонов путь $ABEDC$, который не будет сохранен, так как значение целевой функции для него $f(ABEDC) = 17 > r$.

Далее просмотрим тупиковые пути $ADBC$ и $ADBE$, а затем найдем третий гамильтонов путь $ADCBE$, который имеет меньшую длину, поэтому рекорд поменяется: $r = 14$.

Далее будет опять найден гамильтонов путь с $f(ADEBC) = 17$, который не будет сохранен.

Заметим, что если бы было известно, что в графе нет ребер нулевого веса (или если нас не интересуют все возможные минимальные решения), то в алгоритме нестрогое неравенство $b(x) \leq r$ можно было бы заменить на строгое $b(x) < r$.

В этом случае при рассмотрении частичного решения $ADEB$ значение $b(ADEB) = 14$ и дальнейший перебор не проводится.

Вершины оптимальных гамильтоновых путей отмечены темно-серым цветом, светло-серым — вершины гамильтоновых путей, длина которых хуже рекорда.

Белым цветом (на темном фоне) отмечены вершины тупиковых путей.

Далее будут рассмотрены только частичные решения AEB и AED , которые будут отброшены (и значит, дальнейшее расширение по этим ветвям проводиться не будет) в силу того, что $b(AEB) = 15 > r$ и $b(AED) = 17 > r$.

На рисунке 2.118 показано дерево решений данной задачи методом ветвей и границ, для сравнения штриховыми линиями отмечены ветви, отброшенные оценочной функцией.

Опишем менее очевидный метод построения частичных решений, который, как правило, дает лучшие результаты, чем предыдущий.

В качестве частичного решения рассмотрим теперь не путь, который является частью искомого гамильтонова пути, а набор ребер, которые не входят в него.

В качестве оценочной функции b выберем вес минимального остовного дерева, построенного для графа, из которого уже выброшены ребра частичного решения x (если после отбрасывания граф становится несвязным,

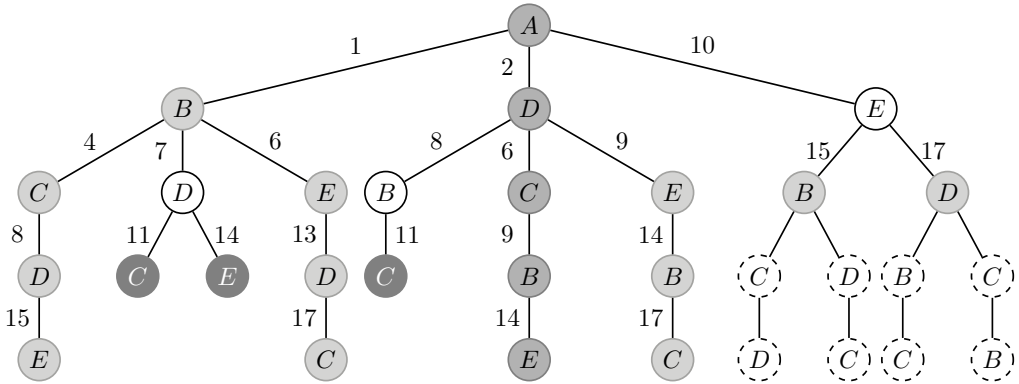


Рис. 2.118.

можно считать вес бесконечным и обрывать расширение данного частичного решения).

Гамильтоновы пути являются подмножеством множества остовных деревьев, а минимум на более широком множестве будет меньше или равен минимуму на более узком. Поэтому $b(x) \leq f(p)$ для любого подмножества ребер исходного графа $E \setminus \{x\}$, из которых можно построить гамильтонов путь.

В качестве расширений частичного решения рассмотрим ребра, выходящие из вершины остовного дерева степени 3 или более (такой вершины не бывает в гамильтоновом пути, поэтому одно из ребер надо удалять; если таких вершин несколько, достаточно взять одну из них).

Задачи и вопросы

1. Нарисуйте двудольный граф по следующим данным. Множество вершин левой доли $V_L = \{6, 7, 8, 9, 10, 15\}$. Множество вершин правой доли $V_R = \{1, 2, 3, 4, 5, 6\}$. Множество ребер определяется отношением делимости числа из левой доли на число из правой. Например, вершина 10 из левой доли соединяется с вершинами 1, 2, 5 правой доли. С помощью алгоритма 2.16 постройте наибольшее паросочетание для данного двудольного графа.

2. Что является инвариантом в алгоритме 2.16 при построении наибольшего паросочетания?

3. Для заданного на рисунке 2.119 графа постройте минимальное остовое дерево, применив:

- алгоритм Прима (построение начинать с вершины K);
- модифицированный алгоритм Краскала.

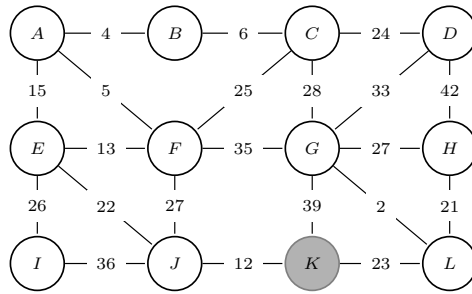


Рис. 2.119.

4. Что сохраняется в качестве инварианта при построении минимального каркаса алгоритмом Прима или алгоритмом Краскала?

5. Модифицируйте алгоритм Прима или алгоритм Краскала для нахождения всех минимальных остовых деревьев графа.

6. На основе алгоритма Форда — Беллмана сформулируйте алгоритм, находящий минимальные длины путей, состоящих ровно из k ребер, от источника до остальных вершин. Докажите корректность алгоритма.

7. В графе с положительными весами ребер известно, что расстояния от источника до остальных вершин различны. Сформулируйте алгоритм, находящий путь от источника до k -й по дальности вершины. Обоснуйте его корректность.

8. Для графа на рисунке 2.120:

а) вычислите длины кратчайших путей от вершины A до остальных вершин с помощью алгоритма Дейкстры;

б) с помощью алгоритма Флойда определите кратчайшие пути между всеми парами вершин. Выпишите кратчайший путь от вершины C до вершины D и его длину;

в) с помощью алгоритма Уоршелла постройте граф и матрицу достижимости.

9. Как в алгоритме Форда — Беллмана можно определить существование в графе цикла отрицательной длины?

10. Какую вычислительную сложность имеет алгоритм Флойда?

11. На основе алгоритма Флойда сформулируйте алгоритм, определяющий наличие циклов отрицательной длины. Докажите корректность алгоритма. Предложите метод нахождения всех отрицательных циклов графа.

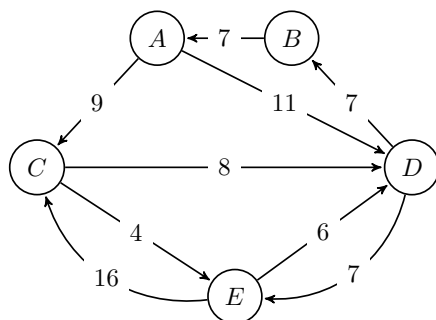


Рис. 2.120.

12. Все вершины графа раскрашены в два цвета (для простоты будем считать, что цвет определяется четностью номера вершины). На основе алгоритма Флойда постройте алгоритм для нахождения длин кратчайших путей, у которых промежуточными могут быть только четные вершины. Докажите корректность алгоритма.

13. Как построить кратчайший путь в графе с «препятствиями»?

14. Используя алгоритм перебора с возвратом, расставьте на шахматной доске восемь ферзей так, чтобы они не били друг друга.

2.3. Бинарные отношения

2.3.1. Введение. Постановка задачи

Бинарные отношения уже встречались в школьном курсе математики. Примеры таких отношений — отношения неравенства, равенства, подобия, параллельности, делимости и пр.

Бинарное отношение каждому двум объектам сопоставляет логическое значение «да», если объекты находятся в этом отношении, и «нет» в ином случае. Другими словами, множество пар объектов разбивается на два подмножества, пары первого подмножества находятся в данном отношении, а второго — не находятся. Это свойство можно положить в основу определения бинарного отношения.

Определение 2.67. Пусть задано множество M . Рассмотрим декартово произведение этого множества на себя $M \times M$. Подмножество R множества $M \times M$ называют *бинарным отношением* R на множестве M .

Если пара $(x; y)$ принадлежит множеству R , то говорят, что элемент x находится в отношении R с элементом y , и записывают xRy .

Пример 2.85. В реляционных базах данных информация хранится в виде отношений: скажем, понятие место жительства рассматривается как отношение между множеством людей и множеством жилых помещений, и т. д.

Такой подход оказался достаточно удобным и универсальным для представления информации различного вида.

Пример 2.86. Введем отношение сравнимости R : x сравнимо с y по модулю m тогда и только тогда, когда x и y имеют одинаковые остатки от деления на m , т. е. $x \equiv y \pmod{m}$.

Рассмотрим введенное отношение R для случая $m = 3$ на множестве $M = \{1; 2; 3; 4; 5; 6\}$, тогда

$$M \times M = \begin{bmatrix} (1; 1); (1; 2); (1; 3); (1; 4); (1; 5); (1; 6); \\ (2; 1); (2; 2); (2; 3); (2; 4); (2; 5); (2; 6); \\ (3; 1); (3; 2); (3; 3); (3; 4); (3; 5); (3; 6); \\ (4; 1); (4; 2); (4; 3); (4; 4); (4; 5); (4; 6); \\ (5; 1); (5; 2); (5; 3); (5; 4); (5; 5); (5; 6); \\ (6; 1); (6; 2); (6; 3); (6; 4); (6; 5); (6; 6) \end{bmatrix}.$$

Отношение R определяется множеством таких пар:

$$R = \left[\begin{array}{ccc} (1; 1); & & (1; 4); \\ & (2; 2); & & (2; 5); \\ & & (3; 3); & & (3; 6); \\ (4; 1); & & & (4; 4); & \\ & (5; 2); & & & (5; 5); \\ & & (6; 3); & & (6; 6) \end{array} \right].$$

Пример 2.87. Рассмотрим в качестве $M = \mathbf{R}$ множество вещественных чисел, или, иными словами, множество точек вещественной прямой. Тогда $M \times M = \mathbf{R}^2$ — множество точек координатной плоскости. Отношение неравенства $<$ определяется множеством пар $R = \{(x; y) | x < y\}$. Это координаты точек, лежащих выше прямой $y = x$.

Упражнение 2.11.

1. На множестве вещественных чисел задано отношение: xRy тогда и только тогда, когда одно из чисел вдвое больше другого. Изобразите на плоскости множество точек, определяющее это отношение.

2. На множестве $M = \{1; 2; 3; 4; 5; 6\}$ задано отношение делимости: xRy тогда и только тогда, когда x делится на y . Сколько пар содержит это отношение? Перечислите эти пары.

3. Введем на множестве $M = \{1; 2; 3; 4; 5; 6\}$ отношение взаимной простоты, т. е. xRy тогда и только тогда, когда x и y взаимно просты: $D(x; y) = 1$. Сколько пар содержит это отношение? Перечислите эти пары.

2.3.2. Свойства бинарных отношений

Определение 2.68. Бинарное отношение R на множестве M называют *рефлексивным*, если каждый элемент этого множества находится в отношении с самим собой: xRx для любого $x \in M$.

Пример 2.88.

1. Отношение сравнимости рефлексивно (при любом натуральном m и на любом множестве целых чисел).

2. Отношение строгого неравенства на множестве вещественных чисел не рефлексивно.

3. Отношение делимости рефлексивно (на любом множестве целых чисел, не содержащем нуля).

Определение 2.69. Бинарное отношение R на множестве M называют *антирефлексивным* (*иррефлексивным*), если ни один элемент этого множества не находится в отношении с самим собой: для любого $x \in M$ неверно, что xRx .

Пример 2.89.

1. Отношение строгого неравенства на множестве вещественных чисел антирефлексивно.

2. Отношение взаимной простоты антирефлексивно на любом множестве целых чисел, не содержащем 1 и -1 , рефлексивно на множествах $\{1\}, \{-1\}, \{-1; 1\}$ и не является ни рефлексивным, ни антирефлексивным в ином случае.

Определение 2.70. Бинарное отношение R на множестве M называют *симметричным*, если вместе с каждой парой $(x; y)$ в отношение входит и симметричная пара $(y; x)$: для любых $x, y \in M$ $xRy \Rightarrow yRx$.

Пример 2.90.

1. Отношение сравнимости симметрично при любом натуральном m и на любом множестве целых чисел.

2. Отношение строгого неравенства на множестве вещественных чисел не симметрично.

3. Отношение делимости симметрично на множестве попарно взаимно простых целых чисел, не содержащем единицу. Например, на множестве простых чисел.

4. Отношение взаимной простоты симметрично на любом множестве целых чисел.

Определение 2.71. Бинарное отношение R на множестве M называют *асимметричным*, если ни одна пара не входит в отношение вместе с симметричной ей: для любых $x, y \in M$, если xRy , то неверно, что yRx .

Пример 2.91.

1. Отношение строгого неравенства на множестве вещественных чисел асимметрично.

2. Отношение делимости не является асимметричным ни на каком множестве целых чисел, не содержащем нуля, вследствие рефлексивности.

Определение 2.72. Бинарное отношение R на множестве M называют *антисимметричным*, если никакая пара, состоящая из разных элементов, не входит в отношение вместе с симметричной ей: для любых $x, y \in M$, если xRy и yRx , то $x = y$.

Пример 2.92.

1. Отношение нестрогого неравенства на множестве вещественных чисел антисимметрично.
2. Отношение делимости является антисимметричным на любом множестве целых чисел.

Упражнение 2.12.

1. Верно ли, что асимметричное отношение всегда антирефлексивно? Докажите.
2. Верно ли, что симметричное отношение всегда рефлексивно? Докажите.
3. Верно ли, что асимметричное отношение всегда антисимметрично? Докажите.
4. Верно ли, что отношение асимметрично тогда и только тогда, когда оно антирефлексивно и антисимметрично? Докажите.

Определение 2.73. Бинарное отношение R на множестве M называют *транзитивным*, если вместе с парами $(x; y)$ и $(y; z)$ в отношении входит и пара (x, z) , т. е. для любых $x, y, z \in M$, если xRy и yRz , то xRz .

Замечание 2.68

Свойство транзитивности хорошо иллюстрируется отношением достижимости: если пункт y достижим из пункта x , а пункт z — из пункта y , то пункт z достижим из пункта x .

Пример 2.93.

1. Отношение сравнимости транзитивно при любом натуральном m и на любом множестве целых чисел.
2. Отношение строгого (нестрогого) неравенства транзитивно на любом подмножестве вещественных чисел.
3. Отношение делимости транзитивно на любом множестве целых чисел.
4. Отношение взаимной простоты не является транзитивным на любом множестве целых чисел. Например, 2 взаимно просто с 3, 3 взаимно просто с 4, но 2 и 4 не взаимно просты.

Упражнение 2.13. Верно ли, что транзитивное и симметричное отношение всегда рефлексивно? Докажите.

2.3.3. Способы задания отношений

Кроме явного перечисления пар, определяющих бинарное отношение, возможны следующие способы задания отношений.

✓ **Задание процедурой проверки.** Рассмотрим этот способ на примере.

Пример 2.94.

1. Отношение взаимной простоты проверяется процедурой нахождения наибольшего общего делителя: если $D(x; y) = 1$, то $(x; y)$ входит в отношение взаимной простоты.

2. Отношение делимости проверяется процедурой деления с остатком: если $x \equiv 0 \pmod{y}$, то $(x; y)$ входит в отношение делимости.

3. Той же процедурой проверяется отношение равенства остатков при делении на m : если $x - y \equiv 0 \pmod{m}$, то $(x; y)$ входит в отношение.

Для отношений на конечных множествах (которые являются основными для дискретной математики) используются также следующие способы задания и описания отношений.

✓ **Задание матрицей смежности.** Определим матрицу A размером $|M| \times |M|$, где $|M|$ — количество элементов множества M . Пронумеруем элементы множества M . Тогда $a_{ij} = 1$, если элемент с номером i находится в отношении с элементом с номером j (iRj), и $a_{ij} = 0$ иначе.

Пример 2.95. Матрица смежности для отношения делимости на множестве $M = \{1; 2; 3; 4; 5; 6\}$ выглядит так:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

✓ **Задание графом.** Элементы множества изображаются точками плоскости и образуют множество вершин графа. Отношение представляют дугами (ребрами) графа: если $(x; y)$ входит в отношение, то из вершины x проводится ориентированная дуга в y .

Пример 2.96. Граф для отношения сравнимости по модулю три на множестве $M = \{1; 2; 3; 4; 5; 6; 7; 8\}$ выглядит, как показано на рисунке 2.121а. Заметим, что он состоит из трех компонент связности: $\{1; 4; 7\}$, $\{3; 6\}$ и $\{2; 5; 8\}$.

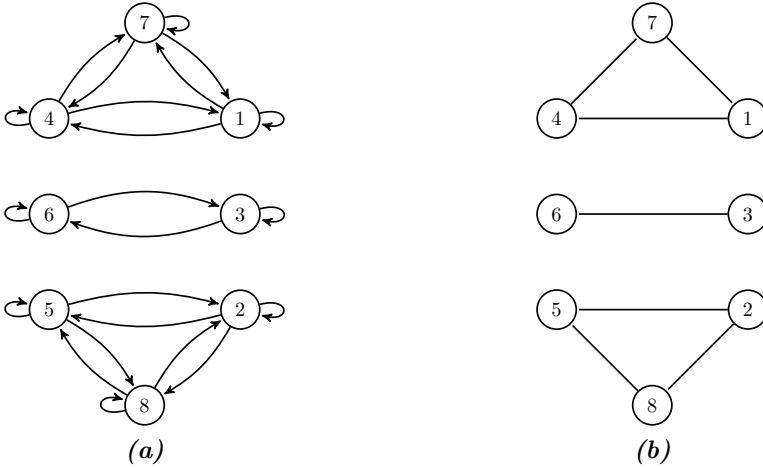


Рис. 2.121.

Замечание 2.69

Для симметричных и рефлексивных отношений петли обычно не изображают, а пары ориентированных дуг, соединяющих данные вершины, заменяют одним неориентированным ребром. Например, граф бинарного отношения из примера 2.96 будет выглядеть так, как показано на рисунке 2.121b.

✓ **Задание списком смежности.** Для каждого элемента множества перечисляются элементы, находящиеся с ним в данном отношении.

Пример 2.97. Список смежности для отношения взаимной простоты на множестве $M = \{1; 2; 3; 4; 5; 6\}$ выглядит так:

$$\begin{array}{lll} 1 \rightarrow 2; 3; 4; 5; 6 & 2 \rightarrow 1; 3; 5 & 3 \rightarrow 1; 2; 4; 5 \\ 4 \rightarrow 1; 3; 5 & 5 \rightarrow 1; 2; 3; 4; 6 & 6 \rightarrow 1; 5 \end{array}$$

Дадим интерпретацию свойств бинарных отношений на описывающих их графах и матрицах.

Теорема 2.55. Справедливы следующие утверждения:

1) диагональ матрицы смежности рефлексивного отношения состоит из единиц;

- 2) у симметричного отношения матрица смежности симметрична;
- 3) граф рефлексивного отношения имеет петли в каждой вершине;
- 4) граф симметричного отношения вместе с дугой, соединяющей x с y , содержит дугу, соединяющую y с x ;
- 5) граф транзитивного отношения обладает следующим свойством: если из вершины x , двигаясь вдоль дуг, можно попасть в вершину y , то в графе должна быть дуга, непосредственно соединяющая x с y .

Упражнение 2.14.

1. Опишите свойства матрицы смежности:

- 1) антирефлексивного отношения;
- 2) асимметричного отношения;
- 3) антисимметричного отношения;
- 4) транзитивного отношения.

2. Опишите свойства графа:

- 1) антирефлексивного отношения;
- 2) асимметричного отношения;
- 3) антисимметричного отношения.

2.3.4. Отношения эквивалентности и толерантности

Определение 2.74. Бинарное отношение, обладающее свойствами рефлексивности, симметричности и транзитивности, называют отношением эквивалентности.

Пример 2.98. Отношение сравнимости (по любому модулю) является отношением эквивалентности.

Сопоставим каждому элементу множества M все элементы, находящиеся с ним в данном отношении эквивалентности: $M_x = \{y \in M \mid xRy\}$. Справедлива следующая теорема.

Теорема 2.56. Множества M_x и M_y либо не пересекаются, либо совпадают.

Доказательство. Все элементы одного класса эквивалентны между собой, т. е. если $x, y \in M_z$, то xRy . Действительно, пусть $x, y \in M_z$, следовательно xRz и yRz . По симметричности отношения R имеем zRy . Тогда в силу транзитивности из xRz и zRy получаем xRy .

Пусть классы M_x и M_y имеют общий элемент z , тогда для любых элементов $x' \in M_x$ и $y' \in M_y$ по доказанному выше $x'Rz$ и zRy' , откуда $x'Ry'$. Таким образом, если классы пересекаются, то они совпадают. ■

Определение 2.75. Множества M_x называют *классами эквивалентности* множества M по отношению R .

Следствие 2.13. Отношение эквивалентности можно интерпретировать как отношение достижимости на неориентированном графе. Тогда классам эквивалентности будут соответствовать компоненты связности графа. Это подграфы, на которые распадается граф: любые вершины из разных компонент связности не достижимы друг из друга, а вершины одной компоненты связности достижимы.

Пример 2.99. Рассмотрим отношение сравнимости целых чисел по модулю m . Классами эквивалентности этого отношения будут классы вычетов по модулю m .

Как построить разбиение конечного множества M на классы эквивалентности для отношения R ? Эту задачу решает следующий алгоритм.

Алгоритм 2.29. Разбиение на классы эквивалентности

```
// Начальная раскраска: цвет элемента  $m_i$  совпадает с его номером
for  $i \leftarrow 1, n$  do
  color( $m_i$ )  $\leftarrow i$ ,  $m_i \in M, i \in 1 : n$ 
end for

// Основной цикл
for  $i \leftarrow 2, n$  do
  for  $j \leftarrow (i - 1), 1$  do
    // если элемент  $m_i$  находится в отношении  $R$  с элементом  $m_j$ ,
    // то «перекрашиваем» его в цвет  $m_j$ 
    if  $m_i R m_j$  then
      color( $m_i$ )  $\leftarrow$  color( $m_j$ )
      break
    end if
  end for
end for
```

После окончания работы алгоритма элементы каждой компоненты связности будут раскрашены в свой цвет.

Пример 2.100. На множестве натуральных чисел

$$M = \{2, 7, 13, 15, 9, 8, 14, 17, 11, 16, 24\}$$

определено бинарное отношение сравнения по модулю 3.

Протокол работы алгоритма представлен в таблице 2.31 (через знак «слеш» обозначен элемент, в цвет которого был перекрашен текущий).

Таблица 2.31.

Исходный цвет	1	2	3	4	5	6	7	8	9	10	11
Элемент	2	7	13	15	9	8	14	17	12	16	24
Новый цвет			2/7		4/5	1/2	1/8	1/14	4/9	2/13	4/12

Элементы раскрашены в три цвета (1, 2, 4).

Отношение эквивалентности является частным случаем более широкого понятия.

Определение 2.76. Отношением *толерантности* (или просто *толерантностью*) на множестве M называется бинарное отношение, удовлетворяющее свойствам *рефлексивности* и *симметричности*, но не обязательно являющееся транзитивным.

В отличие от отношения эквивалентности, дающего разбиение множества M на непересекающиеся подмножества, отношение толерантности дает *покрытие* множества M , т. е. семейство множеств таких, что их объединение содержит множество M .

Пример 2.101. Отношение знакомства между людьми является отношением толерантности. В этом случаях свойство транзитивности не предполагается обязательно быть выполненным. В самом деле, X может быть знаком с Y , Y — с Z , но при этом X и Z могут быть не знакомы друг с другом.

Пример 2.102. Пусть $\{M_i \mid i \in I\}$ — некоторое семейство непустых подмножеств множества M . Тогда отношение пересечения есть отношение толерантности.



Замечание 2.70

На содержательном уровне толерантность означает следующее. Любой объект неразличим сам с собой (свойство рефлексивности), а сходство двух объектов не зависит от того, в каком порядке они сравниваются (свойство симметричности). Однако если один объект сходен с другим, а этот другой — с третьим, то это вовсе не значит, что все три объекта схожи между собой (таким образом, свойство транзитивности может не выполняться).

2.3.5. Отношения предпорядка и порядка

Определение 2.77. Бинарное отношение называют отношением *предпорядка* (*квазипорядка*), если оно рефлексивно и транзитивно.

Отношение предпорядка, обладающее свойством антисимметричности, называют отношением *порядка*.

Замечание 2.71

Иногда определение порядка уточняют так: отношение порядка, определенное выше, называют отношением нестрогого порядка, а асимметричное транзитивное отношение — отношением строгого порядка.

Отношение нестрогого порядка обычно обозначают символом \preceq . Символ \prec используется для обозначения строгого порядка.

Отношение симметричного предпорядка является отношением эквивалентности.

Определение 2.78. Если $x \preceq y$, то говорят, что элемент x *предшествует* элементу y или элемент x *не превосходит* элемента y .

Если $x \prec y$, то говорят, что элемент x *строго предшествует* элементу y или элемент x *меньше* элемента y .

Пример 2.103. Отношение R , заданное на множестве $M = \{a, b, c\}$:

$$R = \{(a, a), (b, b), (c, c), (a, b), (b, a), (b, c), (a, c)\},$$

является предпорядком. Не выполняется свойство антисимметрии (aRb и bRa , но $a \neq b$), однако оно рефлексивно и транзитивно, значит, является предпорядком.

Пример 2.104.

1. На множестве вещественных чисел отношение нестрогого неравенства является отношением (нестрогого) порядка, а отношение строгого неравенства — строгого порядка.

2. Отношение делимости на любом множестве целых чисел, не содержащем нуля, является отношением (нестрогого) порядка.

Определение 2.79. Отношение порядка может обладать или не обладать следующим свойством: для любых $x, y \in M$ либо xRy , либо yRx .

Если отношение порядка обладает этим свойством, то его называют отношением *линейного* порядка, в противном случае — отношением *частичного* порядка.

Пример 2.105.

1. Отношение нестрогого неравенства есть отношение линейного порядка, так как для любых двух вещественных чисел одно будет не меньше другого.

2. Отношение делимости на множестве целых чисел M может являться (в зависимости от множества) отношением частичного порядка или отношением линейного порядка.

Например, если $M = \{1; 2; 3; 4\}$, то это отношение будет отношением частичного порядка, так как ни 3 не делится на 4, ни 4 на 3. Если же $M = \{1; 2; 4; 8\}$, то отношение будет отношением линейного порядка.

Пример 2.106. Пусть множество M состоит из n -мерных векторов, причем для каждого элемента вектора определены отношения $>$ и \geq . Будем считать, что

$$X = (x_1, \dots, x_n) > Y = (y_1, \dots, y_n), \text{ если } x_i \geq y_i, i = 1, \dots, n,$$

и существует индекс $1 \leq k \leq n$ такой, что $x_k > y_k$.

Такое отношение называется *отношением Парето*. Данное отношение транзитивно и асимметрично, то есть это отношение строгого порядка. Отношение используется в многокритериальных задачах принятия решений.

Вектор X называется *Парето-оптимальным решением*, если нет такого вектора Y , что $Y > X$ по Парето.

Рассмотрим следующую задачу. Любая иерархическая система определяет отношение подчинения, которое является отношением порядка, но, как правило, только частичного.

На рисунке 2.122 видно, что, например, для элементов E и C нельзя определить, кто из них «главнее». Это не всегда удобно (например, отношение подчинения желательно определить для любых двух военнослужащих).

Решение этой задачи сводится к построению отношения линейного порядка, согласованного с заданным отношением частичного порядка.

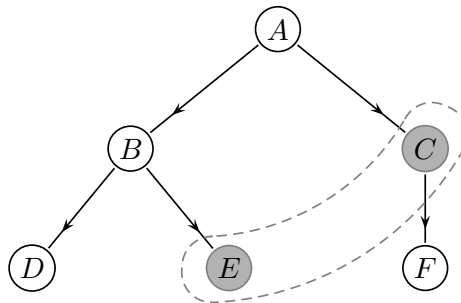


Рис. 2.122.

Определение 2.80. Отношение L называют *согласованным* с отношением R , если любые элементы множества, находящиеся в отношении R , будут находиться и в отношении L .

Определение 2.81. Элемент x множества M называют минимальным по отношению R , если не существует элемента «меньше его», т. е. не существует $y \in M$ такого, что yRx .

Теорема 2.57. Любое отношение порядка (в том числе частичного), заданное на конечном множестве M , имеет минимальный элемент.

Доказательство. Докажем от противного: пусть для каждого элемента M существует меньший его. Тогда получается цепочка (строится справа налево):

$$\dots vRz zRy yRx.$$

Эта цепочка не может состоять из бесконечного количества различных элементов ввиду конечности множества M .

Не умаляя общности, будем считать, что в ней встретится элемент x такой, что цепочка имеет вид

$$\dots xRw \dots vRz zRy yRx.$$

Применим свойство транзитивности к этой цепочке от первого до предпоследнего звена: получим xRy .

Сравним этот результат с последним звеном: для отношения нестрогого порядка отсюда (по антисимметричности) получим $x = y$, что противоречит условию. Для строгого порядка эти отношения (по асимметричности) не могут существовать одновременно, что также приводит к противоречию. ■

2.3.6. Алгоритм топологической сортировки

Алгоритм топологической сортировки по заданному отношению частичного порядка строит согласованное с ним отношение линейного порядка.

В качестве отношения линейного порядка будем рассматривать отношение неравенства на множестве номеров элементов множества M (n — количество элементов M). Процесс нумерации элементов описывается следующим алгоритмом.

Алгоритм 2.30. Топологическая сортировка

```

for  $i \leftarrow 1, n$  do
   $u \leftarrow$  минимальный элемент множества  $M$ 
   $num(u) \leftarrow i$ 
  // Удаляем элемент  $u$  из множества  $M$  вместе с инцидентными ему дугами
   $M \leftarrow M \setminus \{u\}$ 
end for

```

Обоснуем корректность работы алгоритма 2.30.

1. Отношение L , определенное как отношение неравенства на номерах элементов, является отношением линейного порядка.

2. Отношение L согласовано с отношением R . Действительно, выбранный на i -м шаге элемент является минимальным на множестве всех элементов M , за исключением уже выбранных, поэтому он не может быть «больше» (не существует $y \in M$ такого, что yRx) ни одного из них.

По вновь введенному отношению этот элемент будет «больше» всех выбранных и «меньше» всех не выбранных, что подтверждает согласованность.

3. Наконец, алгоритм обязательно (за n шагов) завершит свою работу.

Пример 2.107. Для иерархической системы отношения R , представленного на рисунке 2.123а, можно построить несколько согласованных линейных отношений порядка.

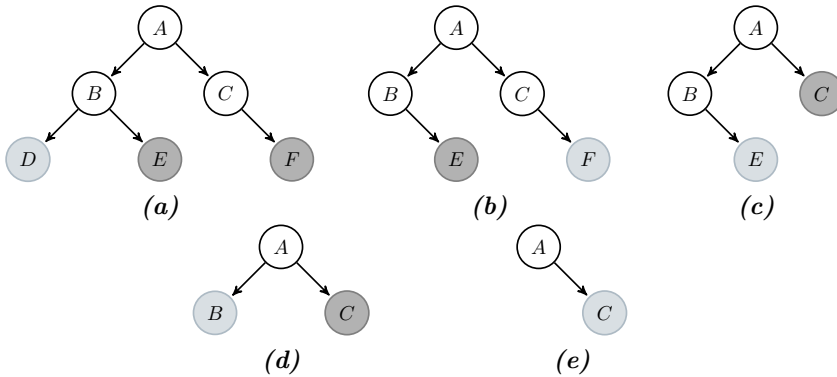


Рис. 2.123.

На рисунках 2.123 выбираемые минимальные элементы показаны светло-серым цветом, возможные кандидаты — темно-серым.

На первом шаге минимальных элементов три — D, E, F (рисунок 2.123а).

Выберем один из них, например D , и присвоим ему номер 1. После удаления этого элемента и всех дуг, входящих в него из графа отношения (рисунок 2.123б), минимальными будут два элемента E и F .

Удалим один из них, например F , и присвоим ему очередной номер — 2 (рисунок 2.123с).

Далее минимальных элементов снова будет два — E и C . Удаляем E и присвоим ему номер 3 (рисунок 2.123д).

Имеем два минимальных элемента — B и C . Удаляем B и присвоим ему номер 4 (рисунок 2.123е).

Удаляем минимальный элемент C и присвоим ему номер 5. Оставшийся элемент A получает номер 6.

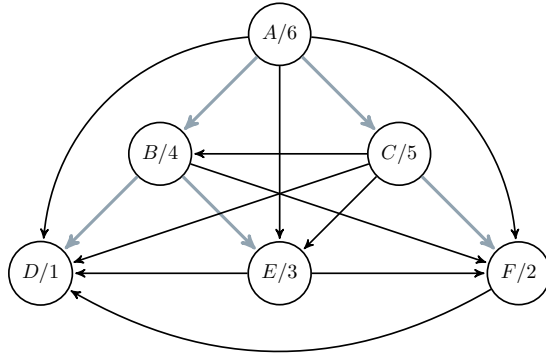


Рис. 2.124.

На рисунке 2.124 исходные связи показаны серым цветом, добавленные для согласованного отношения L связи изображены черным цветом. Протокол работы алгоритма 2.30 представлен в таблице 2.32.

Таблица 2.32.

Вершина	D	F	E	B	C	A
Номер	1	2	3	4	5	6

Замечание 2.72

Как показывает пример 2.107, в зависимости от процедуры выбора минимального элемента могут получиться различные отношения. Но все они будут согласованы с исходным отношением R .

Рассмотрим еще один пример задачи, приводящей к топологической сортировке.

Пример 2.108. Имеется n переменных, значения которых неизвестны. Про некоторые пары переменных известно, что одна переменная в паре меньше другой. Будем считать, что неравенства не противоречат друг другу (переменные находятся в отношении порядка).

Требуется выдать переменные в порядке их возрастания (если решений несколько — выдать любое).

Замечание 2.73

Топологическая сортировка применяется при создании параллельных алгоритмов, когда нужно составить граф зависимостей его операций и, отсортировав его топологически, определить, какие из операций являются независимыми и могут выполняться параллельно (одновременно).

Упражнение 2.15.

1. Дайте определение максимального элемента.
2. Сформулируйте алгоритм топологической сортировки на основе выбора максимального элемента.
3. Примените построенный алгоритм к примеру 2.107.
4. Сколько в этом примере можно построить различных отношений линейного порядка, согласованных с заданным отношением частичного порядка?

Следующая диаграмма (рисунок 2.125) показывает связь между различными классами бинарных отношений.

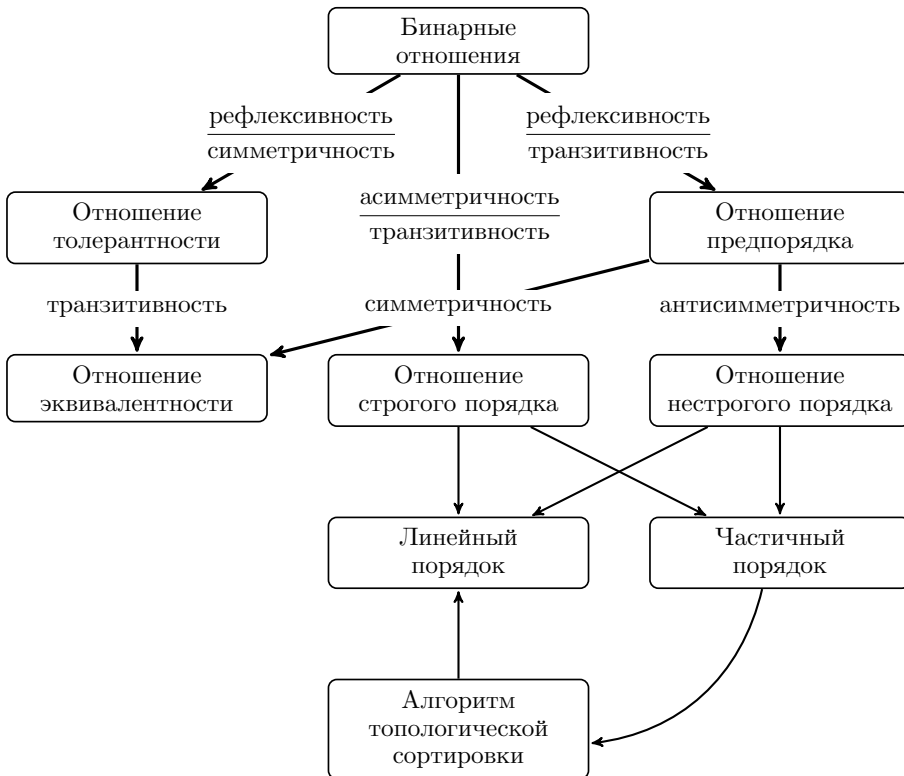


Рис. 2.125.

2.3.7. Частично упорядоченные множества.

Диаграмма Хассе

Определение 2.82. Множество M , на котором задано отношение частичного порядка, называется *частично упорядоченным*. Если на множе-

стве задано отношение линейного порядка, множество называется *линейно упорядоченным*.

Обратимся к графическому представлению упорядоченных множеств. Упорядоченное множество M , имеющее конечное число элементов, удобно представлять с помощью ориентированного графа $H(M)$, который называют *диаграммой Хассе* и строят следующим образом.

Вершины $H(M)$ однозначно соответствуют элементам M . При этом вершину, соответствующую элементу a , называют просто вершиной a . Если элемент b непосредственно следует за элементом a (и только в этом случае), вершину b соединяют дугой с вершиной a , ориентированной от b к a .

Определение 2.83. Говорят, что элемент b непосредственно следует за элементом a , если $a \preceq b (a \prec b)$, и не существует элемента $c \in M$ такого, что $a \preceq c \preceq b (a \prec c \prec b)$.

Историческая справка

Впервые систематически такого рода визуализация описана Биркгофом в 1948 г., название было дано им в честь использовавшего подобные диаграммы немецкого математика Хельмута Хассе (Helmut Hasse).

Обычно в диаграммах Хассе большие элементы помещают над меньшими.

Теорема 2.58. Диаграмма Хассе обладает следующими очевидными свойствами:

- 1) соотношение $a \preceq b (a \prec b)$ эквивалентно существованию ориентированного пути из вершины b в вершину a на диаграмме Хассе;
- 2) если на диаграмме существует ребро e , идущее из вершины b в вершину a , то никакого другого пути из b в a быть не может;
- 3) диаграмма Хассе не имеет циклов.

Пример 2.109. Рассмотрим отношение включения определенное на множестве всех подмножеств четырехэлементного множества $\{a, b, c, d\}$. Это частично упорядоченное множество (строгий порядок). Построим соответствующую диаграмму Хассе (рисунок 2.126).

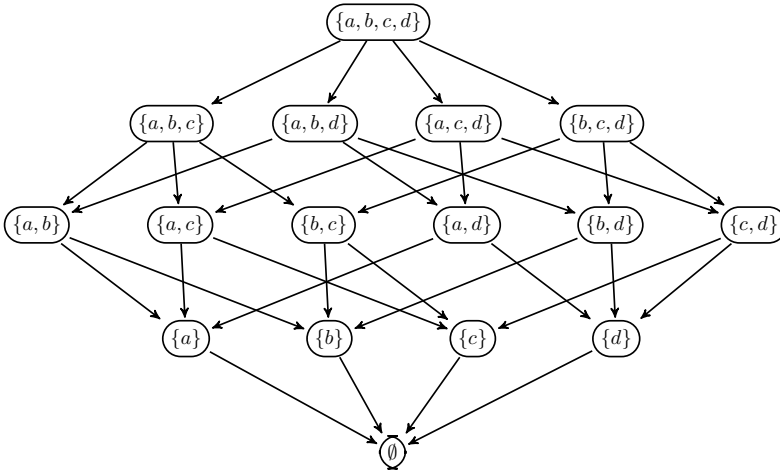


Рис. 2.126.

2.3.8. Транзитивное замыкание бинарного отношения. Алгоритм Уоршелла

Рассмотрим сначала общую постановку задачи.

Пусть R есть некоторое бинарное отношение на множестве M , и это отношение не обладает свойством P . Логично в такой ситуации расширить отношение R до \widehat{R} , чтобы \widehat{R} уже обладало свойством P . Так как R есть подмножество декартового произведения $M \times M$, то под расширением будем понимать добавление в R новых пар элементов множества M .

Ясно, что исходное множество R будет подмножеством в \widehat{R} . В таком случае если вновь построенное множество \widehat{R} будет минимальным среди всех расширений R с выделенным свойством, то говорят, что \widehat{R} является замыканием R относительно данного свойства. Дадим формальное определение.

Определение 2.84. Отношение \widehat{R} называется замыканием отношения R относительно свойства P , если:

- 1) \widehat{R} обладает свойством P ;
- 2) $R \subseteq \widehat{R}$;
- 3) \widehat{R} является подмножеством любого другого отношения, содержащего R и обладающего свойством P .

Пример 2.110. Пусть $R = \{(a, b) | a, b \in N, a > b\}$. Симметрично ли это отношение? Ответ отрицательный.

Построим замыкание этого отношения относительно свойства симметричности. Это означает, что если пара (a, b) принадлежит R , то пара (a, b)

также будет принадлежать \widehat{R} . Кроме того, пара (b, a) также будет принадлежать \widehat{R} . То есть в нашем случае

$$\begin{aligned}\widehat{R} &= \{(a, b) | a, b \in N, a > b\} \cup \{(a, b) | a, b \in N, a < b\} = \\ &= \{(a, b) | a, b \in N, a \neq b\}.\end{aligned}$$

Вывод: симметричным замыканием отношения «меньше» на множестве натуральных чисел является отношение «не равно».

Перейдем к рассмотрению важного частного случая.

Рассмотрим следующую задачу. Пусть на конечном множестве M задано бинарное отношение R , не являющееся, вообще говоря, транзитивным. Как построить согласованное с ним транзитивное бинарное отношение?

Замечание 2.74

1) Эта задача имеет содержательную интерпретацию на графе: надо дополнить граф следующими дугами. Если из вершины x можно построить путь до вершины y , то надо построить дугу $[x, y]$.

2) На матрице смежности задачу можно сформулировать так: по матрице смежности построить матрицу достижимости.

Для унификации с рассмотренным ранее алгоритмом Флойда матрицу достижимости будем обозначать как d :

$$d(u, v) = \begin{cases} 1, & \text{если есть путь из вершины } u \text{ в вершину } v, \\ 0, & \text{в противном случае.} \end{cases} \quad (2.34)$$

В теореме 2.59 будет получен алгоритм построения матрицы достижимости (транзитивного замыкания) с помощью возведения матрицы смежности в степень.

Заменим обычные арифметические операции, участвующие в определении умножения матриц, следующими: операцию сложения двух чисел заменим операцией выбора наибольшего значения (а операцию умножения сохраним прежней). Тогда верна теорема.

Теорема 2.59. Пусть A — матрица смежности рефлексивного отношения R , заданного на конечном множестве M , а n — число элементов M . Тогда A^{n-1} — матрица достижимости.

Доказательство. Рассмотрим умножение матрицы смежности A на себя.

Элемент, стоящий в i -й строке и j -м столбце результирующей матрицы, по определению умножения матриц (со сделанными уточнениями) покажет,

можно ли достичь из i -й вершины j -ю, используя не более одной промежуточной вершины.

Соответственно элементы $(n-1)$ -й степени матрицы A покажут попарные достижимости вершин с использованием не более $(n-2)$ промежуточных вершин. Но на графе из n вершин любой путь без циклов будет иметь максимум n вершин, а значит, не более $(n-2)$ промежуточных. ■

Пример 2.111. Рассмотрим граф рефлексивного отношения на множестве из трех элементов. Пусть A — матрица смежностей этого отношения, $n=3$.

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \text{ тогда после работы алгоритма } A^{n-1} = A^2 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

На рисунке 2.127а представлен исходный граф, на рисунке 2.127б — граф транзитивного замыкания. Дуги исходного отношения серого цвета, добавленные дуги транзитивного замыкания — черного.

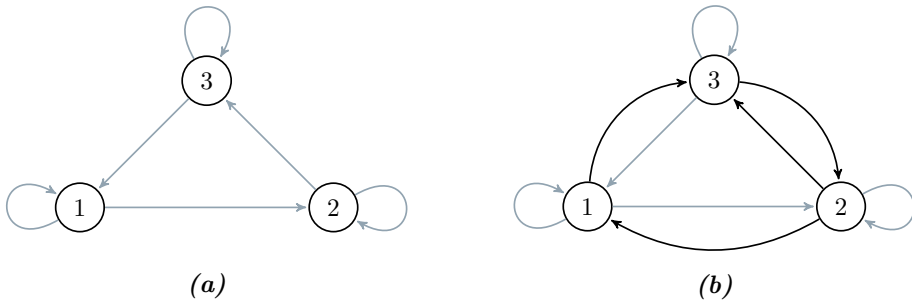


Рис. 2.127.

Замечание 2.75

Граф, соответствующий транзитивному замыканию, является полным трехвершинным графом — все его вершины соединены дугами.

Упражнение 2.16. Примените алгоритм умножения матриц к антирефлексивному отношению. Какой смысл будет иметь матрица A^{n-1} ?

Оценим трудоемкость предложенного алгоритма.

При вычислении одного элемента матрицы, полученной умножением матриц размером $n \times n$ на себя, требуется n умножений и $(n-1)$ сложение. Для вычисления всех элементов потребуется порядка n^3 действий, а для возведения матрицы в $(n-2)$ степень — порядка n^4 действий.

Возникает вопрос: нет ли алгоритма с меньшим порядком трудоемкости? Таким алгоритмом является *алгоритм Уоршелла*, трудоемкость которого имеет порядок n^3 .

Алгоритм Уоршелла. Идея алгоритма Уоршелла состоит в расширении множества промежуточных вершин по следующему правилу: на каждом шаге в рассмотрение добавляется одна новая вершина, после чего достижимости вершин пересчитываются «через нее».

Если w — промежуточная вершина, то достижимость вершины v из вершины u через w (с учетом (2.34)) пересчитывается по правилу

$$d(u, v) := \max\{d(u, v), d(u, w) \cdot d(w, v)\},$$

т. е. не меняется, если v достижима из u , и меняется (с 0 на 1), если достижимости до введения промежуточной вершины w не было, а w достижима из u и v достижима из w .

Таким образом, после k шагов будут соединены те вершины, которые достижимы по путям, проходящим только через первые k вершин (кроме первой и последней).

Замечание 2.76

Так как элементы матрицы смежности по существу логические значения (0, 1), правило модификации матрицы d естественнее записать в логических операциях:

$$d(u, v) := d(u, v) \vee (d(u, w) \wedge d(w, v)).$$

Пусть отношение R задано на множестве M . Тогда алгоритм Уоршелла можно записать следующим образом.

Алгоритм 2.31. Алгоритм Уоршелла

// Инициализация матрицы достижимости

for each $u \in M$ **do**

for each $v \in M$ **do**

$d(u, v) \leftarrow uRv$

end for

end for

// Основной алгоритм

for each $w \in M$ **do**

for each $u \in M$ **do**

for each $v \in M$ **do**

```

       $d(u, v) \leftarrow d(u, v) \vee (d(u, w) \wedge d(w, v))$ 
    end for
  end for
end for

```

Легко видеть, что трудоемкость полученного алгоритма n^3 .

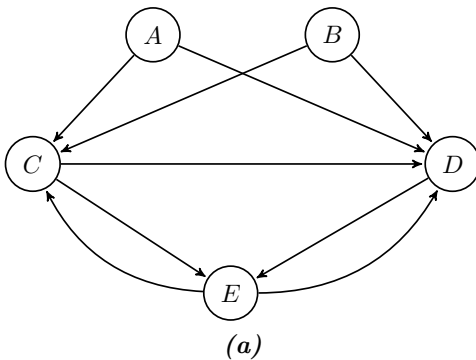
Построение алгоритма Уоршелла осуществлялось на основе теоретических рассуждений, которые фактически составляют доказательство корректности полученного алгоритма. Заметим, что после выполнения k -го шага внешнего цикла значения $d(u, v)$ показывают достижимость вершины v из вершины u , используя первые k вершин в качестве промежуточных.

Оба алгоритма построения транзитивного отношения, согласованного с данным, строят одно и то же отношение. Оно называется транзитивным замыканием исходного отношения, так как является минимальным (остальные согласованные с исходным транзитивные отношения его содержат) среди всех таких расширений исходного отношения.

Алгоритм Уоршелла является частным случаем алгоритма Флойда (для графов с единичными весами).

Обычно в алгоритме Уоршелла петли не рассматриваются: значения матрицы достижимости на диагонали не модифицируются.

Пример 2.112. С помощью алгоритма Уоршелла построим матрицу достижимости для графа на рисунке 2.128а. Матрица смежности графа представлена на рисунке 2.128б.



	A	B	C	D	E
A	0	0	1	1	0
B	0	0	1	1	0
C	0	0	0	1	1
D	0	0	0	0	1
E	0	0	1	1	0

(b)

Рис. 2.128.

Проиллюстрируем работу алгоритма Уоршелла по шагам. В качестве номера шага будет фигурировать соответствующая вершина внешнего цикла. Матрица достижимости d в начале работы совпадает с матрицей смежности.

Замечание 2.77

Как и в алгоритме Флойда, при работе с промежуточной вершиной u не модифицируются элементы строки и столбца u матрицы достижимости.

А Пересчет матрицы достижимости идет через вершину A :

$$d(u, v) := \max\{d(u, v), (d(u, A) \wedge d(A, v))\}.$$

В силу того, что $d(u, A) = 0$ для всех вершин, на этом шаге матрица достижимости не меняется.

В Пересчет матрицы достижимости идет через вершину B :

$$d(u, v) := \max\{d(u, v), (d(u, B) \wedge d(B, v))\}.$$

Так как $d(u, B) = 0$ для всех вершин, то на этом шаге матрица достижимости также не меняется.

С Пересчет матрицы достижимости идет через вершину C :

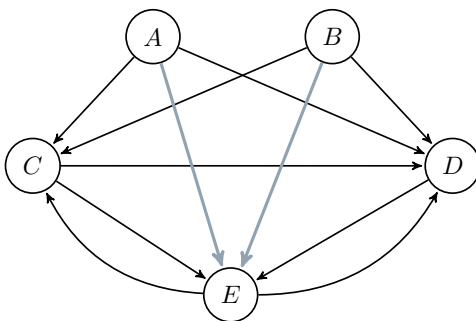
$$d(u, v) := \max\{d(u, v), (d(u, C) \wedge d(C, v))\}.$$

Матрица d модифицируется следующим образом:

$$d(A, E) := d(A, C) \wedge d(C, E) = 1,$$

$$d(B, E) := d(B, C) \wedge d(C, E) = 1.$$

Граф и матрица достижимости после шага C показаны соответственно на рисунке 2.129а и б. Добавленные дуги отмечены серым цветом.



(a)

	A	B	C	D	E
A	0	0	1	1	1
B	0	0	1	1	1
C	0	0	0	1	1
D	0	0	0	0	1
E	0	0	1	1	0

(b)

Рис. 2.129.

D Пересчет матрицы достижимости идет через вершину D :

$$d(u, v) := \max\{d(u, v), (d(u, D) \wedge d(D, v))\}.$$

На этом шаге матрица достижимости не меняется.

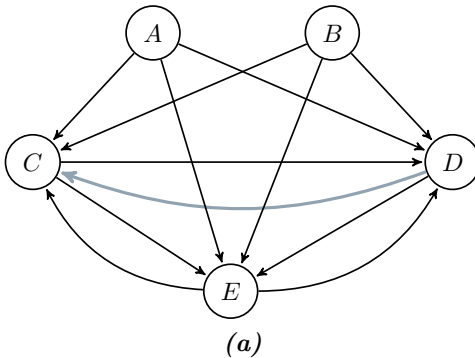
E Пересчет матрицы достижимости идет через вершину E :

$$d(u, v) := \max\{d(u, v), (d(u, E) \wedge d(E, v))\}.$$

Матрица d модифицируется следующим образом:

$$d(D, C) := d(D, E) \wedge d(E, C) = 1.$$

Граф и матрица достижимости представлены соответственно на рисунке 2.130а и б. Добавленная дуга отмечена серым цветом.



	A	B	C	D	E
A	0	0	1	1	1
B	0	0	1	1	1
C	0	0	0	1	1
D	0	0	1	0	1
E	0	0	1	1	0

(b)

Рис. 2.130.

2.3.9. Примеры на бинарные отношения

Пример 2.113. Пусть $M = \{1, 2, 3, 4, 5\}$ и

$$R = \{(x, y) \mid x, y \in M, x + 1 < y\}.$$

Очевидно, что отношение R антирефлексивно, асимметрично и транзитивно.

Пример 2.114. Отношение на множестве слов русского языка определено следующим образом: aRb тогда и только тогда, когда они имеют хотя бы одну общую букву. Исследовать отношение на множестве

$$M = \{\text{лошадь, корова, вагон, нить, пир, липа}\}.$$

Отношение R рефлексивно, симметрично, но не транзитивно: лошадь R корова, корова R пир, но неверно, что лошадь R пир.

Таким образом, отношение R — отношение толерантности. С помощью алгоритма Уоршелла можно построить транзитивное замыкание отношения.

Пример 2.115. Рассмотрим отношение включения множеств $A \subseteq B$, определенное на множестве всех подмножеств четырехэлементного множества $\{a, b, c, d\}$.

Отношение рефлексивно, антисимметрично и транзитивно. Это отношение порядка, но частичного, так как, например, $\{a, b\} \not\subseteq \{a, c\}$. С помощью алгоритма топологической сортировки легко вложить его в полное отношение.

Пример 2.116. Пусть $M = \{2, 3, 4, 5, 6, 7, 8, 9\}$ и

$$R = \{(x, y) \mid x, y \in M, D(x, y) = 1\}.$$

Очевидно, что отношение R на данном множестве M антирефлексивно, симметрично, но не транзитивно: $D(3, 5) = 1, D(5, 9) = 1$, но $D(3, 9) \neq 1$.

Пример 2.117. Пусть $A = \{1, 2, 3\}$, а отношение R на A задано упорядоченными парами:

$$R = \{(1, 1), (1, 2), (1, 3), (3, 1), (2, 3)\}.$$

Оно не рефлексивно, не симметрично и не транзитивно. Построим соответствующие замыкания.

1. Замыкание по рефлексивности: $\hat{R} = R \cup \{(1, 1), (2, 2), (3, 3)\}$.
2. Замыкание по симметричности: $\hat{R} = R \cup \{(2, 1), (3, 2)\}$.
3. Замыкание по транзитивности можно построить с помощью алгоритма Уоршелла.

2.3.10. Введение в семантические сети

Историческая справка

Идея семантических сетей была предложена во второй половине 1990-х гг. Тимом Бернерсом-Ли — одним из авторов проекта «Всемирная паутина» (World Wide Web).

Определение 2.85. *Семантическая сеть* — информационная модель предметной области, имеющая вид ориентированного графа, узлы которого соответствуют объектам предметной области, а дуги задают отношения между ними.

Объектами предметной области могут быть понятия, события, свойства, процессы. Таким образом, семантическая сеть является одним из способов представления знаний.

Если тип отношений в сети один, такие сети называют *однородными*, если количество типов отношений больше одного — *неоднородными*.

Наиболее часто встречаемыми сетями являются сети с бинарными отношениями. Однако на практике используются сети и с n -арными отношениями ($n > 2$). В этом случае используется другой подход — *концептуальные графы*.



Замечание 2.78

Рассмотренные ранее бинарные отношения являются однородными бинарными семантическими сетями.

Пример 2.118. На рисунке 2.131 показан пример неоднородной бинарной семантической сети.

Здесь присутствует несколько типов бинарных отношений: «получает», «учится в», «это», «сдает экзамен», «работает в».

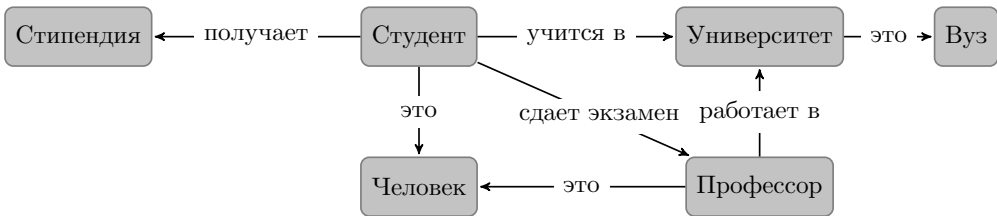


Рис. 2.131.

Задачи и вопросы

1. Построить бинарное отношение R на множестве

$$A = \{1, 2, 3\}, R = \{(1, 1), (2, 2), (3, 3), (1, 2), (2, 1), (2, 3), (3, 2)\} :$$

- рефлексивное, симметричное, нетранзитивное;
- рефлексивное, транзитивное, несимметричное;
- симметричное, транзитивное, нерефлексивное.

2. Показать, что отношение R , определенное на множестве A :

$$A = \{1, 2, 3, 4, 5, 6\}$$

$$R = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (1, 2), (1, 4), (2, 1), (2, 4), (3, 5), (5, 3), (4, 1), (4, 2)\},$$

есть отношение эквивалентности. Построить классы эквивалентности.

3. Исследовать отношение R , заданное на множестве M :

$$M = \{-4, -3, -2, -1, 0, 1, 2, 3\}, \quad R = \{(x, y) \mid x, y \in M, xy > 0\}.$$

4. Граф транзитивного бинарного отношения содержит цикл длиной

4. Может ли отношение быть антирефлексивным?

5. Определим бинарное отношение на множестве статей P следующим образом: пара статей α и β принадлежат P тогда и только тогда, когда они имеют общую ссылку на одну и ту же статью. Является ли это отношение отношением эквивалентности?

6. Является ли отношение перпендикулярности прямых отношением эквивалентности?

7. Отношение M на множестве точек плоскости: $(P_1, P_2) \in M$, если ординаты точек P_1 и P_2 равны. Является ли отношение M отношением эквивалентности?

8. Может ли отношение эквивалентности быть одновременно и отношением порядка?

9. Какими свойствами обладает отношение

$$xRy \leftrightarrow |x - y| = 1, \quad x, y \in \{1, 2, 3, 4, 5\}.$$

10. Отношение R на некотором множестве слов определено следующим образом: пара слов W_1 и W_2 принадлежит M тогда и только тогда, когда они начинаются с одного и того же символа. Является ли R отношением эквивалентности? Отношением толерантности?

11. Отношение R на некотором множестве слов определено следующим образом: пара слов W_1 и W_2 принадлежит M тогда и только тогда, когда они отличаются не более чем на одну букву. Является ли R отношением эквивалентности? Отношением толерантности?

12. Нарисуйте граф с пятью вершинами, соответствующий рефлексивному симметричному и нетранзитивному отношению.

13. Как построить классы эквивалентности для отношения эквивалентности?

14. Как сделать отношение частичного порядка полным?

15. Дано асимметричное отношение. Справедливо ли утверждение: после транзитивного замыкания отношение стало:

а) симметричным;

б) рефлексивным.

16. Бинарное отношение на конечном множестве является отношением строгого порядка. Граф отношения имеет 6 ребер. Сколько элементов содержит множество?

17. Сколько единиц в матрице отношения строгого порядка, заданного на множестве из 5 элементов?

18. Дано множество из трех элементов. Сколько различных:

а) антисимметричных;

б) антирефлексивных;

в) транзитивных

отношений можно задать на этом множестве.

19. Является ли отношение, заданное на множестве $M = \{A, B, C\}$ (рисунок 2.132), транзитивным?

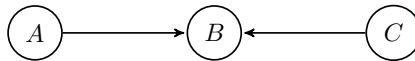


Рис. 2.132.

МАТЕМАТИЧЕСКАЯ ЛОГИКА И ТЕОРИЯ АЛГОРИТМОВ

3.1. Логика высказываний

Историческая справка

Формальную систему логики впервые разработал великий греческий ученый Аристотель (ученик Платона, воспитатель Александра Македонского). В своем логическом своде «Органон» он создал раздел формальной логики — силлогистику. Его труды оказали влияние на развитие логической науки во всем мире. В Европе до XVII в. вся логика развивалась на основе учения Аристотеля.

Первые значительные попытки превращения логики в математическую науку сделал великий немецкий ученый и политический деятель Готфрид Вильгельм Лейбниц (1646–1716). Однако решающего успеха в этом направлении добился английский математик Джордж Буль (1815–1864) в 1847 г., построив алгебру логики, названную в его честь булевой.

Современный вид математическая логика приобрела в 1980-е гг. в трудах немецкого логика, математика и философа Готлоба Фреге (1848–1925). Он привел первую аксиоматику логики высказываний и предикатов и сделал попытку свести математику к логике.

3.1.1. Основные понятия логики высказываний

Логика высказываний рассматривает предложения языка не с точки зрения их смысла, содержания, а только с точки зрения их истинности или ложности.

Высказыванием будем называть повествовательное предложение, которое является или может оказаться либо истинным, либо ложным. Причем оно не может быть истинным и ложным одновременно. В дальнейшем истину будем обозначать цифрой 1, а ложь — цифрой 0.

Пример 3.1 (задача Кислера [20]). Браун, Джонсон и Смит обвиняются в подделке сведений о доходах. Они дают под присягой следующие показания:

Браун: Джонсон виновен, а Смит не виновен.

Джонсон: Если Браун виновен, то виновен и Смит.

Смит: Я не виновен, но хотя бы один из них виновен.

Требуется ответить на вопросы:

1. Совместимы ли показания всех троих?
2. Если все трое не виновны, то кто лжесвидетельствовал?
3. Предполагая, что все показания истинны, найти виновного.

Введем следующие обозначения: B — Браун виновен, J — Джонсон виновен, S — Смит виновен.

Проанализируем показания обвиняемых с точки зрения их «внутреннего строения». Высказывания B, J, S можно назвать простыми, а, например, высказывание Джонсона (если B , то S) — составным, полученным из простых высказываний B и S . Этот пример показывает способ построения составных (сложных) высказываний из простых. Эти способы будем называть *операциями*. В качестве основных выделим пять операций.

Определение 3.1. Пусть X и Y — некоторые высказывания. Тогда высказывание:

- 1) « X и Y » — *конъюнкция* высказываний X и Y ;
- 2) « X или Y » — *дизъюнкция* высказываний X и Y ;
- 3) «не X » — *отрицание* высказывания X ;
- 4) «если X , то Y » — *импликация* высказываний X и Y ;
- 5) « X тогда и только тогда, когда Y » — *эквиваленция* высказываний X и Y .

Например, высказывание Джонсона из примера 3.1 является импликацией высказываний B и S .

Введем следующие обозначения для операций: \wedge — конъюнкция; \vee — дизъюнкция; \neg — отрицание; \rightarrow — импликация; \leftrightarrow — эквиваленция. Символы $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$ называют *связками*.

Следующая таблица определяет, какие значения принимают высказывания, полученные с помощью этих операций, если исходные высказывания X и Y принимают значения 0 или 1. Таковую таблицу называют *таблицей истинности* (таблица 3.1).

Таблица 3.1.

X	Y	$X \wedge Y$	$X \vee Y$	$\neg X$	$X \rightarrow Y$	$X \leftrightarrow Y$
0	0	0	0	1	1	1
0	1	0	1	1	1	0
1	0	0	1	0	0	0
1	1	1	1	0	1	1

Более точно таблица 3.1 содержит пять таблиц истинности — по одной для каждой связи, которые для удобства объединены в одну.

Для операций \vee и \rightarrow введенные значения требуют дополнительных пояснений.

В русском языке союз «или» имеет два значения: *разделительное* — или то, или другое, но не оба например, («Пан или пропал», что равносильно «Или пан, или пропал»); *соединительное* — или то, или другое, или оба вместе. Как видно из таблицы 3.1, союз «или» будем понимать в соединительном смысле.

Таблица истинности для импликации построена на основе принципов логики Аристотеля:

- из лжи следует все, что угодно (см. первую и вторую строки таблицы 3.1);
- истина не нуждается в обосновании (вторая и четвертая строки таблицы 3.1).

Пример 3.2. Если n кратно 6, то n кратно 3. Эта импликация истинна при любом натуральном n .

Например, при $n = 6$ это определяет четвертая строка таблицы 3.1, при $n = 3$ — вторая строка, а при $n = 5$ — первая.

С помощью логических операций можно строить не только конкретные высказывания, но и выражения, которые имеют только форму высказывания, а конкретное содержание в них можно вкладывать любое. Такие выражения называют *пропозициональными (высказывательными) формулами*. Дадим точные определения.

Определение 3.2. Символы истины 1 и лжи 0 называют *логическими константами* (или в дальнейшем просто константами).

*Логическими переменными*¹ (в дальнейшем просто переменными) называют буквы X, Y, Z, \dots с индексами и без них, значениями которых могут быть 1, или 0.

Определение 3.3. *Формулами* логики высказываний называют:

- 1) логические переменные и константы;
- 2) выражения вида $(F) \wedge (G)$, $(F) \vee (G)$, $\neg(F)$, $(F) \rightarrow (G)$, $(F) \leftrightarrow (G)$, где F и G — формулы логики высказываний.

Формулу F , в которую входят переменные X_1, \dots, X_n , будем обозначать $F(X_1, \dots, X_n)$. Более того, будем пользоваться последним обозначением, даже если некоторые из переменных отсутствуют в записи формулы F , т. е. являются *фиктивными* переменными.

Замечание 3.1

Определение 3.3 относится к так называемым индуктивным^a определениям. Такие определения вводят сначала базовые объекты (в нашем случае это введенные в первом пункте логические переменные и константы), а затем способы конструирования новых объектов из уже полученных (в нашем случае это заданные во втором пункте возможности применения логических операций).

^a От лат. *inductio* — выведение, наведение.

Пример 3.3. Рассмотрим выражение $((X) \wedge (Y)) \rightarrow (\neg(Z))$. Является ли данное выражение формулой?

Буквы X, Y, Z — переменные в силу первого пункта определения 3.3. Применяя второй пункт, получаем, что выражение $(X) \wedge (Y)$ — формула, $(\neg(Z))$ — формула, следовательно, $((X) \wedge (Y)) \rightarrow (\neg(Z))$ также формула.

Если строго следовать определению 3.3, то в формуле надо писать много скобок. Это неудобно для восприятия формулы. Чтобы уменьшить количество скобок, условимся переменные и константы в скобки не заключать и введем приоритет (силу связывания) для связок.

Будем считать, что \neg имеет наивысший приоритет, затем в порядке уменьшения приоритета следуют связки $\wedge, \vee, \rightarrow$ и самый маленький приоритет имеет связка \leftrightarrow . Связки одного старшинства применяются в порядке их следования слева направо.

Пример 3.4. Используя эти соглашения, формулу из примера 3.3 $((X) \wedge (Y)) \rightarrow (\neg(Z))$ можно записать в виде $X \wedge Y \rightarrow \neg Z$.

¹ Другие названия этого понятия: пропозициональные переменные, атомарные формулы или атомы логики высказываний.

Для дальнейшего упрощения записи формул введем следующие соглашения.

1. Будем опускать символ конъюнкции \wedge , т. е. вместо $X \wedge Y$ писать просто XY , если его присутствие будет очевидно из контекста.

2. Вместо символа отрицания $\neg X$ будем писать \bar{X} , например, $X \bar{Y} Z \vee X \bar{Z}$ вместо $X \wedge \neg Y \wedge Z \vee X \wedge \neg Z$.

Пример 3.5. Вернемся к задаче Кислера (см. пример 3.1). Запишем показания свидетелей в виде формул. Имеем

$$\text{Браун: } \bar{J}\bar{S}, \quad \text{Джонсон: } B \rightarrow S, \quad \text{Смит: } \bar{S}(B \vee J). \quad (3.1)$$

Задача 3.1. Разработайте алгоритм синтаксического анализа текста в алфавите исходных символов, который решает следующую задачу: по тексту дает ответ на вопрос: является текст формулой логики высказываний или нет.

Можно ли решить эту задачу за один просмотр текста?

В дальнейшем нам понадобится понятие подформулы. Дадим индуктивное определение, аналогичное определению 3.3 для формул.

Определение 3.4.

- 1) Подформулой переменной или константы является она сама.
- 2) Подформулами формулы \bar{F} являются она сама и все подформулы формулы F .
- 3) Подформулами формул FG , $F \vee G$, $F \rightarrow G$, $F \leftrightarrow G$ являются они сами и все подформулы формул F и G .

Пример 3.6. Формула $F = XY \rightarrow X \vee Z$ имеет шесть подформул: X , Y , Z , XY , $X \vee Z$, $XY \rightarrow X \vee Z$.

Определение 3.3 позволяет определять соответствие формулы синтаксису логики высказываний (проводить синтаксический анализ). Введем понятие значения (смысла) формулы, т. е. семантику логики высказываний.

Рассмотрим формулу $F(X_1, \dots, X_n)$. Если задать значения всех входящих в формулу переменных:

$$X_1 = a_1, \dots, X_n = a_n, \quad a_i \in \{0, 1\}, i = 1, 2, \dots, n, \quad (3.2)$$

то, используя таблицу истинности, можно вычислить значение всей формулы на этом наборе:

$$F(a_1, \dots, a_n) = b, \quad b \in \{0, 1\}.$$

В этом случае говорят, что задана *интерпретация формулы F* на наборе (3.2). Такие наборы из нулей и единиц называют *булевыми наборами*.

Таким образом, чтобы определить значение формулы, нужно задать ее интерпретацию на всех возможных булевых наборах значений входящих в нее переменных.

Очевидна следующая теорема.

Теорема 3.1. Число различных булевых наборов формулы от n переменных равно 2^n .

Доказательство. Каждый набор $a = (a_1, \dots, a_n)$ можно интерпретировать как двоичную запись некоторого натурального числа:

$$N(a) = (a_1, \dots, a_n)_2 = a_n + a_{n-1}2 + \dots + a_k 2^{n-k} + \dots + a_1 2^{n-1}. \quad (3.3)$$

Число $N(a)$ будем называть *номером набора*. Согласно результатам примера 1.77 (см. раздел «Теория множеств и комбинаторика») число таких двоичных чисел равно 2^n . ■

Условимся в дальнейшем упорядочивать булевы наборы в лексикографическом порядке, т. е. в порядке возрастания номеров наборов. Таким образом семантическое значение формулы логики высказываний $F(X_1, \dots, X_n)$ можно задать таблицей 3.2.

Таблица 3.2.

X_1	X_2	...	X_{n-1}	X_n	$F(X_1, X_2, \dots, X_{n-1}, X_n)$
0	0	...	0	0	$F(0, 0, \dots, 0, 0)$
0	0	...	0	1	$F(0, 0, \dots, 0, 1)$
...
1	1	...	1	0	$F(1, 1, \dots, 1, 0)$
1	1	...	1	1	$F(1, 1, \dots, 1, 1)$

Эту таблицу называют *таблицей истинности* для формулы F . Каждая строка таблицы 3.2 соответствует интерпретации формулы F на данном булевом наборе.

Пример 3.7. Ответим на вопросы, поставленные в задаче Кислера (см. пример 3.1). Согласно (3.1) составим совместную таблицу истинности для всех свидетельских показаний (таблица 3.3).

Совместны ли показания? Этот вопрос уточняется так: можно ли придать переменным B, J, S такие логические значения, чтобы формулы (3.1)

Таблица 3.3.

B	J	S	$J\bar{S}$	$B \rightarrow S$	$\bar{S}(B \vee J)$	B	J	S	$J\bar{S}$	$B \rightarrow S$	$\bar{S}(B \vee J)$
0	0	0	0	1	0	1	0	0	0	0	1
0	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	1	1	1	1	0	1	0	1
0	1	1	0	1	0	1	1	1	0	1	0

были истинными. Из таблицы 3.3 находим, что показания совместны при $B = 0, J = 1, S = 0$ (третья строка таблицы). Таким образом доказано, что если принять гипотезу об истинности всех показаний, то виновные найдены: \bar{B}, J, \bar{S} .

Рассмотрим другую модель дополнительной гипотезы: невиновный говорит правду, а виновный — неизвестно что. Тогда истинными должны быть формулы:

$$\bar{B} \rightarrow J\bar{S}, \quad \bar{J} \rightarrow (B \rightarrow S), \quad \bar{S} \rightarrow \bar{S}(B \vee J). \quad (3.4)$$

Для решения задачи нужно заполнить таблицу, аналогичную таблице 3.3, и выделить в ней строки, где все формулы (3.4) истинны.

Задача 3.2. Рассмотрите еще одну гипотезу в задаче Кислера: виновный говорит правду, а невиновный лжет.

В задаче Кислера по синтаксической записи формул с помощью таблицы истинности логических связок были получены их семантические значения. Рассмотрим пример обратной задачи: имея таблицу истинности для некоторой формулы, получить синтаксическую запись этой формулы (т. е. синтезировать текст с заданным смыслом).

Пример 3.8 (задача о путешественнике). Путешественник попал в один из двух городов и желает знать, в какой именно. Он имеет право задать один вопрос, причем ответом на него может быть «да» или «нет». Будем считать, что ему не обязательно говорят правду.

Введем две переменные X и Y :

$$X = \begin{cases} 1, & \text{Санкт-Петербург,} \\ 0, & \text{Москва.} \end{cases} \quad Y = \begin{cases} 1, & \text{собеседник правдив,} \\ 0, & \text{в противном случае.} \end{cases}$$

Путешественник должен задать вопрос: *верно ли, что $F(X, Y)$* ? Таблица истинности для формулы $F(X, Y)$, очевидно, выглядит следующим образом (см. таблицу 3.4).

Таблица 3.4.

X	Y	$F(X, Y)$	X	Y	$F(X, Y)$
0	0	1	1	0	0
0	1	0	1	1	1

Задача построения формулы $F(X, Y)$ по таблице истинности (см. таблицу 3.4) будет решена в разделе 3.1.5 (см. пример 3.22).

3.1.2. Равносильность формул логики высказываний

Нетрудно привести примеры формул, которые, являясь синтаксически различными, имеют одинаковые значения. Например, формулы $X \vee Y$ и $Y \vee X$. Подобные формулы будем называть *равносильными*. Дадим точное определение.

Определение 3.5. Формулы $F(X_1, \dots, X_n)$ и $G(X_1, \dots, X_n)$ называют *равносильными* (символически обозначают $F = G$), если на любом булевом наборе (a_1, \dots, a_n) выполняется равенство $F(a_1, \dots, a_n) = G(a_1, \dots, a_n)$.

Замечание 3.2

Благодаря введению фиктивных переменных можно считать, что наборы переменных у формул F и G совпадают.

Пример 3.9. Покажем, что две формулы логики высказываний $F = X \rightarrow Y$ и $G = \bar{X} \vee Y$ равносильны. Построим совместную таблицу истинности для F и G (таблица 3.5).

Таблица 3.5.

X	Y	$X \rightarrow Y$	$\bar{X} \vee Y$	X	Y	$X \rightarrow Y$	$\bar{X} \vee Y$
0	0	1	1	1	0	0	0
0	1	1	1	1	1	1	1

Из таблицы 3.5 видно, что столбцы формул F и G совпадают. Это означает, что формулы F и G равносильны.

Используя определение равносильности, легко доказать справедливость следующей теоремы.

Теорема 3.2. Отношение равносильности на множестве формул логики высказываний есть отношение эквивалентности.

Близким к понятию равносильности является понятие тождественной истинности.

Определение 3.6. Формулу $F(X_1, \dots, X_n)$ называют *тождественно истинной*, или *тавтологией*, если на любом булевом наборе (a_1, \dots, a_n) справедливо $F(a_1, \dots, a_n) = 1$.

Пример 3.10. Покажем, что формула $F = XY \rightarrow X$ является тождественно истинной. Составим таблицу истинности формулы F (таблица 3.6).

Таблица 3.6.

X	Y	$XY \rightarrow X$	X	Y	$XY \rightarrow X$
0	0	1	1	0	1
0	1	1	1	1	1

Видно, что столбец формулы F состоит из одних единиц. Это означает, что формула F тождественно истинна.

Двойственным к понятию тождественной истинности является понятие тождественной ложности.

Определение 3.7. Формулу $F(X_1, \dots, X_n)$ называют *тождественно ложной*, или *противоречием*, если на любом булевом наборе (a_1, \dots, a_n) справедливо $F(a_1, \dots, a_n) = 0$.

Понятия равносильности и тождественной истинности связывает следующая теорема.

Теорема 3.3. Формулы F и G равносильны тогда и только тогда, когда формула $F \leftrightarrow G$ является тождественно истинной.

Доказательство. Используя определение равносильности и таблицу истинности связок, для операции \leftrightarrow получаем требуемое. ■

В логике высказываний часто приходится преобразовывать формулы, сохраняя их равносильность. Для таких преобразований используются так называемые *законы логики высказываний*. Приведем некоторые из этих законов.

Теорема 3.4. Пусть F, G и H — некоторые формулы логики высказываний. Тогда имеют место следующие равносильности (законы).

- | | |
|--|---|
| 1) $FF = F$; | 2) $F \vee F = F$; |
| 3) $FG = GF$; | 4) $F \vee G = G \vee F$; |
| 5) $F(GH) = (FG)H$; | 6) $F \vee (G \vee H) = (F \vee G) \vee H$; |
| 7) $F(G \vee H) = FG \vee FH$; | 8) $F \vee GH = (F \vee G)(F \vee H)$; |
| 9) $F\bar{F} = 0$; | 10) $F \vee \bar{F} = 1$; |
| 11) $\overline{FG} = \bar{F} \vee \bar{G}$; | 12) $\overline{F \vee G} = \bar{F} \bar{G}$; |
| 13) $\overline{\bar{F}} = F$; | 14) $F \rightarrow G = \bar{F} \vee G$; |
| 15) $F \leftrightarrow G = (F \rightarrow G)(G \rightarrow F)$. | |

Приведенные равносильности легко доказать с помощью таблиц истинности. Например, в примере 3.9 была установлена равносильность 14.

Прокомментируем список законов (равносильностей).

Законы 1 и 2 называют *идемпотентностью*, 3 и 4 — *коммутативностью*, 5 и 6 — *ассоциативностью* соответственно конъюнкции и дизъюнкции. Ассоциативность конъюнкции означает, что в конъюнкции трех формул скобки можно ставить как угодно, а следовательно, вообще не ставить. Из этого утверждения следует, что в конъюнкции четырех, пяти и т. д. (любого конечного числа) формул скобки можно ставить как угодно или вообще не ставить. Аналогичное замечание можно сделать и для дизъюнкции.

Закон 7 называют *дистрибутивностью конъюнкции относительно дизъюнкции*, а закон 8 — *дистрибутивностью дизъюнкции относительно конъюнкции*. Для применения этих законов в преобразованиях формул удобно иметь в виду следующий аналог. В законе 7 заменим формулы F, G и H соответственно буквами a, b и c , конъюнкцию — умножением $*$, дизъюнкцию — сложением $+$. Получим известное числовое тождество:

$$a * (b + c) = a * b + a * c.$$

Данное тождество есть *дистрибутивность умножения чисел относительно сложения*. В средней школе применение этого равенства слева направо называют раскрытием скобок, а справа налево — вынесением общего множителя. Отличие операций над высказываниями \wedge и \vee от числовых операций $*$ и $+$ состоит в том, что для высказываний выполняются обе дистрибутивности, а для чисел — только одна. Сложение не дистрибутивно относительно умножения.

Закон 9 называют *законом противоречия*, закон 10 — *законом исклю-*

ченного третьего¹, законы 11 и 12 — законами де Моргана, закон 13 — снятием двойного отрицания или инволютивностью, а установленный в примере 3.9 закон 14 — законом контрапозиции.

Историческая справка

Огастес (Август) де Морган (англ. Augustus de Morgan) — шотландский математик и логик, профессор математики в Университетском колледже Лондона. Первый президент Лондонского математического общества. Основные труды де Моргана посвящены алгебре, математическому анализу и математической логике. Его работа «Формальная логика, или Исчисление необходимых и вероятностных умозаключений» была опубликована одновременно с «Математическим анализом логики» Джорджа Буля.

Лемма 3.1 (о подстановке). Если в формуле F заменить подформулу G равносильной ей формулой \widehat{G} , то получим формулу \widehat{F} , равносильную исходной формуле F .

Доказательство. При любой подстановке констант в формулы F и \widehat{F} , начиная с некоторого места, процессы их вычисления совпадают, так как результаты вычисления формул G и \widehat{G} совпадают по условию леммы. ■

Используя лемму 3.1 и законы логики высказываний теоремы 3.4, получаем еще один способ доказательства равносильности двух формул. Этот способ состоит в переходе от одной формулы к другой с помощью равносильных преобразований. Проиллюстрируем второй способ на примере.

Пример 3.11. Доказать равносильность формул

$$X(Z \rightarrow Y) \vee (X \rightarrow Z)Y \text{ и } (X \vee Y)(Y \vee \overline{Z}).$$

Запишем цепочку равносильностей. Над знаком равносильности ($=$) будем ставить номера использованных равносильностей из теоремы 3.4.

$$\begin{aligned} X(Z \rightarrow Y) \vee (X \rightarrow Z)Y &\stackrel{14}{=} X(\overline{Z} \vee Y) \vee (\overline{X} \vee Z)Y \stackrel{8}{=} \\ &\stackrel{8}{=} (X(\overline{Z} \vee Y) \vee \overline{X} \vee Z) (X(\overline{Z} \vee Y) \vee Y) \stackrel{8}{=} \\ &\stackrel{8}{=} (X \vee \overline{X} \vee Z)(\overline{Z} \vee Y \vee \overline{X} \vee Z)(X \vee Y)(\overline{Z} \vee Y \vee Y) \stackrel{2,4,10}{=} \\ &\stackrel{2,4,10}{=} (X \vee Y)(Y \vee \overline{Z}). \end{aligned}$$

3.1.3. Принцип двойственности

В этом подразделе будем рассматривать формулы, содержащие только логические связки \wedge , \vee и \neg .

¹ Равносильности 9 и 10 называют также законами *дополнительности*.

Заметим, что большинство равносильностей, приведенных в теореме 3.4, существует в двух похожих формах. Рассмотрим логические законы 1–12. Если конъюнкцию заменить дизъюнкцией, дизъюнкцию — конъюнкцией, 0 на 1, а 1 на 0, то равносильность, записанная в левом столбце, перейдет в равносильность, записанную в правом столбце.

Дадим формальные определения.

Определение 3.8. Операцию конъюнкции называют *двойственной* для операции дизъюнкции, а операцию дизъюнкции называют двойственной для операции конъюнкции.

Формулы F и F^* называют *двойственными*, если формула F^* получается из формулы F путем замены в ней каждой операции на двойственную.

Для булевого набора $a = (a_1, \dots, a_n)$ определим *двойственный* набор $a^* = (b_1, \dots, b_n)$, где $b_i = \neg a_i$, $i \in 1 : n$.

Пример 3.12. Для формулы $F = X(Y \vee Z)$ двойственной является $F^* = X \vee YZ$.

Для формулы $G = \overline{XY}$ двойственной является $G^* = \overline{X \vee Y}$.

Прежде чем ввести принцип двойственности, рассмотрим вспомогательное утверждение.

Лемма 3.2. Формула $F(X_1, \dots, X_n)$ принимает значение 1 на булевом наборе $a = (a_1, \dots, a_n)$ тогда и только тогда, когда двойственная формула $F^*(X_1, \dots, X_n)$ принимает значение 0 на двойственном наборе a^* , т. е. справедливо равенство

$$\overline{F(X_1, \dots, X_n)} = F^*(\overline{X_1}, \dots, \overline{X_n}). \quad (3.5)$$

Доказательство. Проведем индукцию по k — числу логических связок формулы F .

Если $k = 0$, то $F = F^* = X_i$ и утверждение очевидно.

Предположим, что утверждение леммы справедливо при числе логических связок меньше k , и докажем его справедливость для k . Тогда формула F должна иметь вид: $\neg G, G \vee H$ или $G \wedge H$. Рассмотрим, например, второй случай $F = G \vee H$. Тогда $F^* = G^* \wedge H^*$. По индукционному предположению утверждение леммы справедливо для G^* и H^* . Имеем

$$\begin{aligned} \overline{F(X_1, \dots, X_n)} &= \overline{G(X_1, \dots, X_n) \vee H(X_1, \dots, X_n)} \stackrel{12}{=} \\ &\stackrel{12}{=} \overline{G(X_1, \dots, X_n)} \wedge \overline{H(X_1, \dots, X_n)} = \\ &= G^*(\overline{X_1}, \dots, \overline{X_n}) \wedge H^*(\overline{X_1}, \dots, \overline{X_n}) = F^*(\overline{X_1}, \dots, \overline{X_n}). \end{aligned}$$

Аналогично доказываются два других случая. ■

Пример 3.13. Рассмотрим формулу $F = X(Y \vee Z)$ из примера 3.12. Имеем:

$$\overline{F(X, Y, Z)} = \overline{X(Y \vee Z)} \stackrel{11}{=} \overline{X} \vee \overline{(Y \vee Z)} \stackrel{12}{=} \overline{X} \vee \overline{Y} \overline{Z} = F^*(\overline{X}, \overline{Y}, \overline{Z}).$$

Следствие 3.1. Равенство (3.5) можно переписать в виде:

$$F^*(X_1, \dots, X_n) = \overline{F(\overline{X_1}, \dots, \overline{X_n})}. \quad (3.6)$$

Это соотношение может служить другим равносильным определением двойственной формулы. Из равенства (3.6) очевидно следует тождество

$$(F^*)^* = F.$$

Из леммы 3.2 и определения равносильности очевидным образом вытекает *принцип (закон) двойственности*.

Теорема 3.5 (принцип двойственности). Если формулы F и G равносильны, то равносильны и двойственные им формулы, т. е. формула F^* равносильна формуле G^* .

В качестве примера можно привести уже упоминавшиеся в начале подраздела «парные» равносильности из теоремы 3.4.

Замечание 3.3

В дальнейшем принцип двойственности и соотношения (3.5) и (3.6) будут использоваться неоднократно (см. разделы 3.1.5 и 3.2.6). Например, установив справедливость некоторых представлений для дизъюнкции, легко составить аналогичные соотношения и для конъюнкции.

3.1.4. Логическое следствие

Рассмотрим вопрос: является ли данное утверждение следствием других? Введем необходимые понятия.

Определение 3.9. Формулу $G(X_1, \dots, X_n)$ называют *логическим следствием* формул $F_1(X_1, \dots, X_n), \dots, F_k(X_1, \dots, X_n)$, если для любой интерпретации (a_1, \dots, a_n) из того, что

$$F_1(a_1, \dots, a_n) = 1, \dots, F_k(a_1, \dots, a_n) = 1,$$

следует, что $G(a_1, \dots, a_n) = 1$.

Пример 3.14. На складе совершено хищение. Подозрение пало на трех человек: Брауна, Джонсона и Смита, они были доставлены для допроса. Установлено следующее.

1. Никто, кроме Брауна, Джонсона и Смита, не был замешан в деле.
2. Браун никогда не ходит на дело без по крайней мере одного соучастника.
3. Смит не виновен.

Ответим на вопрос: *виновен ли Джонсон?*

Обозначим через X утверждение «Браун виновен», через Y — «Джонсон виновен», через Z — «Смит виновен». Запишем факты 1–3 с помощью формул исчисления высказываний:

$$F_1 = X \vee Y \vee Z, \quad F_2 = X \rightarrow Y \vee Z, \quad F_3 = \bar{Z}.$$

Предполагаемый ответ: «Джонсон виновен» обозначим через G . Проверим, является ли формула G логическим следствием формул F_1, F_2, F_3 . Построим совместную таблицу истинности формул F_1, F_2, F_3 и G (таблица 3.7).

Таблица 3.7.

X	Y	Z	F_1	F_2	F_3	G	X	Y	Z	F_1	F_2	F_3	G
0	0	0	0	1	1		1	0	0	1	0	1	
0	0	1	1	1	0		1	0	1	1	1	0	
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	1	1	0		1	1	1	1	1	0	

Рассматриваем только те интерпретации, в которых все формулы F_1, F_2, F_3 истинны. Таких интерпретаций только две (в таблице 3.7 они выделены серым цветом). В этих интерпретациях значение формулы G также истинно. Следовательно, формула G — логическое следствие, т. е. Джонсон виновен.

Пример 3.15. Докажем, что формула $G = Y \rightarrow X$ не является логическим следствием формул $F_1 = X \vee Y$, $F_2 = X \rightarrow Y$, $F_3 = Y$. Для этого построим совместную таблицу истинности формул F_1, F_2, F_3 и G (таблица 3.8).

Рассмотрим интерпретацию $X = 0, Y = 1$. Из таблицы 3.8 видно, что $F_1(0, 1) = F_2(0, 1) = F_3(0, 1) = 1$, но $G(0, 1) = 0$. Следовательно, формула G не является логическим следствием формул F_1, F_2, F_3 .

Понятие логического следствия тесно связано с понятием выполнимости.

Таблица 3.8.

X	Y	F_1	F_2	F_3	G	X	Y	F_1	F_2	F_3	G
0	0	0	1	0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	1	1	1	1	1

Определение 3.10. Множество формул

$$\{F_1(X_1, \dots, X_n), \dots, F_k(X_1, \dots, X_n)\}, k \geq 1$$

называют *выполнимым* (*satisfiable*), если существует интерпретация (a_1, \dots, a_n) такая, что

$$F_1(a_1, \dots, a_n) = \dots = F_k(a_1, \dots, a_n) = 1.$$

Интерпретацию (a_1, \dots, a_n) называют *выполняющей* (*satisfiable assignment*).

Проверить выполнимость множества формул $\{F_1, \dots, F_k\}$ можно построением совместной таблицы истинности этих формул. Если найдется хотя бы одна строка, в которой в столбцах формул F_1, \dots, F_k стоят единицы, то это множество формул выполнимо. Если такой строки нет, то множество формул невыполнимо.

Так, множество формул $\{F_1, F_2, F_3, G\}$ из примера 3.15 выполнимо, поскольку в таблице 3.8 в последней строке в столбцах этих формул стоят единицы.

В дальнейшем будет необходимо следующее утверждение.

Теорема 3.6. Формула $G(X_1, \dots, X_n)$ является логическим следствием формул $F_1(X_1, \dots, X_n), \dots, F_k(X_1, \dots, X_n)$ тогда и только тогда, когда множество формул

$$L = \{F_1(X_1, \dots, X_n), \dots, F_k(X_1, \dots, X_n), \overline{G(X_1, \dots, X_n)}\}$$

невыполнимо.

Доказательство. Пусть формула G является следствием множества формул F_1, \dots, F_k . Предположим от противного, что множество L выполнимо. Это означает, что существует интерпретация (a_1, \dots, a_n) такая, что

$$F_1(a_1, \dots, a_n) = \dots = F_k(a_1, \dots, a_n) = \overline{G(a_1, \dots, a_n)} = 1. \quad (3.7)$$

Так как G — логическое следствие формул F_1, \dots, F_k , из (3.7) следует, что $G(a_1, \dots, a_n) = 1$. Полученное противоречие доказывает, что множество формул $\{F_1, \dots, F_k, \overline{G}\}$ невыполнимо.

Пусть теперь множество формул L невыполнимо. Рассмотрим интерпретацию (a_1, \dots, a_n) такую, что

$$F_1(a_1, \dots, a_n) = \dots = F_k(a_1, \dots, a_n) = 1. \quad (3.8)$$

Поскольку L невыполнимо, то $\overline{G(a_1, \dots, a_n)} = 0$. Тогда $G(a_1, \dots, a_n) = 1$. Следовательно, из равенств (3.8) следует равенство $G(a_1, \dots, a_n) = 1$. Это означает, что G — логическое следствие множества формул F_1, \dots, F_k . ■

Следствие 3.2. Формула G является логическим следствием формул F_1, \dots, F_k тогда и только тогда, когда формула $F_1 \wedge \dots \wedge F_k \wedge \overline{G}$ есть противоречие.

Пример 3.16. На некотором острове, населенном рыцарями и лжецами (рыцари говорят только правду, лжецы — только ложь), разнесся слух о том, что там зарыты сокровища. Путешественник прибывает на остров и спрашивает одного из местных жителей: «Есть ли на острове сокровища?» В ответ на вопрос путешественника он заявляет: «Сокровища на этом острове есть в том и только в том случае, если я рыцарь».

Можно ли определить, есть ли сокровища на острове?

Обозначим через X утверждение «местный житель — рыцарь», через Y — «Сокровища на острове есть». Тогда высказывание местного жителя «Сокровища на острове есть в том и только в том случае, если я рыцарь» запишется формулой: $X \leftrightarrow Y$. Тогда если он рыцарь, то его высказывание истинно, если лжец, то ложно.

Таким образом, вопрос сводится к следующему: является ли формула Y логическим следствием формулы $X \leftrightarrow (X \leftrightarrow Y)$. Для проверки этого утверждения достаточно на основании следствия 3.2 к теореме 3.6 доказать, что формула $(X \leftrightarrow (X \leftrightarrow Y))\overline{Y}$ есть противоречие. Построим ее таблицу истинности (таблица 3.9).

Таблица 3.9.

X	Y	$(X \leftrightarrow (X \leftrightarrow Y))\overline{Y}$	X	Y	$(X \leftrightarrow (X \leftrightarrow Y))\overline{Y}$
0	0	0	1	0	0
0	1	0	1	1	0

Формула $(X \leftrightarrow (X \leftrightarrow Y))\bar{Y}$ ложна в любой интерпретации, следовательно, она есть противоречие, и поэтому верно утверждение, что сокровища на острове есть.

Замечание 3.4

| Примеры 3.14 и 3.16 взяты из [53].

3.1.5. Нормальные формы в логике высказываний

Во многих случаях удобно рассматривать вместо формул общего вида формулы в некоторой канонической форме. Дадим необходимые определения.

Определение 3.11. *Литералом* называют переменную или ее отрицание, *элементарной конъюнкцией*, или *простой конъюнкцией*, или *конъюнктом* — литерал или конъюнкцию литералов.

Определение 3.12. Формулу, в которой знак отрицания находится только перед переменными, называют формулой с *тесными* отрицаниями.

Пример 3.17. Формулы $X \leftrightarrow \bar{Y}$, $\bar{X} \rightarrow \bar{Y}$ являются формулами с тесными отрицаниями, а формулы \overline{XY} , $X \vee \bar{Y} \vee \bar{Z}$ — нет.

Определение 3.13. Формула G имеет *дизъюнктивную нормальную форму* (ДНФ), если она является дизъюнкцией элементарных конъюнкций.

Пример 3.18. Формулы X , \bar{Y} , $X\bar{Y}$, $(X\bar{Y}) \vee (\bar{X}Z)$ имеют ДНФ, а формулы \overline{XY} , $X \vee Y \vee 0$, $X \rightarrow Y$ — нет.

Теорема 3.7. Для любой формулы F существует формула G , равносильная F и имеющая дизъюнктивную нормальную форму.

Доказательство. Следует из рассмотрения алгоритма 3.1, который по данной формуле F выдает формулу G , удовлетворяющую условию теоремы. ■

Алгоритм 3.1. Приведение к ДНФ

Исключаем из исходной формулы эквиваленции и импликации

// Используем законы 14 и 15 теоремы 3.4

Преобразуем формулу к формуле с тесными отрицаниями.

// Используем законы 11 и 12 теоремы 3.4

if формула содержит подформулу вида $F(G \vee H)$ **then**

 заменяем ее на равносильную формулу $FG \vee FH$

end if

Пример 3.19. Применение алгоритма 3.1 проиллюстрируем на примере формулы $F = \overline{(X \leftrightarrow Y)}X$. Как и ранее (см. пример 3.11), в цепочке равносильностей будем ставить номера использованных тавтологий из теоремы 3.4.

1 Выполним первый шаг.

$$\overline{(X \leftrightarrow Y)}X \stackrel{15}{=} \overline{(X \rightarrow Y)(Y \rightarrow X)}X \stackrel{14}{=} \overline{(\overline{X} \vee Y)(\overline{Y} \vee X)}X.$$

2 Перейдем ко второму шагу.

$$\overline{(\overline{X} \vee Y)(\overline{Y} \vee X)}X \stackrel{11}{=} \overline{(\overline{X} \vee Y) \vee (\overline{Y} \vee X)}X \stackrel{12,13}{=} (\overline{X\overline{Y}} \vee \overline{Y\overline{X}})X.$$

3 Выполнение третьего шага заключается в применении дистрибутивности.

$$(\overline{X\overline{Y}} \vee \overline{Y\overline{X}})X \stackrel{7}{=} \overline{X\overline{Y}}X \vee \overline{Y\overline{X}}X = G.$$

Алгоритм на этом завершен. Формула G имеет ДНФ. Но эту формулу можно упростить. Действительно, формула $\overline{Y\overline{X}}X = 0$, а формула $\overline{X\overline{Y}}X = \overline{X\overline{Y}}$. Тогда

$$\overline{X\overline{Y}}X \vee \overline{Y\overline{X}}X = \overline{X\overline{Y}} = \widehat{G}.$$

Формула \widehat{G} , как и G , имеет ДНФ и равносильна исходной формуле F .

Пример 3.19 показывает, что может существовать несколько равносильных формул, имеющих разное представление в дизъюнктивной нормальной форме. Иногда это обстоятельство является неудобным. Чтобы его исключить, вводится более узкое понятие, чем ДНФ.

Определение 3.14. Формула $F(X_1, \dots, X_n)$ имеет *совершенную дизъюнктивную нормальную форму* (СДНФ), если выполнены следующие условия:

- 1) F имеет дизъюнктивную нормальную форму;
- 2) каждая элементарная конъюнкция ДНФ содержит один и только один из литералов X_i или $\overline{X_i}$ для любого $i = 1, \dots, n$;
- 3) F не содержит одинаковых элементарных конъюнкций.

Пример 3.20. Формулы X , $\overline{X\overline{Y}}$, $\overline{X\overline{Y}} \vee X\overline{Y}$ имеют СДНФ.

Формулы $\overline{X\overline{Y}}$, $X\overline{Y} \vee \overline{X}Z$, $X\overline{Y} \vee X\overline{Y} \vee YX$ не имеют СДНФ. Для первой формулы не выполняется первое условие, для второй — второе условие, для третьей формулы — последнее условие из определения СДНФ.

Теорема 3.8. Для любой выполнимой формулы F существует равносильная ей формула G , имеющая совершенную дизъюнктивную нормальную форму.

Доказательство. Доказательство данной теоремы, как и теоремы 3.7, следует из алгоритма 3.2, который по формуле F выдает формулу G , удовлетворяющую условию теоремы. ■

Алгоритм 3.2. Первый алгоритм приведения к СДНФ

Применяя алгоритм 3.1, приводим формулу к ДНФ: $F(X_1, \dots, X_n) = C_1 \vee \dots \vee C_k$

for each C_i **do**

if существует $j \in 1 : n \mid X_j \notin C_i$ и $\overline{X_j} \notin C_i$ **then**

 // равносильным преобразованием заменяем C_i на две элементарные конъюнкции

$$C_i \leftarrow C_i (X_j \vee \overline{X_j}) = C_i X_j \vee C_i \overline{X_j} \quad (3.9)$$

 // преобразование (3.9) равносильно, так как $X_i \vee \overline{X_i} = 1$ и $C \wedge 1 = C$

end if

if C_i содержит два вхождения одного литерала **then**

 одно из них вычеркиваем

end if

if $C_i = \dots X_j \dots \wedge \overline{X_j} \dots$ **then**

 удаляем из F элементарную конъюнкцию C_i

end if

end for

// Приводим «подобные» члены

if формула F содержит одинаковые элементарные конъюнкции **then**

 оставляем одну, остальные удаляем

end if

Пример 3.21. Применением алгоритма 3.2 к формуле $F = XY \vee X\overline{Z}$.

Эта формула уже имеет ДНФ, поэтому выполнение алгоритма начинается с шага 2. Применяя преобразование (3.9) к XY и $X\overline{Z}$, имеем

$$XY \vee X\overline{Z} = XYZ \vee XY\overline{Z} \vee X\overline{Z}Y \vee X\overline{Z}\overline{Y}.$$

Одинаковых и противоположных литералов в элементарных конъюнкциях полученной формулы нет, поэтому шаг 3 не выполняется.

Формула содержит одинаковые элементарные конъюнкции — вторую и третью. При выполнении четвертого шага будет оставлена одна из них. Получается формула G :

$$G = XYZ \vee XY\bar{Z} \vee X\bar{Z}\bar{Y}.$$

Формула G равносильна F и имеет СДНФ.

Теорема 3.9. Требование выполнимости формулы F в формулировке теоремы 3.8 существенно.

Доказательство. Покажем, что если формула $F(X_1, \dots, X_n)$ невыполнима, то после ее приведения к ДНФ каждая элементарная конъюнкция будет содержать хотя бы одну пару противоположных литералов X и \bar{X} .

Предположим, что в полученной ДНФ найдется элементарная конъюнкция, не содержащая противоположных литералов. Пусть это конъюнкция $X_{i_1} \wedge \dots \wedge X_{i_k}$. Тогда для интерпретации (a_1, \dots, a_n) , где

$$a_i = \begin{cases} 1, & i = i_1, \dots, i_k, \\ 0, & \text{иначе} \end{cases}$$

справедливо равенство $F(a_1, \dots, a_n) = 1$, что противоречит невыполнимости формулы F .

Так как каждая элементарная конъюнкция будет содержать хотя бы одну пару противоположных литералов, то на шаге 3 алгоритма 3.2 все элементарные конъюнкции будут вычеркнуты. ■

Имеется еще один способ приведения к СДНФ, основанный на построении таблицы истинности исходной формулы. Введем обозначения.

$$X^\alpha = \begin{cases} X, & \text{если } \alpha = 1, \\ \bar{X}, & \text{если } \alpha = 0. \end{cases}$$

Алгоритм 3.3. Второй алгоритм приведения к СДНФ

// Инициализация

Составляем таблицу истинности формулы $F(X_1, \dots, X_n)$

$G(X_1, \dots, X_n) \leftarrow 0$

for each $(\alpha_1, \dots, \alpha_n) \mid \alpha_i \in \{0, 1\}$ **do**

 // рассматриваем строки таблицы истинности, в которых в столбце F стоит 1

if $F(\alpha_1, \dots, \alpha_n) = 1$ **then**

// каждой такой строке сопоставляем элементарную конъюнкцию $X_1^{\alpha_1} \dots X_n^{\alpha_n}$
 $G(X_1, \dots, X_n) \leftarrow G(X_1, \dots, X_n) \vee X_1^{\alpha_1} \dots X_n^{\alpha_n}$
 end if
 end for

Теорема 3.10. Построенная согласно алгоритму 3.3 формула G имеет СДНФ и равносильна исходной формуле F .

Доказательство. Очевидно, что формула G имеет СДНФ по построению.

Заметим, что каждая элементарная конъюнкция $X_1^{\alpha_1} \dots X_n^{\alpha_n}$, добавляемая к G , принимает значение 1 только на наборе $(\alpha_1, \dots, \alpha_n)$. Поэтому формула G принимает значение 1 на тех и только тех наборах, где $F = 1$. Отсюда следует, что таблицы истинности F и G совпадают. ■

Определение 3.15. Элементарную конъюнкцию $X_1^{\alpha_1} \dots X_n^{\alpha_n}$ называют *конституентой единицы* набора $(\alpha_1, \dots, \alpha_n)$ (символически обозначают $K_{\alpha_1, \dots, \alpha_n}^1$).

Пример 3.22. Вернемся к задаче о путешественнике (см. пример 3.8). Как было установлено, таблица истинности формулы $F(X, Y)$ имеет вид (таблица 3.10).

Таблица 3.10.

X	Y	$F(X, Y)$	X	Y	$F(X, Y)$
0	0	1	1	0	0
0	1	0	1	1	1

Применив алгоритм 3.3, имеем $F(X, Y) = XY \vee \overline{X}\overline{Y}$. Вопрос, который должен задать путешественник, звучит так: «Правда ли, что это Санкт-Петербург и ты правдив, или это Москва и ты лжешь?»

Пример 3.23. С помощью алгоритма 3.3 построим СДНФ для формулы $F(X_1, X_2, X_3) = X_1(X_2 \rightarrow X_3)$.

Составим таблицу истинности формулы F (таблица 3.11).

Применив алгоритм 3.3, получим искомое представление:

$$F(X_1, X_2, X_3) = X_1(X_2 \rightarrow X_3) = X_1X_2X_3 \vee X_1\overline{X_2}X_3 \vee X_1\overline{X_2}\overline{X_3}.$$

Рассмотрим еще одну каноническую форму записи формул логики высказываний — конъюнктивную нормальную форму. Она получается из ДНФ заменой \wedge на \vee и \vee на \wedge . Дадим точные определения.

Таблица 3.11.

X_1	X_2	X_3	$X_1(X_2 \rightarrow X_3)$	X_1	X_2	X_3	$X_1(X_2 \rightarrow X_3)$
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	0	1	1	1	1

Определение 3.16. *Элементарной дизъюнкцией, или простой дизъюнкцией, или дизъюнктом называют литерал или дизъюнкцию литералов.*

Замечание 3.5

Очевидно, что элементарная дизъюнкция является двойственной к элементарной конъюнкции. Это свойство позволяет свести определяемые новые понятия к ДНФ и СДНФ.

Определение 3.17. Формула G имеет конъюнктивную нормальную форму (КНФ), если она является конъюнкцией элементарных дизъюнкций.

Используя понятие двойственной формулы, можно дать и другое равносильное определение КНФ.

Формула G имеет КНФ, если двойственная ей формула G^* определена (т. е. не содержит связок \rightarrow и \leftrightarrow) и находится в ДНФ.

Пример 3.24. Формулы $X, \bar{Y}, X \vee \bar{Y}, X\bar{Y}, (X \vee \bar{Y})(X \vee Z)$ имеют КНФ, а формулы $X \rightarrow Y, \bar{X} \vee \bar{Y}, X\bar{Y} \vee X\bar{Z}$ — нет.

Для КНФ справедливо утверждение, аналогичное теореме 3.7.

Теорема 3.11. Для любой формулы F существует формула G , равносильная F и имеющая конъюнктивную нормальную форму.

Доказательство. Доказательство теоремы легко следует из анализа алгоритма приведения к КНФ.

Этот алгоритм легко получить из алгоритма 3.1 приведения к ДНФ, если на шаге 3 вместо закона дистрибутивности 7 (см. теорему 3.4) использовать двойственную равносильность 8, т. е. если формула содержит подформулу вида $F \vee GH$, то заменить ее на формулу $(F \vee G)(F \vee H)$. ■

Как и для случая ДНФ, введем более узкое понятие, нежели КНФ.

Определение 3.18. Формула $F(X_1, \dots, X_n)$ имеет совершенную конъюнктивную нормальную форму (СКНФ), если выполнены следующие условия:

- 1) F имеет КНФ;
- 2) каждая элементарная дизъюнкция содержит один и только один из литералов X_i или $\neg X_i$ для любого $i \in 1 : n$;
- 3) F не содержит одинаковых элементарных дизъюнкций.

Как и для КНФ, можно дать другое равносильное определение СКНФ, используя понятие двойственности.

Формула G имеет СКНФ, если двойственная ей формула G^* находится в СДНФ.

Пример 3.25. Формулы X , $\bar{X} \vee Y$, $(\bar{X} \vee Y)(X \vee \bar{Y})$ имеют СКНФ.

Формулы $\bar{X} \vee \bar{Y}$, $(X \vee Y)(\bar{X} \vee Z)$, $(X \vee Y)(X \vee \bar{Y})(Y \vee X)$ не имеют СКНФ. Для первой формулы не выполняется первое условие, для второй формулы — второе условие, для третьей формулы — последнее условие из определения СКНФ.

Теорема 3.12. Для любой не тождественно истинной формулы F существует равносильная ей формула G , имеющая совершенную конъюнктивную нормальную форму.

Доказательство. Доказательство этой теоремы, как теоремы 3.8, легко следует из алгоритма 3.4, который по формуле F выдает формулу G , удовлетворяющую условию теоремы. ■

Алгоритм 3.4. Первый алгоритм приведения к СКНФ

Применяя рассуждения теоремы 3.11, приводим формулу к КНФ:

$$F(X_1, \dots, X_n) = D_1 \wedge \dots \wedge D_k$$

for each D_i **do**

if существует $j \in 1 : n \mid X_j \notin D_i$ и $\bar{X}_j \notin D_i$ **then**

 // равносильным преобразованием заменяем D_i на две элементарные дизъюнкции

$$D_i \leftarrow (D_i \vee X_j)(D_i \vee \bar{X}_j) \tag{3.10}$$

 // преобразование (3.10) равносильно, так как $X_j \bar{X}_j = 0$ и $D_i \vee 0 = D_i$

end if

if D_i содержит два вхождения одного литерала **then**

 одно из них вычеркиваем

end if

if $D_i = \dots X_j \vee \dots \vee \bar{X}_j \dots$ **then**

 удаляем из F элементарную дизъюнкцию D_i

end if

end for

// Приводим «подобные» члены

if формула F содержит одинаковые элементарные дизъюнкции then

оставляем одну, остальные удаляем

end if

Замечание 3.6

В алгоритме 3.2 приведения к СДНФ используется равносильное преобразование (3.9). В силу двойственности СДНФ и СКНФ на том же шаге алгоритма 3.4 применяется двойственное равносильное преобразование (3.10).

Пример 3.26. В качестве примера работы алгоритма 3.4 рассмотрим формулу $F(X, Y, Z) = \overline{(X \vee Z)}(X \rightarrow Y)$.

Приводим формулу к КНФ.

$$\overline{(X \vee Z)}(X \rightarrow Y) \stackrel{14}{=} \overline{(X \vee Z)}(\overline{X \vee Y}) \stackrel{12}{=} \overline{X} \overline{Z}(\overline{X \vee Y}).$$

Эта формула имеет КНФ, но не имеет СКНФ. Построим ее СКНФ. Применяя преобразование (3.10), имеем:

$$\begin{aligned} \overline{X} \overline{Z}(\overline{X \vee Y}) &= (\overline{X} \vee Y \overline{Y} \vee Z \overline{Z})(\overline{X} \overline{X} \vee Y \overline{Y} \vee \overline{Z})(\overline{X} \vee Y \vee Z \overline{Z}) = \\ &= (\overline{X} \vee Y \vee Z)(\overline{X} \vee \overline{Y} \vee Z)(\overline{X} \vee Y \vee \overline{Z}) \wedge \\ &\wedge (\overline{X} \vee \overline{Y} \vee \overline{Z})(X \vee Y \vee \overline{Z})(X \vee \overline{Y} \vee \overline{Z}). \end{aligned}$$

Задание 3.1. Докажите, что требование нетождественной истинности формулы F в формулировке теоремы 3.12 существенно.

Имеется еще один способ приведения к СКНФ, основанный на построении таблицы истинности исходной формулы. Он очевидным образом следует из алгоритма 3.3 с учетом двойственности СДНФ и СКНФ и соотношения (3.5).

Алгоритм 3.5. Второй алгоритм приведения к СКНФ

// Инициализация

Составляем таблицу истинности формулы $F(X_1, \dots, X_n)$

$G(X_1, \dots, X_n) \leftarrow 0$

for each $(\alpha_1, \dots, \alpha_n) \mid \alpha_i \in \{0, 1\}$ do

// рассматриваем строки таблицы истинности, в которых в столбце F стоит 0

if $F(\alpha_1, \dots, \alpha_n) = 0$ then

// строке сопоставляем элементарную дизъюнкцию $\overline{X_1^{\alpha_1}} \vee \dots \vee \overline{X_n^{\alpha_n}}$

$G(X_1, \dots, X_n) \leftarrow G(X_1, \dots, X_n) \wedge (\overline{X_1^{\alpha_1}} \vee \dots \vee \overline{X_n^{\alpha_n}})$

end if

end for

Теорема 3.13. Построенная согласно алгоритму 3.5 формула G имеет СКНФ и равносильна исходной формуле F .

Доказательство. Очевидно, что формула G имеет СКНФ по построению.

Заметим, что каждая элементарная дизъюнкция $\overline{X_1^{\alpha_1}} \vee \dots \vee \overline{X_n^{\alpha_n}}$, добавляемая к G , принимает значение 0 только на наборе $(\alpha_1, \dots, \alpha_n)$. Поэтому формула G принимает значение 0 на тех и только на тех наборах, где $F = 0$. Отсюда следует, что таблицы истинности F и G совпадают. ■

Определение 3.19. Элементарную дизъюнкцию $\overline{X_1^{\alpha_1}} \vee \dots \vee \overline{X_n^{\alpha_n}}$ называют *конституентой нуля* набора $(\alpha_1, \dots, \alpha_n)$ (символически обозначают $K_{\alpha_1, \dots, \alpha_n}^0$).

Очевидно следующее утверждение.

Лемма 3.3. Конституента единицы и конституента нуля связаны соотношением

$$\overline{K_{\alpha_1, \dots, \alpha_n}^0} = K_{\alpha_1, \dots, \alpha_n}^1.$$

Пример 3.27. С помощью алгоритма 3.5 построим СКНФ для формулы

$$F(X_1, X_2, X_3) = (\overline{X_1} \rightarrow X_3) \rightarrow (\overline{X_2} \rightarrow \overline{X_1}).$$

Составим таблицу истинности формулы F (таблица 3.12).

Таблица 3.12.

X_1	X_2	X_3	$F(X_1, X_2, X_3)$	X_1	X_2	X_3	$F(X_1, X_2, X_3)$
0	0	0	1	1	0	0	1
0	0	1	0	1	0	1	1
0	1	0	1	1	1	0	0
0	1	1	0	1	1	1	0

Применив алгоритм 3.5, получим искомое представление:

$$F(X_1, X_2, X_3) = (X_1 \vee X_2 \vee \overline{X_3})(X_1 \vee \overline{X_2} \vee \overline{X_3})(\overline{X_1} \vee \overline{X_2} \vee X_3)(\overline{X_1} \vee \overline{X_2} \vee \overline{X_3})$$

3.1.6. Контактные схемы

В этом подразделе рассмотрим применение логики высказываний к анализу так называемых контактных схем.

Определение 3.20. *Контактом* будем называть устройство, которое в процессе работы может быть в двух состояниях: контакт замкнут и контакт разомкнут.

Контакт X на чертеже будем изображать следующим образом (рисунок 3.1).

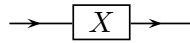


Рис. 3.1.

Контакты можно соединять между собой различными способами (рисунок 3.2).

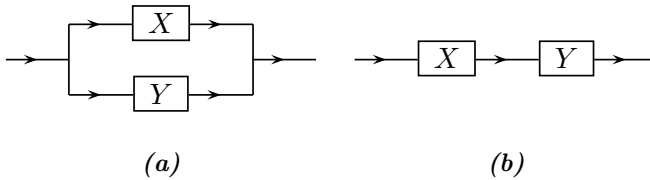


Рис. 3.2.

Определение 3.21. Первое соединение контактов (рисунок 3.2a) называют *параллельным*, второе (рисунок 3.2b) — *последовательным*. Контакты, соединенные между собой, называют *контактной схемой*.

Будем предполагать наличие у схемы двух выделенных точек входа и выхода. Схему назовем *замкнутой*, если существует последовательность замкнутых контактов X_1, \dots, X_n такая, что X_i соединен с X_{i+1} , X_1 — с входом, X_n — с выходом. Схему, не являющуюся замкнутой, назовем *разомкнутой*.

Каждому контакту поставим в соответствие переменную логики высказываний, которая равна 1 тогда и только тогда, когда контакт замкнут. Переменную и контакт будем обозначать одной буквой.

Пусть схема S построена из контактов X_1, \dots, X_n с помощью параллельного и последовательного соединений. Тогда по схеме S можно построить формулу логики высказываний F_S так, что параллельному соединению соответствует дизъюнкция, последовательному — конъюнкция.

Пример 3.28. Схеме S на рисунке 3.3 соответствует формула

$$F_S = X(Z \vee \bar{Y}) \vee \bar{X}Z \vee (X \vee \bar{Y})\bar{Z}.$$

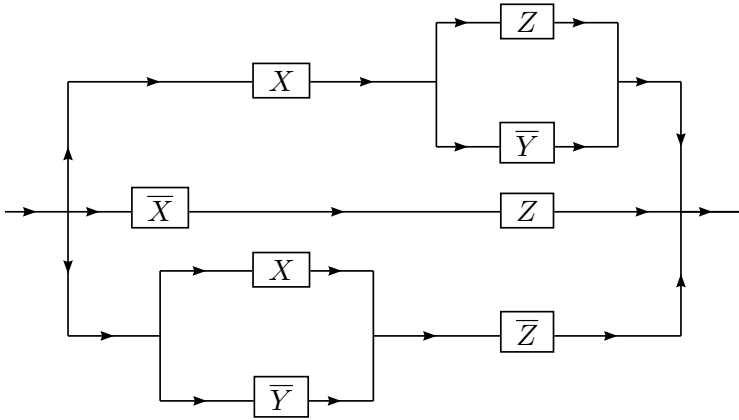


Рис. 3.3.

Через \bar{X} обозначают контакт, который замкнут тогда и только тогда, когда X разомкнут. Формула F_S «представляет схему» в следующем смысле: схема S замкнута в том и только в том случае, если F_S принимает значение 1.

В силу определения контактными схемам соответствуют формулы в дизъюнктивной или конъюнктивной нормальной формах. Нетрудно понять, что по всякой такой формуле F можно восстановить схему, которую формула F «представляет».

Пусть схемам S и T соответствуют формулы F_S и F_T в описанном ранее смысле. Тогда, если схемы S и T эквивалентны (т. е. замкнуты и разомкнуты одновременно), то F_S и F_T равносильны, и наоборот. Этот факт используется для решения задачи минимизации контактных схем, которая состоит в том, чтобы по данной схеме S найти схему T , эквивалентную S и содержащую меньше контактов.

Один из путей решения этой задачи состоит в переходе к формуле F_S и в отыскании формулы G , равносильной F_S и содержащей меньше вхождений переменных.

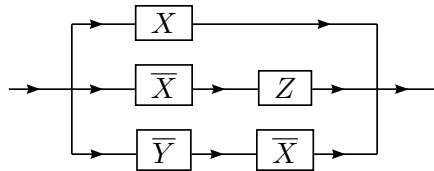


Рис. 3.4.

Например, формула F_S равносильна формуле $X \vee \bar{X}Z \vee \bar{Y}\bar{Z}$. Следо-

вательно, схема, приведенная на рисунке 3.3, эквивалентна схеме, которая имеет на три контакта меньше (рисунок 3.4).

Пример 3.29. Требуется, чтобы включение света в комнате осуществлялось с помощью трех различных выключателей таким образом, чтобы нажатие на любой из них приводило к включению света, если он был выключен, и выключению, если он был включен. Построить контактную схему, удовлетворяющую этому требованию.

Обозначим выключатели переменными X_1 , X_2 и X_3 , а формулу для контактной схемы — $F(X_1, X_2, X_3)$. Построим ее таблицу истинности.

Очевидно, что $F(0, 0, 0) = 0$. Далее заметим, что если значения переменных X_1 , X_2 и X_3 в каждой следующей строке отличаются от предыдущей только в одной позиции, то значение формулы F меняется на противоположное. Это отражает требование о том, чтобы при нажатии на любой из выключателей свет выключался, если он был включен, и включался, если он был выключен.

Составим таблицу истинности для данного порядка строк (таблица 3.13).

Таблица 3.13.

X	Y	Z	$F(X_1, X_2, X_3)$	X	Y	Z	$F(X_1, X_2, X_3)$
0	0	0	0	1	0	1	0
0	1	0	1	1	0	0	1
0	1	1	0	1	1	0	0
0	0	1	1	1	1	1	1

По таблице истинности, используя алгоритм 3.3, построим СДНФ для формулы $F(X_1, X_2, X_3)$:

$$F(X_1, X_2, X_3) = XYZ \vee X\bar{Y}\bar{Z} \vee \bar{X}\bar{Y}Z \vee \bar{X}Y\bar{Z}.$$

Преобразуем формулу F к равносильной формуле G , более удобной для реализации схемы:

$$G = X(YZ \vee \bar{Y}\bar{Z}) \vee \bar{X}(\bar{Y}Z \vee Y\bar{Z}).$$

По формуле G построим искомую схему (рисунок 3.5).

Задача 3.3. Пусть каждый из трех членов комитета голосует за, нажимая кнопку. Постройте по возможности более простую цепь, которая бы

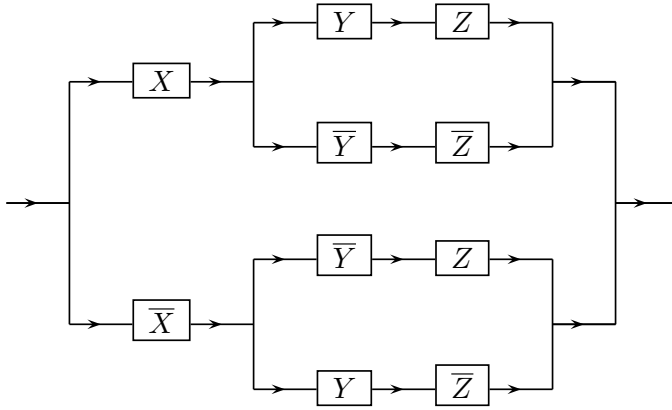


Рис. 3.5.

ла бы замкнута тогда и только тогда, когда не менее двух членов комитета голосуют за.

3.1.7. Метод минимизирующих карт

Определение 3.22. Дизъюнктивную нормальную форму называют *минимальной*, если она содержит наименьшее общее число вхождений переменных по сравнению со всеми равносильными ей дизъюнктивными нормальными формами.

Следовательно, минимальную ДНФ данной формулы можно найти, перебрав конечное число равносильных ей ДНФ и выбрав среди них ту, которая содержит минимальное число вхождений переменных. Однако при большом числе переменных такой перебор практически невыполним. Существуют эффективные способы нахождения минимальной ДНФ. Рассмотрим один из них — *метод минимизирующих карт*¹. Хотя он и не является самым эффективным, зато прост для изложения и не требует введения дополнительных понятий.

Пусть формула задана таблицей истинности или СДНФ. Составим следующую карту (таблица 3.14).

Каждую строку таблицы 3.14 определяет один из булевых наборов

$$\alpha_1, \alpha_2, \dots, \alpha_n, \quad \alpha_i = 0, 1, \quad i \in 1 : n. \quad (3.11)$$

¹ Вариант такого подхода известен под названием *метод неопределенных коэффициентов* или *гарвардский метод*.

Таблица 3.14.

0	\bar{X}_1	\bar{X}_2	...	\bar{X}_n	$\bar{X}_1\bar{X}_2$	$\bar{X}_1\bar{X}_3$...	$\bar{X}_{n-1}\bar{X}_n$...	$\bar{X}_1 \dots \bar{X}_{n-1} \bar{X}_n$
1	\bar{X}_1	\bar{X}_2	...	X_n	$\bar{X}_1\bar{X}_2$	$\bar{X}_1\bar{X}_3$...	$\bar{X}_{n-1}X_n$...	$\bar{X}_1 \dots \bar{X}_{n-1} X_n$
...
k	X_1	\bar{X}_2	...	\bar{X}_n	$X_1\bar{X}_2$	$X_1\bar{X}_3$...	$\bar{X}_{n-1}\bar{X}_n$...	$X_1 \dots \bar{X}_{n-1} \bar{X}_n$
...
$2^n - 2$	X_1	X_2	...	\bar{X}_n	X_1X_2	X_1X_3	...	$X_{n-1}\bar{X}_n$...	$X_1 \dots X_{n-1} \bar{X}_n$
$2^n - 1$	X_1	X_2	...	X_n	X_1X_2	X_1X_3	...	$X_{n-1}X_n$...	$X_1 \dots X_{n-1} X_n$

Таким образом, всего в таблице 3.14 2^n строк, столько же, сколько в таблице истинности для формулы от n переменных.

В первом столбце для удобства изложения будем записывать двоичный номер строки — номер набора (3.11), далее столбцы карты заполняют сначала одиночными литералами, соответствующими набору (3.11): $X_1^{\alpha_1}, \dots, X_n^{\alpha_n}$ — их всего C_n^1 .

Затем добавляют всевозможные конъюнкции из двух различных литералов: $X_1^{\alpha_1} X_2^{\alpha_2}, \dots, X_{n-1}^{\alpha_{n-1}} X_n^{\alpha_n}$ — их всего C_n^2 .

Далее — все конъюнкции из трех различных литералов и т. д.

В последний столбец записывают конститuentу единицы набора (3.11) — $X_1^{\alpha_1} X_2^{\alpha_2} \dots X_n^{\alpha_n}$. Всего в таблице 3.14

$$C_n^1 + C_n^2 + \dots + C_n^n = 2^n - 1$$

столбцов. При этом размеры и содержание карты *не зависят* от вида формулы и определяются только количеством ее переменных.

Строки таблицы 3.14 будем записывать в обычном лексикографическом порядке, т. е. в порядке возрастания номеров булевых наборов (3.11).

Теорема 3.14. Если элементарная конъюнкция $X_1^{\alpha_1} \dots X_n^{\alpha_n}$, принадлежащая j -й строке (см. таблицу 3.14), не входит в СДНФ, выражающую формулу $F(X_1, \dots, X_n)$, то любая конъюнкция j -й строки не входит ни в какую ДНФ, выражающую исходную формулу F .

Доказательство. Действительно, если конъюнкция $X_1^{\alpha_1} \dots X_n^{\alpha_n}$ не входит в СДНФ, выражающую формулу $F(X_1, \dots, X_n)$, то согласно второму алгоритму построения СДНФ $F(\alpha_1, \dots, \alpha_n) = 0$. Если бы какая-то конъюнкция j -й строки вошла в некоторую ДНФ, выражающую формулу F , то $F(\alpha_1, \dots, \alpha_n) = 1$. ■

Согласно теореме 3.14 опишем способ построения минимальной ДНФ.

Алгоритм 3.6. Метод минимизирующих карт

- 1: Отмечаем в таблице 3.14 строки, в которых соответствующая им конъюнкция последнего столбца $X_1^{\alpha_1} \dots X_n^{\alpha_n}$ не принадлежит СДНФ.
- 2: Вычеркиваем *все* конъюнкции в этих строках.
- 3: Удаленные конъюнкции вычеркиваем также во всех остальных строках таблицы 3.14.
- 4: Составляем ДНФ, выбирая из каждой оставшейся строки таблицы 3.14 по одной конъюнкции.
- 5: Из всех ДНФ, полученных на предыдущем шаге, выбираем минимальную.

Замечание 3.7

Заметим, что на последнем шаге алгоритма предусматривается перебор различных ДНФ для нахождения минимальной из них. Однако при этом вариантов перебора, как правило, значительно меньше, чем в случае, когда перебираются все равносильные ДНФ.

Стратегии перебора пунктов 4–5 алгоритма 3.6 могут быть различными. В следующих примерах будем начинать выбор со строк, имеющих по одному представителю (без учета полной конъюнкции последнего столбца).

Пример 3.30. Пусть

$$F(X_1, X_2, X_3) = X_1 X_2 \bar{X}_3 \vee X_1 \bar{X}_2 X_3 \vee X_1 \bar{X}_2 \bar{X}_3 \vee \bar{X}_1 \bar{X}_2 X_3.$$

Составим соответствующую таблицу (таблица 3.15).

Таблица 3.15.

0	\bar{X}_1	\bar{X}_2	\bar{X}_3	$\bar{X}_1 \bar{X}_2$	$\bar{X}_1 \bar{X}_3$	$\bar{X}_2 \bar{X}_3$	$\bar{X}_1 \bar{X}_2 \bar{X}_3$
1	\bar{X}_1	\bar{X}_2	X_3	$\bar{X}_1 \bar{X}_2$	$\bar{X}_1 X_3$	$\bar{X}_2 X_3$	$\bar{X}_1 \bar{X}_2 X_3$
2	\bar{X}_1	X_2	\bar{X}_3	$\bar{X}_1 X_2$	$\bar{X}_1 \bar{X}_3$	$X_2 \bar{X}_3$	$\bar{X}_1 X_2 \bar{X}_3$
3	\bar{X}_1	X_2	X_3	$\bar{X}_1 X_2$	$\bar{X}_1 X_3$	$X_2 X_3$	$\bar{X}_1 X_2 X_3$
4	X_1	\bar{X}_2	\bar{X}_3	$X_1 \bar{X}_2$	$X_1 \bar{X}_3$	$\bar{X}_2 \bar{X}_3$	$X_1 \bar{X}_2 \bar{X}_3$
5	X_1	\bar{X}_2	X_3	$X_1 \bar{X}_2$	$X_1 X_3$	$\bar{X}_2 X_3$	$X_1 \bar{X}_2 X_3$
6	X_1	X_2	\bar{X}_3	$X_1 X_2$	$X_1 \bar{X}_3$	$X_2 \bar{X}_3$	$X_1 X_2 \bar{X}_3$
7	X_1	X_2	X_3	$X_1 X_2$	$X_1 X_3$	$X_2 X_3$	$X_1 X_2 X_3$

Отметим серым цветом вычеркнутые строки таблицы. После применения шагов 1–3 алгоритма 3.6 таблица примет следующий вид (таблица 3.16).

Таблица 3.16.

0							
1						$\bar{X}_2 X_3$	
2							
3							
4			$X_1 \bar{X}_2$	$X_1 \bar{X}_3$			
5			$X_1 \bar{X}_2$			$\bar{X}_2 X_3$	
6				$X_1 \bar{X}_3$			
7							

При выборе конъюнкций будем придерживаться стратегии замечания 3.7. Начнем выбор со строк, имеющих по одному представителю.

В строке 1 это конъюнкция $\bar{X}_2 X_3$. Она представляет строки 1 и 5.

Далее из строки 6 берем единственного представителя — конъюнкцию $X_1 \bar{X}_3$. Она представляет строки 6 и 4.

Все невычеркнутые строки имеют представителя в ДНФ, алгоритм заканчивает работу, получаем минимальную ДНФ, равносильную исходной формуле:

$$F(X_1, X_2, X_3) = X_1 \bar{X}_3 \vee \bar{X}_2 X_3.$$

Пример 3.31. Построим минимальную ДНФ для формулы, заданной в виде СДНФ:

$$F(X_1, X_2, X_3) = \bar{X}_1 \bar{X}_2 \bar{X}_3 \vee \bar{X}_1 X_2 \bar{X}_3 \vee \bar{X}_1 X_2 X_3 \vee X_1 X_2 \bar{X}_3 \vee X_1 X_2 X_3.$$

Строим для данной формулы минимизирующую карту и работаем с ней по алгоритму 3.6. Отметим справа от последнего столбца те конъюнкции, которые входят в СДНФ данной формулы. Вычеркнем неотмеченные строки, затем вычеркнем в остальных строках (действуя по столбцам) те элементы, которые попали в вычеркнутые строки.

После применения алгоритма карта примет следующий вид (таблица 3.17). Вычеркнутые конъюнкции отмечены светло-серым цветом, оставшиеся — темно-серым (конъюнкции последнего столбца не отмечаем при наличии других кандидатов).

Как и в предыдущем примере 3.30, при выборе конъюнкций будем придерживаться стратегии замечания 3.7. Начнем выбор со строк, имеющих по одному представителю.

В строке 0 это конъюнкция $\bar{X}_1 \bar{X}_3$. Она представляет строки 0 и 2.

Таблица 3.17.

0	\bar{X}_1	\bar{X}_2	\bar{X}_3	$\bar{X}_1\bar{X}_2$	$\bar{X}_1\bar{X}_3$	$\bar{X}_2\bar{X}_3$	$\bar{X}_1\bar{X}_2\bar{X}_3$
1	\bar{X}_1	\bar{X}_2	X_3	$\bar{X}_1\bar{X}_2$	\bar{X}_1X_3	\bar{X}_2X_3	$\bar{X}_1\bar{X}_2X_3$
2	\bar{X}_1	X_2	\bar{X}_3	\bar{X}_1X_2	$\bar{X}_1\bar{X}_3$	$X_2\bar{X}_3$	$\bar{X}_1X_2\bar{X}_3$
3	\bar{X}_1	X_2	X_3	\bar{X}_1X_2	\bar{X}_1X_3	X_2X_3	$\bar{X}_1X_2X_3$
4	X_1	\bar{X}_2	\bar{X}_3	$X_1\bar{X}_2$	$X_1\bar{X}_3$	$\bar{X}_2\bar{X}_3$	$X_1\bar{X}_2\bar{X}_3$
5	X_1	\bar{X}_2	X_3	$X_1\bar{X}_2$	X_1X_3	\bar{X}_2X_3	$X_1\bar{X}_2X_3$
6	X_1	X_2	\bar{X}_3	X_1X_2	$X_1\bar{X}_3$	$X_2\bar{X}_3$	$X_1X_2\bar{X}_3$
7	X_1	X_2	X_3	X_1X_2	X_1X_3	X_2X_3	$X_1X_2X_3$

Далее из строки 3 берем самую короткую конъюнкцию X_2 . Она представляет строки 3, 6 и 7 (строка 2 уже представлена ранее).

Все невычеркнутые строки имеют представителя в ДНФ, алгоритм заканчивает работу.

Таким образом, получаем

$$F(X_1, X_2, X_3) = X_2 \vee \bar{X}_1\bar{X}_3.$$

Пример 3.32. Построим минимальную ДНФ для формулы, заданной в виде СДНФ:

$$F(X_1, X_2, X_3) = \bar{X}_1\bar{X}_2\bar{X}_3 \vee \bar{X}_1X_2\bar{X}_3 \vee X_1\bar{X}_2\bar{X}_3 \vee X_1X_2X_3$$

Как и в примере 3.31, строим для данной формулы минимизирующую карту и применяем алгоритм 3.6. Карта примет следующий вид (таблица 3.18).

Как и в предыдущих примерах, начинаем выбор со строк, имеющих по одному представителю. В строке 2 это конъюнкция $\bar{X}_1\bar{X}_3$. Она представляет строки 2 и 0.

Далее из строки 4 берем конъюнкцию $\bar{X}_2\bar{X}_3$, а из строки 7 — конъюнкцию $X_1X_2X_3$.

Таким образом, $F(X_1, X_2, X_3) = \bar{X}_1\bar{X}_3 \vee \bar{X}_2\bar{X}_3 \vee X_1X_2X_3$.

Пример 3.33. Рассмотрим пример работы алгоритма 3.6 для размерности $n = 4$. Построим минимальную ДНФ для формулы, заданной в виде

Таблица 3.18.

0	\bar{X}_1	\bar{X}_2	\bar{X}_3	$\bar{X}_1\bar{X}_2$	$\bar{X}_1\bar{X}_3$	$\bar{X}_2\bar{X}_3$	$\bar{X}_1\bar{X}_2\bar{X}_3$
1	\bar{X}_1	\bar{X}_2	X_3	$\bar{X}_1\bar{X}_2$	\bar{X}_1X_3	\bar{X}_2X_3	$\bar{X}_1\bar{X}_2X_3$
2	\bar{X}_1	X_2	\bar{X}_3	\bar{X}_1X_2	$\bar{X}_1\bar{X}_3$	$X_2\bar{X}_3$	$\bar{X}_1X_2\bar{X}_3$
3	\bar{X}_1	X_2	X_3	\bar{X}_1X_2	\bar{X}_1X_3	X_2X_3	$\bar{X}_1X_2X_3$
4	X_1	\bar{X}_2	\bar{X}_3	$X_1\bar{X}_2$	$X_1\bar{X}_3$	$\bar{X}_2\bar{X}_3$	$X_1\bar{X}_2\bar{X}_3$
5	X_1	\bar{X}_2	X_3	$X_1\bar{X}_2$	X_1X_3	\bar{X}_2X_3	$X_1\bar{X}_2X_3$
6	X_1	X_2	\bar{X}_3	X_1X_2	$X_1\bar{X}_3$	$X_2\bar{X}_3$	$X_1X_2\bar{X}_3$
7	X_1	X_2	X_3	X_1X_2	X_1X_3	X_2X_3	$X_1X_2X_3$

СДНФ:

$$F(X_1, X_2, X_3, X_4) = \bar{X}_1\bar{X}_2X_3X_4 \vee \bar{X}_1X_2\bar{X}_3\bar{X}_4 \vee \bar{X}_1X_2\bar{X}_3X_4 \vee \bar{X}_1X_2X_3X_4 \vee X_1\bar{X}_2\bar{X}_3X_4 \vee X_1X_2\bar{X}_3X_4 \vee X_1X_2X_3\bar{X}_4 \vee X_1X_2X_3X_4.$$

Таблица 3.19.

0	\bar{X}_1	\bar{X}_2	\bar{X}_3	\bar{X}_4	$\bar{X}_1\bar{X}_2$	$\bar{X}_1\bar{X}_3$	$\bar{X}_1\bar{X}_4$	$\bar{X}_2\bar{X}_3$
	$\bar{X}_2\bar{X}_4$	$\bar{X}_3\bar{X}_4$	$\bar{X}_1\bar{X}_2\bar{X}_3$	$\bar{X}_1\bar{X}_2\bar{X}_4$	$\bar{X}_1\bar{X}_3\bar{X}_4$	$\bar{X}_2\bar{X}_3\bar{X}_4$	$\bar{X}_1\bar{X}_2\bar{X}_3\bar{X}_4$	
1	\bar{X}_1	\bar{X}_2	\bar{X}_3	X_4	$\bar{X}_1\bar{X}_2$	$\bar{X}_1\bar{X}_3$	\bar{X}_1X_4	$\bar{X}_2\bar{X}_3$
	\bar{X}_2X_4	\bar{X}_3X_4	$\bar{X}_1\bar{X}_2\bar{X}_3$	$\bar{X}_1\bar{X}_2X_4$	$\bar{X}_1\bar{X}_3X_4$	$\bar{X}_2\bar{X}_3X_4$	$\bar{X}_1\bar{X}_2\bar{X}_3X_4$	
2	\bar{X}_1	\bar{X}_2	X_3	\bar{X}_4	$\bar{X}_1\bar{X}_2$	\bar{X}_1X_3	$\bar{X}_1\bar{X}_4$	\bar{X}_2X_3
	$\bar{X}_2\bar{X}_4$	$X_3\bar{X}_4$	$\bar{X}_1\bar{X}_2X_3$	$\bar{X}_1\bar{X}_2\bar{X}_4$	$\bar{X}_1X_3\bar{X}_4$	$\bar{X}_2X_3\bar{X}_4$	$\bar{X}_1\bar{X}_2X_3\bar{X}_4$	
3	\bar{X}_1	\bar{X}_2	X_3	X_4	$\bar{X}_1\bar{X}_2$	\bar{X}_1X_3	\bar{X}_1X_4	\bar{X}_2X_3
	\bar{X}_2X_4	X_3X_4	$\bar{X}_1\bar{X}_2X_3$	$\bar{X}_1\bar{X}_2X_4$	$\bar{X}_1X_3X_4$	$\bar{X}_2X_3X_4$	$\bar{X}_1\bar{X}_2X_3X_4$	
4	\bar{X}_1	X_2	\bar{X}_3	\bar{X}_4	\bar{X}_1X_2	$\bar{X}_1\bar{X}_3$	$\bar{X}_1\bar{X}_4$	$X_2\bar{X}_3$
	$X_2\bar{X}_4$	$\bar{X}_3\bar{X}_4$	$\bar{X}_1X_2\bar{X}_3$	$\bar{X}_1X_2\bar{X}_4$	$\bar{X}_1\bar{X}_3\bar{X}_4$	$X_2\bar{X}_3\bar{X}_4$	$\bar{X}_1X_2\bar{X}_3\bar{X}_4$	
5	\bar{X}_1	X_2	\bar{X}_3	X_4	\bar{X}_1X_2	$\bar{X}_1\bar{X}_3$	\bar{X}_1X_4	$X_2\bar{X}_3$
	X_2X_4	\bar{X}_3X_4	$\bar{X}_1X_2\bar{X}_3$	$\bar{X}_1X_2X_4$	$\bar{X}_1\bar{X}_3X_4$	$X_2\bar{X}_3X_4$	$\bar{X}_1X_2\bar{X}_3X_4$	
6	\bar{X}_1	X_2	X_3	\bar{X}_4	\bar{X}_1X_2	\bar{X}_1X_3	$\bar{X}_1\bar{X}_4$	X_2X_3
	$X_2\bar{X}_4$	$X_3\bar{X}_4$	$\bar{X}_1X_2X_3$	$\bar{X}_1X_2\bar{X}_4$	$\bar{X}_1X_3\bar{X}_4$	$X_2X_3\bar{X}_4$	$\bar{X}_1X_2X_3\bar{X}_4$	

Продолжение таблицы 3.19

7	\bar{X}_1	X_2	X_3	X_4	$\bar{X}_1 X_2$	$\bar{X}_1 X_3$	$\bar{X}_1 X_4$	$X_2 X_3$
	$X_2 X_4$	$X_3 X_4$	$\bar{X}_1 X_2 X_3$	$\bar{X}_1 X_2 X_4$	$\bar{X}_1 X_3 X_4$	$X_2 X_3 X_4$	$\bar{X}_1 X_2 X_3 X_4$	
8	X_1	\bar{X}_2	\bar{X}_3	\bar{X}_4	$X_1 \bar{X}_2$	$X_1 \bar{X}_3$	$X_1 \bar{X}_4$	$\bar{X}_2 \bar{X}_3$
	$\bar{X}_2 \bar{X}_4$	$\bar{X}_3 \bar{X}_4$	$X_1 \bar{X}_2 \bar{X}_3$	$X_1 \bar{X}_2 \bar{X}_4$	$X_1 \bar{X}_3 \bar{X}_4$	$\bar{X}_2 \bar{X}_3 \bar{X}_4$	$X_1 \bar{X}_2 \bar{X}_3 \bar{X}_4$	
9	X_1	\bar{X}_2	\bar{X}_3	X_4	$X_1 \bar{X}_2$	$X_1 \bar{X}_3$	$X_1 X_4$	$\bar{X}_2 \bar{X}_3$
	$\bar{X}_2 X_4$	$\bar{X}_3 X_4$	$X_1 \bar{X}_2 \bar{X}_3$	$X_1 \bar{X}_2 X_4$	$X_1 \bar{X}_3 X_4$	$\bar{X}_2 \bar{X}_3 X_4$	$X_1 \bar{X}_2 \bar{X}_3 X_4$	
10	X_1	\bar{X}_2	X_3	\bar{X}_4	$X_1 \bar{X}_2$	$X_1 X_3$	$X_1 \bar{X}_4$	$\bar{X}_2 X_3$
	$\bar{X}_2 \bar{X}_4$	$X_3 \bar{X}_4$	$X_1 \bar{X}_2 X_3$	$X_1 \bar{X}_2 \bar{X}_4$	$X_1 X_3 \bar{X}_4$	$\bar{X}_2 X_3 \bar{X}_4$	$X_1 \bar{X}_2 X_3 \bar{X}_4$	
11	X_1	\bar{X}_2	X_3	X_4	$X_1 \bar{X}_2$	$X_1 X_3$	$X_1 X_4$	$\bar{X}_2 X_3$
	$\bar{X}_2 X_4$	$X_3 X_4$	$X_1 \bar{X}_2 X_3$	$X_1 \bar{X}_2 X_4$	$X_1 X_3 X_4$	$\bar{X}_2 X_3 X_4$	$X_1 \bar{X}_2 X_3 X_4$	
12	X_1	X_2	\bar{X}_3	\bar{X}_4	$X_1 X_2$	$X_1 \bar{X}_3$	$X_1 \bar{X}_4$	$X_2 \bar{X}_3$
	$X_2 \bar{X}_4$	$\bar{X}_3 \bar{X}_4$	$X_1 X_2 \bar{X}_3$	$X_1 X_2 \bar{X}_4$	$X_1 \bar{X}_3 \bar{X}_4$	$X_2 \bar{X}_3 \bar{X}_4$	$X_1 X_2 \bar{X}_3 \bar{X}_4$	
13	X_1	X_2	\bar{X}_3	X_4	$X_1 X_2$	$X_1 \bar{X}_3$	$X_1 X_4$	$X_2 \bar{X}_3$
	$X_2 X_4$	$\bar{X}_3 X_4$	$X_1 X_2 \bar{X}_3$	$X_1 X_2 X_4$	$X_1 \bar{X}_3 X_4$	$X_2 \bar{X}_3 X_4$	$X_1 X_2 \bar{X}_3 X_4$	
14	X_1	X_2	X_3	\bar{X}_4	$X_1 X_2$	$X_1 X_3$	$X_1 \bar{X}_4$	$X_2 X_3$
	$X_2 \bar{X}_4$	$X_3 \bar{X}_4$	$X_1 X_2 X_3$	$X_1 X_2 \bar{X}_4$	$X_1 X_3 \bar{X}_4$	$X_2 X_3 \bar{X}_4$	$X_1 X_2 X_3 \bar{X}_4$	
15	X_1	X_2	X_3	X_4	$X_1 X_2$	$X_1 X_3$	$X_1 X_4$	$X_2 X_3$
	$X_2 X_4$	$X_3 X_4$	$X_1 X_2 X_3$	$X_1 X_2 X_4$	$X_1 X_3 X_4$	$X_2 X_3 X_4$	$X_1 X_2 X_3 X_4$	

Как и в предыдущих примерах, начинаем выбор со строк, имеющих по одному представителю (таблица 3.19).

В строке 3 это конъюнкция $\bar{X}_1 X_3 X_4$. Она представляет строки 3 и 7.

В строке 4 это конъюнкция $\bar{X}_1 X_2 \bar{X}_3$, представляющая строки 4 и 5.

В строке 9 это конъюнкция $X_1 \bar{X}_3 X_4$, «закрывающая» строки 9 и 13.

В строке 14 это конъюнкция $X_1 X_2 X_3$. Она представляет строки 14 и 15.

Таким образом,

$$F(X_1, X_2, X_3, X_4) = \bar{X}_1 X_3 X_4 \vee \bar{X}_1 X_2 \bar{X}_3 \vee X_1 \bar{X}_3 X_4 \vee X_1 X_2 X_3.$$

Заметим, что самая короткая конъюнкция $X_2 X_4$, представляющая строки 5, 7, 13, 15, в решение не вошла.

3.1.8. Метод Куайна — Мак-Класки

Рассмотрим еще один метод построения минимальной ДНФ с помощью сокращения множества перебираемых элементарных конъюнкций. Метод основан на очевидных тождествах:

$$XY \vee X\bar{Y} = X, \quad (X \vee Y)(X \vee \bar{Y}) = X.$$

Введем необходимые определения.

Определение 3.23. Говорят, что элементарная конъюнкция K *покрывает* элементарную конъюнкцию L (обозначается $K \succ L$), если любой литерал из K входит в L . Например, $X_1X_3 \succ X_1\bar{X}_2X_3$.

Далее будем использовать кодовое расстояние Хемминга, введенное в разделе 1.6.4 (определение 1.43). Расстояние между булевыми векторами a и b есть число несовпадающих позиций наборов, обозначается как $d(a, b)$.

Определение 3.24. Пусть для двух булевых векторов $d(a, b) = 1$. По определению 1.43 они отличаются в одной позиции. Эту позицию заменим символом $*$. Таковую операцию назовем *склеивкой*, а получившийся вектор — *импликантом*. Будем обозначать его, как $i(a, b)$

Например, векторы $a = \{01011\}$ и $b = \{11011\}$ склеиваются в импликант $i(a, b) = \{*1011\}$.

Далее символ $*$ будем учитывать при вычислении расстояния Хемминга.

Алгоритм 3.7. Метод Куайна — Мак-Класки

```
// Инициализация:
// формулу для минимизации представляем в виде СДНФ
F(X1, ..., Xn) = K1 ∨ ... ∨ Kp
l ← 1 // счетчик уровней

// каждую конъюнкцию представляем двоичным вектором литералов
for j ← 1, p do
    Kj = Xα1 ... Xαn → {α1 ... αn} = vj // например: X1 $\bar{X}_2$ X3 $\bar{X}_4$  → {1010}
end for

// Первый этап алгоритма
repeat
    m ← 1 // счетчик импликант на уровне l
    while существуют вектора vil, vjl | d(vil, vjl) = 1 do
        vml+1 ← i(vil, vjl)
```

```

    m ← m + 1
  end while
  l ← l + 1 // переход на следующий уровень
until m ≥ 2

// Второй этап алгоритма
Выделяем простые импликанты (не участвовавшие ни в одной склейке)

// Строим матрицу для минимизации:
столбцы матрицы — двоичные векторы исходной СДНФ:  $v_j^1, j \in 1 : p$ .
строки матрицы — простые импликанты.
В каждом столбце отмечаем те строки, которые покрывают вектор столбца.
// Начинаем решение задачи с выбора столбцов, содержащих единственную от-
метку
Выбираем минимальное число строк, покрывающих все столбцы (определение 3.23),

```

Замечание 3.8

Изначально алгоритм был предложен Куайном. Модификация Мак-Класки заключалась в разбиении двоичных векторов v_j^1 на непересекающиеся множества (классы) по числу единиц, имеющихся в их двоичной записи (например, $\{1010\} \rightarrow$ класс 2), чтобы исключить сравнения, заведомо не дающие склеивания.

Рассмотрим работу алгоритма 3.7 на формуле из примера 3.33. Простые импликанты отмечены серым цветом.

Пример 3.34. Напомним, что формула F имеет вид:

$$F(X_1, X_2, X_3, X_4) = \overline{X_1}\overline{X_2}X_3X_4 \vee \overline{X_1}X_2\overline{X_3}\overline{X_4} \vee \overline{X_1}X_2\overline{X_3}X_4 \vee \overline{X_1}X_2X_3\overline{X_4} \vee X_1\overline{X_2}\overline{X_3}X_4 \vee X_1X_2\overline{X_3}X_4 \vee X_1X_2X_3\overline{X_4} \vee X_1X_2X_3X_4.$$

После шага 1 получаем множество двоичных векторов:

$$0011, 0100, 0101, 0111, 1001, 1101, 1110, 1111.$$

Разбиваем полученные двоичные наборы на четыре класса:

$$\underbrace{(0100)}_1 \quad \underbrace{(0011, 0101, 1001)}_2 \quad \underbrace{(0111, 1101, 1110)}_3 \quad \underbrace{(1111)}_4$$

Процесс склеивания векторов представлен на рисунке 3.6.

На следующем этапе строим матрицу для минимизации (таблица 3.20). Покрытие столбцов отмечено серым цветом.

Необходимо выбрать минимальное число строк, покрывающих все столбцы. Начинаем с выбора столбцов, содержащих единственную отметку. Такими являются столбцы 0011, 0100, 1001, 1110. Имеем:

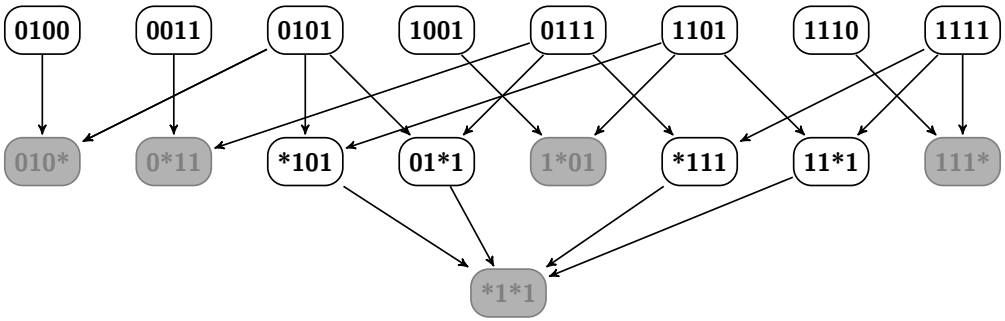


Рис. 3.6.

Таблица 3.20.

	0011	0100	0101	0111	1001	1101	1110	1111
010*								
0*11								
1*01								
111*								
*1*1								

- 1) 0*11, покрываем 0011 и 0111;
- 2) 010*, покрываем 0101 и 0100;
- 3) 1*01, покрываем 1001 и 1101;
- 4) 111*, покрываем 1110 и 1111.

Таким образом:

$$F(X_1, X_2, X_3, X_4) = \bar{X}_1 X_3 X_4 \vee \bar{X}_1 X_2 \bar{X}_3 \vee X_1 \bar{X}_3 X_4 \vee X_1 X_2 X_3.$$

Полученная минимальная ДНФ совпадает с результатом примера 3.33. Как и в примере 3.33, самая короткая конъюнкция $X_2 X_4$, покрывающая наибольшее число исходных конъюнкций, в решение не вошла.

Замечание 3.9

Время работы метода растет экспоненциально с увеличением входных данных. Можно показать, что для функции от n переменных верхняя граница количества основных импликант $3^n/n$. Если $n = 32$, их может быть больше чем $6.5 \cdot 10^{15}$.

Пример 3.35. Построим минимальную ДНФ для функции:

$$F(X_1, X_2, X_3, X_4) = \bar{X}_1 \bar{X}_2 X_3 X_4 \vee \bar{X}_1 X_2 \bar{X}_3 \bar{X}_4 \vee \bar{X}_1 X_2 \bar{X}_3 X_4 \vee \bar{X}_1 X_2 X_3 X_4 \vee X_1 \bar{X}_2 \bar{X}_3 X_4 \vee X_1 \bar{X}_2 X_3 X_4 \vee X_1 X_2 \bar{X}_3 \bar{X}_4 \vee X_1 X_2 \bar{X}_3 X_4.$$

Получаем множество двоичных векторов:

0011, 0100, 0101, 0111, 1001, 1011, 1100, 1101.

Проводим склеивание векторов (рисунок 3.7).

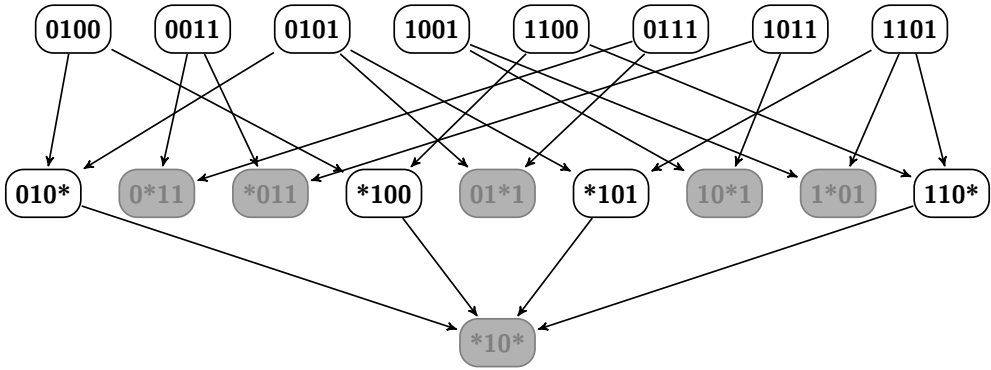


Рис. 3.7.

Матрица для минимизации имеет вид (таблица 3.21).

Таблица 3.21.

	0100	0011	0101	1001	1100	0111	1011	1101
0*11								
*011								
01*1								
10*1								
1*01								
10								

Начинаем с выбора столбцов, содержащих минимальное число отметок (в данном случае одну). Это столбец 0100. Имеем:

- 1) *10*, покрываем 0100, 0101, 1100 и 1101;
- 2) 0*11, покрываем 0011 и 0111;
- 3) 10*1, покрываем 1001 и 1011.

Таким образом:

$$F(X_1, X_2, X_3, X_4) = X_2\bar{X}_3 \vee \bar{X}_1X_3X_4 \vee X_1\bar{X}_2X_4.$$

Пример 3.36. Построим минимальную ДНФ для функции:

$$F(X_1, X_2, X_3, X_4) = \bar{X}_1\bar{X}_2X_3\bar{X}_4 \vee \bar{X}_1X_2\bar{X}_3X_4 \vee \bar{X}_1X_2X_3\bar{X}_4 \vee \bar{X}_1X_2X_3X_4 \vee X_1\bar{X}_2X_3\bar{X}_4 \vee X_1\bar{X}_2\bar{X}_3\bar{X}_4 \vee X_1X_2\bar{X}_3\bar{X}_4 \vee X_1X_2\bar{X}_3X_4.$$

Множество двоичных векторов имеет вид:

0010, 0101, 0110, 0111, 1010, 1100, 1101, 1110.

Далее определяем простые импликанты (рисунок 3.8).

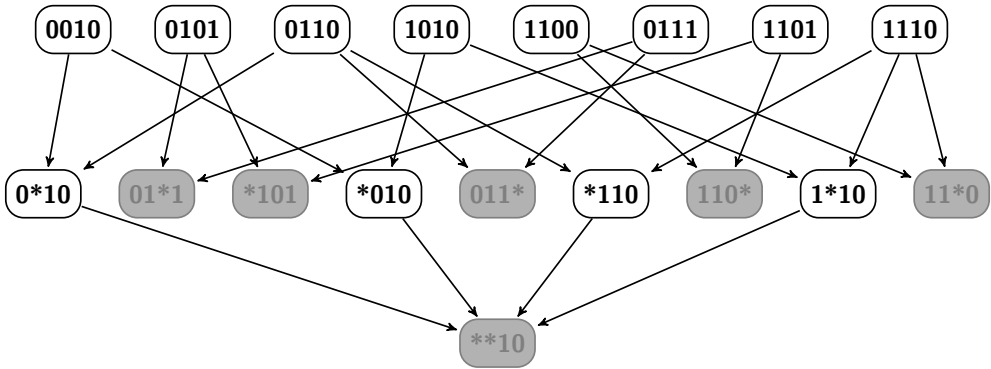


Рис. 3.8.

Составляем матрицу для минимизации (таблица 3.22).

Таблица 3.22.

	0010	0101	0110	1010	1100	0111	1101	1110
01*1								
*101								
011*								
110*								
11*0								
**10								

Начинаем с выбора столбцов, содержащих минимальное число отметок (в данном случае одну). Это столбец 0010. Имеем:

- 1) $**10$, покрываем 0010 , 0110 , 1010 и 1110 ;
- 2) $01*1$, покрываем 0101 и 0111 ;
- 3) $110*$, покрываем 1100 и 1101 .

Таким образом:

$$F(X_1, X_2, X_3, X_4) = X_3\bar{X}_4 \vee \bar{X}_1X_2X_4 \vee X_1X_2\bar{X}_3.$$

Задача 3.4. Используя метод минимизирующих карт и метод Куайна — Мак-Класки, постройте минимальную ДНФ для формулы

$$F(X_1, X_2, X_3, X_4) = \overline{(X_1 \vee X_2 \bar{X}_3)} \overline{(X_2 \rightarrow X_3)} X_4.$$

Замечание 3.10

По вышеизложенным причинам ряд известных методов минимизации, таких как методы Блейка, карт Карнапа, остались вне рамок данной книги. Более подробно познакомиться с этой темой можно, например, в [60].

3.1.9. Метод резолюций в логике высказываний

Рассмотрим еще один метод доказательства того, что формула G является логическим следствием формул F_1, \dots, F_k , который называют *методом резолюций*. Особенность метода состоит в том, что он оперирует не с произвольными формулами, а с дизъюнктами.

Замечание 3.11

Условимся не различать дизъюнкты, которые получаются один из другого с помощью равносильностей, установленных в теореме 3.4: коммутативности и ассоциативности дизъюнкции (законы 4 и 6), а также идемпотентности (закон 2). Последнее, например, означает что дизъюнкты $X \vee \bar{Y} \vee X \vee Z$ и $Z \vee X \vee \bar{Y}$ равны.

Введем в рассмотрение *пустой* дизъюнкт, т. е. дизъюнкт, не содержащий литералов. В методе резолюций его обычно обозначают \square . Считаем, что пустой дизъюнкт ложен при любой интерпретации, т. е. равен константе 0. Например, формула $F \wedge \square$ равносильна \square , а формула $F \vee \square$ равносильна F .

Определение 3.25. Литералы L и \bar{L} называют *противоположными*, или *контрарными*.

Определение 3.26. *Правилом резолюций* в логике высказываний называют следующее правило: из дизъюнктов $X \vee F$ и $\bar{X} \vee G$ выводим дизъюнкт $F \vee G$.

Дизъюнкт $F \vee G$ называют *резольвентой* дизъюнктов $X \vee F$ и $\bar{X} \vee G$.

Пример 3.37. Из дизъюнктов $\bar{X} \vee Y \vee Z$ и $X \vee \bar{Y}$ выводим дизъюнкт $Y \vee Z \vee \bar{Y}$.

Замечание 3.12

Заметим, что в первых двух дизъюнктах последнего примера есть еще одна пара противоположных литералов.

Применять правило резолюций не обязательно к самым левым литералам (поскольку не различаются дизъюнкты, отличающиеся порядком записи литералов). Тогда правило резолюций, примененное к литералам Y и \bar{Y} этих дизъюнктов, даст $\bar{X} \vee Z \vee X$.

В дизъюнктах не будем писать повторяющиеся литералы и пустой дизъюнкт \square , если есть другие литералы.

Лемма 3.4. Резольвента есть логическое следствие породивших ее дизъюнктов.

Доказательство. Рассмотрим дизъюнкты $X \vee F$ и $\bar{X} \vee G$. Составим совместную таблицу истинности дизъюнктов и резольвенты (таблица 3.23).

Таблица 3.23.

X	F	G	$X \vee F$	$\bar{X} \vee G$	$F \vee G$	X	F	G	$X \vee F$	$\bar{X} \vee G$	$F \vee G$
0	0	0	0	1	0	1	0	0	1	0	0
0	0	1	0	1	1	1	0	1	1	1	1
0	1	0	1	1	1	1	1	0	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1

Согласно определению 3.9 формула $F \vee G$ есть логическое следствие формул $X \vee F$ и $\bar{X} \vee G$ ■

Определение 3.27. Пусть S — множество дизъюнктов. *Резолютивным выводом*, или просто *выводом* из S , называется последовательность дизъюнктов D_1, D_2, \dots, D_n такая, что каждый дизъюнкт этой последовательности принадлежит S или следует из предыдущих по правилу резолюций.

Дизъюнкт D выводим из S , если существует вывод из S , последним дизъюнктом которого является D .

Вывод пустого дизъюнкта из S называют *опровержением* S .

Пример 3.38. Пусть $S = \{\bar{X} \vee Y, Z \vee X \vee \bar{Y}, Y\}$, тогда последовательность

$$D_1 = Z \vee X \vee \bar{Y}, D_2 = Y, D_3 = Z \vee X, D_4 = \bar{X} \vee Y, D_5 = Y \vee Z$$

является выводом из S . Дизъюнкт $Y \vee Z$ выводим из S .

Определение 3.28. Дизъюнкт \widehat{D} *зависит* от дизъюнкта D , если при выводе \widehat{D} использовалась резольвента D .

Применение метода резолюций основано на следующем утверждении.

Теорема 3.15 (о полноте метода резолюций). Множество дизъюнктов логики высказываний S невыполнимо тогда и только тогда, когда из S выводим пустой дизъюнкт.

Доказательство. Пусть из S выводим пустой дизъюнкт. Предположим противное: множество S выполнимо, т. е. существует интерпретация, при которой все дизъюнкты из S истинны.

Рассмотрим вывод пустого дизъюнкта $D_1, \dots, D_n = \square$. Если дизъюнкт $D_j \in S$, то по предположению на этой интерпретации $D_j = 1$. Если же он получается из предыдущих по правилу резолюций, то согласно лемме 3.4 также $D_j = 1$. При $j = n$ $\square = 1$. Противоречие доказывает достаточность.

Докажем необходимость. Доказательство проведем индукцией по следующему параметру: $d(S)$ есть сумма числа вхождений литералов в дизъюнкты из S минус число дизъюнктов.

Пусть множество дизъюнктов S невыполнимо. Если пустой дизъюнкт принадлежит S , то он, очевидно, выводим из S . Будем считать, что $\square \notin S$. Тогда каждый дизъюнкт из S содержит хотя бы один литерал, следовательно, $d \geq 0$.

База индукции: $d(S) = 0$. Тогда все дизъюнкты состоят из одного литерала. Поскольку множество S невыполнимо, то в нем должна найтись пара контрарных литералов, например, X и \bar{X} . В этом случае очевидно, что пустой дизъюнкт выводим из S .

Шаг индукции: $d(S) > 0$. Предположим, что для любого множества дизъюнктов \tilde{S} такого, что $d(\tilde{S}) < d(S)$, необходимость теоремы доказана. Пусть $S = \{D_1, \dots, D_k\}$. Так как $d(S) > 0$, в S существует хотя бы один дизъюнкт, содержащий более одного литерала. Не умаляя общности, считаем, что это дизъюнкт $D_k = L_k \vee F_k$, где L_k — литерал и $F_k \neq \square$.

Рассмотрим два множества дизъюнктов:

$$S_1 = S \setminus \{D_k\} \cup \{L_k\}, \quad S_2 = S \setminus \{D_k\} \cup \{F_k\}.$$

Из невыполнимости S очевидно, что S_1 и S_2 также невыполнимы и при этом $d(S_1) < d(S)$ и $d(S_2) < d(S)$. Тогда по индукционному предположению из S_1 и S_2 выводим пустой дизъюнкт. Запишем выводы пустого дизъюнкта

из S_1 и S_2 соответственно

$$A_1, \dots, A_i, \dots, A_l = \square, \quad (3.12)$$

$$B_1, \dots, B_j, \dots, B_n = \square. \quad (3.13)$$

Если в (3.12) не содержится дизъюнкта L_k или в (3.13) нет дизъюнкта F_k , то такая последовательность дизъюнктов будет выводом из S и необходимость теоремы доказана. Считаем, что в (3.12) содержится L_k , например $L_k = A_i$, а в (3.13) — $F_k = B_j$.

Преобразуем (3.13) следующим образом: к дизъюнкту B_j и всем зависимым от него дизъюнктам, добавим литерал L_k . Получим следующую последовательность дизъюнктов:

$$B_1, \dots, B_{j-1}, \widehat{B}_j = F_k \vee L_k, \widehat{B}_{j+1}, \dots, \widehat{B}_m. \quad (3.14)$$

По определению (3.14) является выводом из S . Если дизъюнкт B_m не зависит от B_j , то очевидно, что $\widehat{B}_m = \square$. В этом случае необходимость теоремы доказана. Предположим, что B_m зависит от B_j . Тогда $\widehat{B}_m = L_k$.

В вывод (3.12) вместо дизъюнкта $A_i = L_k$ подставим последовательность (3.14). Получим последовательность

$$A_1, \dots, A_{i-1}, \underbrace{B_1, \dots, B_{j-1}, \widehat{B}_j, \dots, \widehat{B}_m}_{A_i} = L_k, A_{i+1}, \dots, A_l = \square. \quad (3.15)$$

Последовательность (3.15) есть вывод пустого дизъюнкта из S . Теорема доказана. ■

Используя теорему 3.15, построим алгоритм для определения того, является ли формула G логическим следствием множества формул F_1, \dots, F_k .

Алгоритм 3.8. Метод резолюций для логики высказываний

// Инициализация:

$T \leftarrow \{F_1, \dots, F_k, F_{k+1} = \overline{G}\}$

$S \leftarrow \emptyset$

for each $F_i \in T$ **do**

 приводим F_i к КНФ, $F_i = D_{i,1} \wedge \dots \wedge D_{i,n}$

 // добавляем полученные дизъюнкты $D_{i,j}$ в множество дизъюнктов S

$S \leftarrow S \cup \{D_{i,1}, \dots, D_{i,n}\}$

end for

Строим опровержение S

if опровержение S построено **then**

return: «формула G есть логическое следствие формул $\{F_1, \dots, F_k\}$ »

else

return: «формула G не является логическим следствием $\{F_1, \dots, F_k\}$ »
end if

Обоснованием алгоритма 3.8 служит следующая теорема.

Теорема 3.16. Множество формул T и множество дизъюнктов S , полученное на шаге 2 алгоритма 3.8, одновременно выполнимы (или невыполнимы).

Доказательство. Пусть формула G_i есть КНФ формулы $F_i \in T$. Тогда $G_i = D_{i_1} \wedge \dots \wedge D_{i_n}$. В силу определения конъюнкции очевидно, что выполнимость G_i равносильна выполнимости множества дизъюнктов $\{D_{i_1}, \dots, D_{i_n}\}$. ■

Пример 3.39. Применим алгоритм 3.8 для решения задачи о хищении на складе из примера 3.14.

Множество формул T имеет вид:

$$\{F_1 = X \vee Y \vee Z, F_2 = X \rightarrow Y \vee Z, F_3 = \bar{Z}, \bar{G} = \bar{Y}\}.$$

После выполнения шага 2 получаем множество дизъюнктов

$$S = \{D_1 = X \vee Y \vee Z, D_2 = \bar{X} \vee Y \vee Z, D_3 = \bar{Z}, D_4 = \bar{Y}\}.$$

Строим опровержение S .

$$D_1, D_2, D_5 \stackrel{D_2, D_1}{=} Y \vee Z, D_3, D_6 \stackrel{D_5, D_3}{=} Y, D_4, D_7 \stackrel{D_6, D_4}{=} \square.$$

Следовательно, формула G — логическое следствие, т. е. Джонсон виновен.

Пример 3.40. Петров, Иванов и Сидоров сдавали экзамен по «математической логике и теории алгоритмов». Если Петров не сдал экзамен на «отлично», то и Иванов не сдал на «отлично». Сидоров и еще один из друзей сдали экзамен на «отлично». Следует ли отсюда, что не верно, что Петров сдал экзамен не на «отлично», а Иванов на «отлично»?

Для ответа на вопрос применим метод резолюций. Запишем данные в виде формул логики высказываний. Введем обозначения:

X — Петров сдал экзамен на «отлично».

Y — Иванов сдал экзамен на «отлично».

Z — Сидоров сдал экзамен на «отлично».

Получаем следующие формулы:

$$F_1(X, Y) = \bar{X} \rightarrow \bar{Y}, F_2(X, Y, Z) = Z \wedge (X \vee Y), G(X, Y) = \overline{\bar{X} \wedge \bar{Y}}.$$

Приводим формулы к КНФ и составляем множество дизъюнктов S :

$$S = \{D_1 = X \vee \bar{Y}, D_2 = Z, D_3 = X \vee Y, D_4 = \bar{X}, D_5 = Y\}$$

Строим опровержение S .

$$D_1, D_4, D_6 \stackrel{D_1, D_4}{=} \bar{Y}, D_5, D_7 \stackrel{D_5, D_6}{=} \square.$$

Задача 3.5. Методом резолюций решите задачу о сокровище на острове рыцарей и лжецов (см. пример 3.16).

Задача 3.6. Запишите следующие рассуждения в виде последовательности формул логики высказываний. Докажите логичность рассуждений методом резолюций.

Или Анна и Антон одного возраста, или Анна старше Антона. Если Анна и Антон одного возраста, то Наташа и Антон не одного возраста. Если Анна старше Антона, то Антон старше Николая. Следует ли отсюда, что либо Наташа и Антон не одного возраста, либо Антон старше Николая?

Задача 3.7. Известно, что хроничные сепульки всегда латентны или бифуркальны. Какие из следующих утверждений в этом случае истинны:

- 1) сепульки не хроничны только в случае отсутствия у них свойства латентности;
- 2) латентность сепулек не является необходимым условием их хроничности или бифуркальности;
- 3) хроничность сепулек является достаточным условием их латентности или бифуркальности;
- 4) для нехроничности сепулек необходимо отсутствие у них как бифуркальности, так и латентности.

3.1.10. Стратегии метода резолюций

В множестве дизъюнктов часто существует не одна пара дизъюнктов, к которым можно применить правило резолюций. Способ выбора дизъюнктов и литералов в них, к которым применяется правило резолюций для получения резольвенты, называют *стратегией метода*. Рассмотрим две стратегии: стратегию насыщения уровней и стратегию вычеркивания.

Далее считаем, что множество дизъюнктов S упорядочено. Наиболее простой способ выбора дизъюнктов из S для получения резольвенты состоит в организации полного перебора возможных вариантов.

Введем необходимые обозначения. Множество дизъюнктов, полученных на шаге n алгоритма, будем обозначать как S_n .

Эти множества далее будем называть *уровнями*. Резольвенту двух дизъюнктов D_1 и D_2 будем обозначать как $\text{res}(D_1, D_2)$.

Алгоритм 3.9. Стратегия насыщения уровней

```

// Инициализация:
 $S_0 \leftarrow S$ 
 $n \leftarrow 0$ 
while  $S_n \neq \emptyset$  do
     $n \leftarrow n + 1$ 
    // Формируем множество дизъюнктов уровня  $n$ 
     $S_n \leftarrow \emptyset$ 
    //  $D_2$  «пробегают» множество дизъюнктов последнего уровня  $S_{n-1}$ 
    for each  $D_2 \in S_{n-1}$  do
        //  $D_1$  «пробегают» все предшествующие  $D_2$  дизъюнкты, начиная с уровня
         $S_0$ 
        for each  $D_1 \in S_0 \cup S_1 \cup \dots \cup S_{n-1} \mid D_1$  предшествует  $D_2$  do
            if существует резольвента  $D_1$  и  $D_2$  then
                if  $\text{res}(D_1, D_2) = \square$  then
                    return: «построено опровержение множества  $S$ »
                else
                     $S_n \leftarrow S_n \cup \text{res}(D_1, D_2)$ 
                end if
            end if
        end for
    end for
end while
return: «невозможно построить опровержение множества  $S$ »

```

Пример 3.41. Построим опровержение следующего множества дизъюнктов:

$$S = \{D_1 = X \vee Y, D_2 = \bar{X} \vee \bar{Y}, D_3 = X \vee Z, D_4 = \bar{X} \vee Z, D_5 = \bar{Z}\},$$

используя стратегию насыщения уровней.

1 Строим уровень S_1 :

$$S_1 = \{D_6 \stackrel{D_2, D_1}{=} Y \vee \bar{Y}, D_7 \stackrel{D_2, D_1}{=} X \vee \bar{X}, D_8 \stackrel{D_3, D_2}{=} \bar{Y} \vee Z, D_9 \stackrel{D_4, D_1}{=} Y \vee Z, \\ D_{10} \stackrel{D_4, D_3}{=} Z, D_{11} \stackrel{D_5, D_3}{=} X, D_{12} \stackrel{D_5, D_4}{=} \bar{X}\}.$$

2 Строим уровень S_2 :

$$S_2 = \{D_{13} \stackrel{D_6, D_1}{=} X \vee Y, D_{14} \stackrel{D_6, D_2}{=} \bar{X} \vee \bar{Y}, D_{15} \stackrel{D_7, D_1}{=} X \vee Y, D_{16} \stackrel{D_7, D_2}{=} \bar{X} \vee \bar{Y}, \\ D_{17} \stackrel{D_7, D_3}{=} X \vee Z, D_{18} \stackrel{D_7, D_4}{=} \bar{X} \vee Z, D_{19} \stackrel{D_8, D_1}{=} X \vee Z, D_{20} \stackrel{D_8, D_5}{=} \bar{Y}, \\ D_{21} \stackrel{D_8, D_6}{=} \bar{Y} \vee Z, D_{22} \stackrel{D_9, D_2}{=} \bar{X} \vee Z, D_{23} \stackrel{D_9, D_6}{=} Y \vee Z, D_{24} \stackrel{D_9, D_8}{=} Z, \\ D_{25} \stackrel{D_{10}, D_5}{=} \square\}.$$

Замечание 3.13

Из примера 3.41 видно, что стратегия насыщения уровней порождает много лишних дизъюнктов. Например, дизъюнкты D_6 и D_7 тождественно истинны. Удаление или добавление тождественно истинного дизъюнкта не влияет на выполнимость S , поэтому такие дизъюнкты должны быть удалены из вывода.

Некоторые дизъюнкты порождаются в выводе неоднократно, например:

$$D_{13} = D_{15} = X \vee Y, D_8 = D_{20} = \bar{Y} \vee Z.$$

Оптимизируем алгоритм 3.9 с учетом сделанных замечаний.

Определение 3.29. Дизъюнкт D называют *расширением* дизъюнкта F , если $D = F \vee G$ (с учетом возможной перестановки литералов). Например, $X \vee Y \vee Z$ есть расширение $X \vee Z$.

Оптимизация алгоритма 3.9 заключается в следующем: после того как получена очередная резольвента D дизъюнктов D_1 и D_2 , проверяется, является ли она тавтологией или расширением некоторого дизъюнкта $F \in S_0 \cup \dots \cup S_{n-1}$, и в случае положительного ответа D вычеркивается, т. е. не включается в последовательность S_n .

Данная стратегия носит название *стратегии вычеркивания*.

Пример 3.42. Применим стратегию вычеркивания ко множеству дизъюнктов S из примера 3.41.

1 Строим уровень S_1 :

$$S_1 = \{D_6 \stackrel{D_3, D_2}{=} \bar{Y} \vee Z, D_7 \stackrel{D_4, D_1}{=} Y \vee Z, D_8 \stackrel{D_4, D_3}{=} Z, D_9 \stackrel{D_5, D_3}{=} X, D_{10} \stackrel{D_5, D_4}{=} \bar{X}\}.$$

2 Строим уровень S_2 :

$$S_2 = \{D_{11} \stackrel{D_6, D_5}{=} \bar{Y}, D_{12} \stackrel{D_7, D_5}{=} Y, D_{13} \stackrel{D_8, D_5}{=} \square\}.$$

Пример 3.43. Рассмотрим следующее множество дизъюнктов:

$$S = \{D_1 = \bar{A} \vee C, D_2 = \bar{B} \vee C, D_3 = \bar{A} \vee B \vee C, D_4 = \bar{C} \vee D, \\ D_5 = A \vee D, D_6 = \bar{D}\}.$$

Построим опровержение S , используя стратегию вычеркивания.

1 Строим уровень S_1 :

$$S_1 = \{D_7 \stackrel{D_1, D_4}{=} \bar{A} \vee D, D_8 \stackrel{D_1, D_5}{=} C \vee D, D_9 \stackrel{D_2, D_4}{=} \bar{B} \vee D, \\ D_{10} \stackrel{D_3, D_4}{=} \bar{A} \vee B \vee D, D_{11} \stackrel{D_4, D_6}{=} \bar{C}, \\ D_{12} \stackrel{D_5, D_6}{=} A\}.$$

2 Строим уровень S_2 :

$$S_2 = \{D_{13} \stackrel{D_1, D_{12}}{=} \bar{A}, D_{14} \stackrel{D_1, D_{13}}{=} C, D_{15} \stackrel{D_2, D_{12}}{=} \bar{B}, D_{16} \stackrel{D_4, D_8}{=} D, \\ D_{17} \stackrel{D_6, D_{17}}{=} \square\}.$$

Задача 3.8. Построить опровержение следующего множества дизъюнктов S , используя стратегию вычеркивания:

$$S = \{D_1 = A \vee B \vee C, D_2 = \bar{A} \vee B \vee C, D_3 = B \vee \bar{C}, D_4 = \bar{B} \vee C, \\ D_5 = \bar{C} \vee A, D_6 = \bar{C}\}.$$

Теорема 3.17. Если множество дизъюнктов S невыполнимо, то, используя стратегию насыщения уровней и стратегию вычеркивания, можно построить опровержение S .

Доказательство. Для стратегии насыщения уровней это очевидно. Рассмотрим стратегию вычеркивания. Заметим, что если D и F — дизъюнкты из S и D есть расширение F , то множество S невыполнимо в том и только в том случае, когда невыполнимо множество $S \setminus \{D\}$. ■

3.1.11. Линейная резолюция

Линейная резолюция соответствует следующей схеме: выбирается некоторый дизъюнкт из исходного множества дизъюнктов, к нему и к другому дизъюнкту применяется правило резолюции. К полученной резольвенте и какому-либо другому дизъюнкту опять применяем правило резолюции, получаем следующий дизъюнкт и т. д. Процесс повторяется до тех пор, пока не будет получен пустой дизъюнкт, или для текущего дизъюнкта будет невозможно применить правило резолюции ни с одним другим дизъюнктом.

В последнем случае осуществляется откат к тому дизъюнкту, где был возможен множественный выбор дизъюнкта для резольвирования.

Определение 3.30. Рассмотрим множество дизъюнктов S . Возьмем дизъюнкт $C_0 \in S$. *Линейный вывод* дизъюнкта C из множества дизъюнктов S — это вывод, в котором выполняются условия:

- 1) для $i = 0, \dots, k - 1$ дизъюнкт C_{i+1} есть резольвента дизъюнкта C_i (называемого *центральным дизъюнктом*) и B_t (называемого *боковым*);
- 2) каждый B_t либо принадлежит S , либо есть C_j для некоторого $j < i$.

Общая схема линейной резолюции изображена на рисунке 3.9.

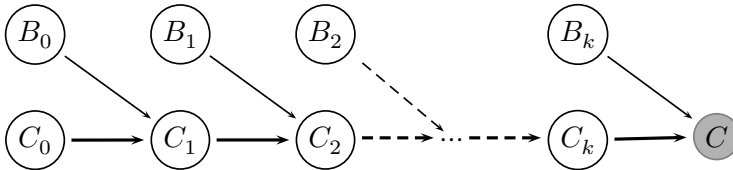


Рис. 3.9.

Пример 3.44. Формируется экипаж для кругосветного путешествия. Браун, Джонсон и Смит проходят тестирование. Окончательное слово остается за командиром экипажа. После беседы с ними командир высказал свое мнение:

- 1) в путешествие могут отправиться Браун, Джонсон и Смит;
- 2) если я возьму Брауна, но не возьму Джонсона, то возьму также и Смита;
- 3) членами экипажа станут либо Смит и Джонс, либо ни один из них;
- 4) если в экипаж будет принят Смит, то также будет принят и Браун.

Является ли утверждение «Смит отправится в путешествие» логическим следствием этих высказываний?

Введем следующие обозначения:

- 1) B — Браун отправится в путешествие;
- 2) J — Джонс отправится в путешествие;
- 3) S — Смит отправится в путешествие.

Запишем в виде формул высказывания командира экипажа и вопрос о Смите (обозначим его через G):

$$F_1 = B \vee J \vee S; F_2 = B \wedge \bar{J} \rightarrow S; F_3 = J \wedge S \vee \bar{J} \wedge \bar{S}; F_4 = S \rightarrow B; G = S.$$

Приводим формулы к КНФ и строим множество дизъюнктов:

$$D_1 = B \vee J \vee S, D_2 = \bar{B} \vee J \vee S, D_3 = \bar{J} \vee S, D_4 = J \vee \bar{S}, D_5 = \bar{S} \vee B, D_6 = \bar{S}.$$

Применяем линейную резолюцию, начиная с D_6 (рисунок 3.10).

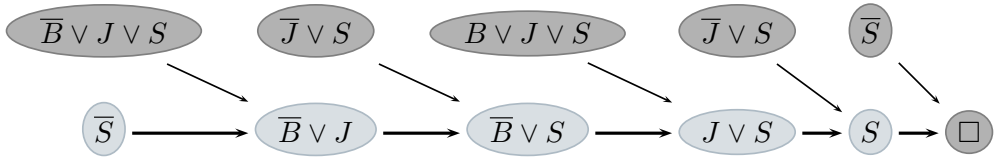


Рис. 3.10.

3.1.12. Решетки, булевы алгебры, кольца

Вернемся еще раз к логическим законам, установленным в теореме 3.4.

Доказанные для формул логики высказываний эти равносильности позволяют определить новые алгебраические структуры, в дополнение к уже рассмотренным в главе 1 понятиям группы, кольца и поля (см. раздел 1.3.1) При этом множество формул логики высказываний будет одним из примеров реализации новых понятий.

Дадим соответствующие определения. Как и прежде, запись $(\mathfrak{M}, \vee, \wedge)$ будет означать, что на непустом множестве \mathfrak{M} заданы операции \vee и \wedge .

Определение 3.31. Множество $(\mathfrak{M}, \vee, \wedge)$ называют *решеткой*, если для любых элементов $a, b, c \in \mathfrak{M}$ выполнены следующие условия:

- 1) *ассоциативность*: $(a \vee b) \vee c = a \vee (b \vee c)$; $(a \wedge b) \wedge c = a \wedge (b \wedge c)$;
- 2) *коммутативность*: $a \vee b = b \vee a$; $a \wedge b = b \wedge a$;
- 3) *законы поглощения*: $(a \vee b) \wedge a = a$; $(a \wedge b) \vee a = a$.

Пример 3.45. Рассмотрим примеры решеток:

- 1) $(\mathbb{N}, D(x, y), M(x, y))$ — множество натуральных чисел с операциями вычисления наибольшего общего делителя и наименьшего общего кратного;
- 2) $(\mathbb{R}, \max\{x, y\}, \min\{x, y\})$ — множество вещественных чисел с операциями вычисления большего и меньшего числа из двух данных чисел.

 **Замечание 3.14**

Аксиомы решетки двойственны относительно операций \vee и \wedge . Поэтому из любого утверждения, доказанного с помощью аксиом, можно получить двойственное предположение, поменяв местами в исходном утверждении знаки \vee и \wedge .

Лемма 3.5. Для любого элемента a решетки \mathfrak{M} выполняются законы идемпотентности¹:

$$a \vee a = a, \quad a \wedge a = a. \quad (3.16)$$

Доказательство. Применив второй закон поглощения для элементов a и $a \vee b$, имеем

$$a = (a \wedge (a \vee b)) \vee a = ((a \vee b) \wedge a) \vee a = a \vee a.$$

Согласно замечанию 3.14, заменив в рассуждениях операцию \vee на \wedge , получаем второе равенство в (3.16). ■

В произвольной решетке легко ввести отношение нестрогого частичного порядка (см. определение 2.79).

Определим это отношение R следующим образом: aRb , если $a \wedge b = a$. Справедлива теорема.

Теорема 3.18. Введенное отношение R является отношением частичного порядка.

Доказательство. Согласно второму закону идемпотентности (3.16) отношение R рефлексивно. Докажем его антисимметричность. Пусть aRb и bRa . Тогда

$$a \stackrel{aRb}{=} a \wedge b = b \wedge a \stackrel{bRa}{=} b.$$

Установим транзитивность отношения. Пусть aRb и bRc . Покажем, что тогда aRc . Имеем:

$$a \wedge c \stackrel{aRb}{=} (a \wedge b) \wedge c = a \wedge (b \wedge c) \stackrel{bRc}{=} a \wedge b \stackrel{aRb}{=} a. \quad \blacksquare$$

¹ Иногда их включают в определение решетки.

Определение 3.32. Решетку $(\mathfrak{M}, \vee, \wedge)$, в которой операции \vee и \wedge дистрибутивны одна относительно другой, т. е. для любых элементов $a, b, c \in \mathfrak{M}$ выполняются следующие тождества:

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c), \quad a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c),$$

называют *дистрибутивной решеткой*.

Пример 3.46. Рассмотрим множество всех подмножеств произвольного множества M с операциями объединения и пересечения подмножеств. Такое множество называют *булеаном* и обозначают $B(M)$ или 2^M . Очевидно, что $(2^M, \cup, \cap)$ — дистрибутивная решетка.

Определение 3.33. Нейтральные элементы решетки $(\mathfrak{M}, \vee, \wedge)$ по операциям \vee и \wedge называют соответственно *нулем* или *нижней гранью* (обозначают 0) и *единицей*, или *верхней гранью* (обозначают 1), т. е. для любого элемента $a \in \mathfrak{M}$ выполняются тождества:

$$0 \vee a = a, \quad 1 \wedge a = a. \quad (3.17)$$

Решетку, для которой существуют нижняя и верхняя грани, называют *ограниченной*.

Лемма 3.6. Если в решетке существуют ноль и единица, то они единственны.

Доказательство. Пусть $0'$ и $0''$ — нули решетки. Тогда, используя первое равенство в (3.17) для $0'$ и $0''$ и коммутативность операции \vee , получаем

$$0'' = 0' \vee 0'' = 0'' \vee 0' = 0'.$$

Аналогично доказывается единственность единицы. ■

Лемма 3.7. Для любого элемента решетки $a \in \mathfrak{M}$ выполняются тождества

$$0 \wedge a = 0, \quad 1 \vee a = 1. \quad (3.18)$$

Доказательство. Используя (3.17) и аксиому поглощения, получаем

$$0 \wedge a = 0 \wedge (0 \vee a) = 0.$$

Второе тождество в (3.18) доказывается аналогично. ■

Определение 3.34. Элемент $\bar{a} \in \mathfrak{M}$ называют *дополнением* элемента a в ограниченной решетке $(\mathfrak{M}, \vee, \wedge)$ (обозначают $\neg a$), если

$$a \vee \neg a = 1, \quad a \wedge \neg a = 0.$$

Если для каждого элемента ограниченной решетки существует дополнение, то решетку называют *решеткой с дополнением*.

Дистрибутивную ограниченную решетку с дополнением называют *булевой алгеброй* и обозначают $(\mathfrak{M}, \vee, \wedge, \neg)$.

Лемма 3.8. Если дополнение элемента существует, то оно единственно.

Доказательство. Предположим, что для элемента a существуют два дополнения a_1 и a_2 . Имеем

$$\begin{aligned} a_1 &= a_1 \wedge 1 = a_1 \wedge (a \vee a_2) = (a_1 \wedge a) \vee (a_1 \wedge a_2) = (a_2 \wedge a) \vee (a_2 \wedge a_1) = \\ &= a_2 \wedge (a \vee a_1) = a_2 \wedge 1 = a_2. \quad \blacksquare \end{aligned}$$

Пример 3.47. Приведем примеры булевых алгебр.

- Самая простая булева алгебра содержит два элемента 0 и 1, а действия в ней определяются таблицей истинности связок для \vee и \wedge (см. таблицу 3.1). При этом 0 есть дополнение 1, а 1 — дополнение 0.

- Решетка $(2^M, \cup, \cap)$ из примера 3.46 является булевой алгеброй. При этом 0 есть пустое множество, а 1 — само множество M . Дополнением произвольного элемента $A \subseteq M$ является множество $M \setminus A$. Эта булева алгебра носит название *алгебры Кантора*.

- Множество всех формул исчисления высказываний есть булева алгебра, если отождествить равносильные формулы, т. е. в качестве элементов множества рассматривать классы эквивалентности, определяемые отношением равносильности. Дополнение при этом определяется логической связкой отрицания \neg . Единицей является класс тавтологий, а нулем — класс противоречий. Эту алгебру часто называют *алгеброй Линденбаума — Тарского*.

- Рассмотрим произвольное натуральное число n . В качестве \mathfrak{M} возьмем множество всех его натуральных делителей. Для элементов $x, y \in \mathfrak{M}$ введем операции:

$$x \wedge y = D(x, y); \quad x \vee y = M(x, y); \quad \neg x = \frac{n}{x}.$$

Нулем алгебры является число 1, а единицей — n .

Установим справедливость некоторых равносильностей теоремы 3.4 для произвольной булевой алгебры.

Теорема 3.19. Пусть $(\mathfrak{M}, \vee, \wedge, \neg)$ — булева алгебра. Тогда для любых элементов $a, b \in \mathfrak{M}$ справедливы законы де Моргана и снятия двойного отрицания:

$$\neg(a \wedge b) = \neg a \vee \neg b; \quad \neg(a \vee b) = \neg a \wedge \neg b; \quad \neg(\neg a) = a. \quad (3.19)$$

Доказательство. Последнее равенство в (3.19) очевидно в силу коммутативности операций \vee, \wedge и единственности дополнения (см. лемму 3.8)). Докажем, например, второе тождество в (3.19). Имеем

$$\begin{aligned} (a \vee b) \vee (\neg a \wedge \neg b) &= ((a \vee b) \vee \neg a) \wedge ((a \vee b) \vee \neg b) = \\ &= ((a \vee \neg a) \vee b) \wedge ((b \vee \neg b) \vee a) = \\ &= (1 \vee b) \wedge (1 \vee a) = 1. \end{aligned}$$

Аналогично доказывается, что $(a \vee b) \wedge (\neg a \wedge \neg b) = 0$. Таким образом, элемент $\neg a \wedge \neg b$ есть дополнение элемента $a \vee b$. ■

В теории графов было введено понятие изоморфизма графов как взаимно однозначное отображение, сохраняющее соответствие между ребрами графов (см. определение 2.46). Применим эти соображения для булевых алгебр.

Определение 3.35. Взаимно однозначное отображение φ между двумя булевыми алгебрами \mathfrak{M} и \mathfrak{N} называют *изоморфизмом*, если ноль и единица алгебры \mathfrak{M} переходят соответственно в ноль и единицу алгебры \mathfrak{N} и для любых элементов $a, b \in \mathfrak{M}$:

$$\varphi(\neg a) = \neg \varphi(a), \quad \varphi(a \vee b) = \varphi(a) \vee \varphi(b), \quad \varphi(a \wedge b) = \varphi(a) \wedge \varphi(b).$$

Булевы алгебры \mathfrak{M} и \mathfrak{N} в этом случае называют *изоморфными*.

Как уже отмечалось, аксиомы булевой алгебры соответствуют логическим законам формул логики высказываний, установленным в теореме 3.4. Согласно принципу двойственности логики высказываний, если конъюнкцию заменить дизъюнкцией, дизъюнкцию — конъюнкцией, 0 — на 1, а 1 — на 0, то должна получиться та же система аксиом. Это рассуждение приводит к очень важному примеру изоморфизма булевых алгебр.

Пример 3.48. Пусть $(\mathfrak{M}, \vee, \wedge, \neg)$ — булева алгебра. Построим изоморфную булеву алгебру \mathfrak{M}^* , состоящую из элементов алгебры \mathfrak{M} , но с другими операциями. Изоморфизм осуществляет отображение $\varphi : a \rightarrow \neg a$. То, что отображение φ является изоморфизмом, очевидным образом следует из теоремы 3.19.

Определение 3.36. Булеву алгебру \mathfrak{M}^* называют *двойственной* к булевой алгебре \mathfrak{M} .

Упражнение 3.1. Докажите, что $(\mathfrak{M}^*)^* = \mathfrak{M}$.

В общем случае вопрос о существовании изоморфизма решает следующая теорема.

Теорема 3.20. Любая конечная булева алгебра изоморфна булевой алгебре всех подмножеств некоторого множества (алгебре Кантора).

Следствие 3.3. Количество элементов в любой конечной булевой алгебре есть степень двойки.

С булевой алгеброй тесно связано еще одно понятие.

Определение 3.37. Рассмотрим ассоциативное кольцо с единицей (см. определение 1.18), в котором для каждого элемента выполнен закон идемпотентности по умножению: $aa = a$. Такое кольцо называют *булевым*.

Булево кольцо легко можно преобразовать в булеву алгебру. Введем операции \vee , \wedge и \neg следующим образом:

$$a \vee b = a + b + ab, \quad a \wedge b = ab, \quad \neg a = a + 1. \quad (3.20)$$

Обратно, если \mathfrak{M} — булева алгебра, то ее можно превратить в булево кольцо, полагая:

$$ab = a \wedge b, \quad a + b = (a \wedge \neg b) \vee (b \wedge \neg a). \quad (3.21)$$

Упражнение 3.2. Докажите, что из булевой алгебры преобразованиями (3.20) и (3.21) можно получить булево кольцо и соответственно из булева кольца — булеву алгебру.

Историческая справка

Джордж Буль родился в 1815 г. в г. Линкольне. Сын сапожного мастера, увлекающегося математикой, он окончил только начальную школу и дальнейшие знания приобретал самостоятельно. На жизнь Буль зарабатывал как учитель. В 1849 г. стал профессором математики в Куинс-колледже (Ирландия), где преподавал до конца жизни. Одна из его пяти дочерей — Этель Лилиан, в замужестве Войнич, стала автором известного романа «Овод».

Задачи и вопросы

1. Для какой формулы логики высказываний нет равносильной ей, имеющей СДНФ.

2. Для какой формулы логики высказываний нет равносильной ей, имеющей СКНФ.

3. Определите, сколькими способами можно расставить скобки в выражении

$$X \rightarrow \neg Y \vee Z \vee Y \wedge Z,$$

чтобы получилась формула.

4. Существует ли формула F такая, что формула G будет тождественно истинна:

$$G = (F \wedge Y \rightarrow \neg Z) \rightarrow ((Z \rightarrow \neg Y) \rightarrow F).$$

5. Формула логики высказываний F не содержит никаких логических связок кроме \leftrightarrow . Докажите, что формула F является тавтологией тогда и только тогда, когда каждая переменная входит в F четное число раз.

6. Докажите, что если формулы $F \vee G$ и $\neg F \vee H$ — тавтологии, то и формула $G \vee H$ — тавтология.

7. Является ли формула G логическим следствием формул F_1, F_2, F_3, F_4 :

$$F_1 = X \vee Y \vee Z, F_2 = X \rightarrow X_1, F_3 = Y \rightarrow X_1 \vee Y_1, F_4 = \neg Y_1, G = Z \rightarrow X_1.$$

8. Является ли формула G логическим следствием формул F_1, F_2, F_3 :

$$F_1 = X \neg Y \vee Z, F_2 = Y \rightarrow W, F_3 = Z \rightarrow X, G = X \rightarrow W.$$

9. Будет ли логичным следующее рассуждение: *Намеченная атака удастся, если захватить противника врасплох или его позиции плохо защищены. Захватить противника врасплох можно только тогда, когда он беспечен. Он не будет беспечен, если его позиции плохо защищены. Следовательно, намеченная атака не удастся.*

10. Какой дизъюнкт выводим по правилу резолюций в логике высказываний из дизъюнктов $X \vee F$ и $\neg X \vee G$.

11. Методом минимизирующих карт найдите минимальную ДНФ для формулы логики высказываний $F(X, Y, Z)$, принимающую значение 1 на наборах $(1, 1, 1)$, $(1, 0, 1)$, $(0, 0, 1)$ и $(0, 0, 0)$.

12. Методом Куайна — Мак-Класки найдите минимальную ДНФ для формулы логики высказываний $F(X, Y, Z)$, принимающую значение 1 на наборах $(1, 1, 0)$, $(1, 0, 0)$, $(0, 1, 1)$ и $(0, 0, 0)$.

3.2. Булевы функции

3.2.1. Основные понятия

Определение 3.38. Функцию $f(x_1, \dots, x_n)$, аргументы и значение которой определены на множестве $\{0, 1\}$ называют *булевой функцией*

Булевы функции также называют *переключательными, логическими* функциями или функциями алгебры логики.

Множество $\{0, 1\}$ будем в дальнейшем обозначать B .

Для задания булевой функции $f(x_1, \dots, x_n)$ достаточно определить, какое значение функции соответствует каждому из наборов значений аргументов, т. е. заполнить таблицу 3.24.

Таблица 3.24.

x_1	x_2	...	x_{n-1}	x_n	$f(x_1, x_2, \dots, x_{n-1}, x_n)$
0	0	...	0	0	$f(0, 0, \dots, 0, 0)$
0	0	...	0	1	$f(0, 0, \dots, 0, 1)$
...
1	1	...	1	0	$f(1, 1, \dots, 1, 0)$
1	1	...	1	1	$f(1, 1, \dots, 1, 1)$

Заметим, что таблица 3.24 и таблица истинности для формулы логики высказываний функции (см. таблицу 3.2) идентичны. Таким образом, формулы логики высказываний и булевы функции — это два различных подхода к интерпретации таблицы истинности.

Формулу логики высказываний можно считать синтаксической записью таблицы истинности, а для булевой функции это табличное задание функциональной зависимости.

Пусть $F(X_1, \dots, X_n)$ и $G(X_1, \dots, X_n)$ — формулы логики высказываний. Заменяем в формулах прописные буквы X_i на строчные x_i ($1 \leq i \leq n$). Получим булевы функции $f(x_1, \dots, x_n)$ и $g(x_1, \dots, x_n)$. Тогда формулы $F(X_1, \dots, X_n)$ и $G(X_1, \dots, X_n)$ равносильны в том и только в том случае, когда функции $f(x_1, \dots, x_n)$ и $g(x_1, \dots, x_n)$ равны.

Эти соображения позволяют перенести на булевы функции результаты предыдущих разделов: равносильность формул логики высказываний, нормальные формы и алгоритмы приведения к ним.

Справедлива следующая теорема.

Теорема 3.21. Число различных булевых функций от n переменных равно 2^{2^n} .

Доказательство. Зафиксируем число аргументов n . Очевидно, что число различных булевых функций от n переменных равно числу различных таблиц истинности. При лексикографическом упорядочении строк число различных таблиц равно числу способов заполнения последнего столбца (значений функции). Число способов заполнения этого столбца равно 2^h , где h — высота столбца, но $h = 2^n$. Таким образом, число различных булевых функций от n переменных равно 2^{2^n} . ■

Упражнение 3.3. Найдите число булевых функций от n переменных, которые на фиксированных $m \leq n$ наборах принимают значение 0.

 **Замечание 3.15**

Для обозначения булевых функций двух переменных наряду с префиксной записью $f(x_1, x_2)$ будем применять и инфиксную $x_1 f x_2$.

Рассмотрим примеры булевых функций. Основную роль здесь играют функции одной (таблица 3.25а) и двух (таблица 3.25b) переменных.

Таблица 3.25.

x	0	1	x	\bar{x}
0	0	1	0	1
1	0	1	1	0

(a)

x	y	xy	$x \vee y$	$x \rightarrow y$	$x \leftrightarrow y$	$x \oplus y$	$x y$	$x \downarrow y$
0	0	0	0	1	1	0	1	1
0	1	0	1	1	0	1	1	0
1	0	0	1	0	0	1	1	0
1	1	1	1	1	1	0	0	0

(b)

Условимся также об обозначении некоторых функций.

Для функций $x \vee y$, $x \rightarrow y$, $x \leftrightarrow y$ сохраним названия, идущие из логики высказываний: *дизъюнкция*, *импликация* и *эквиваленция*.

Функции xy и $x \oplus y$ — это, естественно, *умножение* и *сложение*; умножение можно было бы назвать *конъюнкцией*, сложение иногда называют *сложением по модулю 2*, или *функцией неравнозначности*.

Функция $x | y$ — *штрих Шеффера*, $x \downarrow y$ — *стрелка Пирса*, *стрелка Лукасевича*, *кинжал Куайна* или *функция Вебба*.

Историческая справка

Функцию $x \downarrow y$ ввел в рассмотрение американский логик и математик Чарльз Пирс в 1880 г.

Функция $x | y$ была предложена американским логиком Генри Шеффером в 1913 г.

Определение булевой функции позволяет рассматривать функции только от фиксированного числа аргументов. Аналогично формулам логики высказываний введем понятие *фиктивных* переменных.

Определение 3.39. Булева функция $f(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n)$ существенно зависит от переменной x_i , если существует такой булевый набор значений переменных $(\alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_n)$, что

$$f(\alpha_1, \dots, \alpha_{i-1}, 0, \alpha_{i+1}, \dots, \alpha_n) \neq f(\alpha_1, \dots, \alpha_{i-1}, 1, \alpha_{i+1}, \dots, \alpha_n). \quad (3.22)$$

В этом случае переменную x_i называют *существенной*. Если x_i не является существенной переменной, то ее называют *фиктивной*.

Пусть для функции $f(x_1, \dots, x_n)$ переменная x_i является фиктивной. По таблице функции $f(x_1, \dots, x_n)$ построим новую таблицу, вычеркнув все строки вида $(\alpha_1, \dots, \alpha_{i-1}, 1, \alpha_{i+1}, \dots, \alpha_n)$ и столбец переменной x_i . Полученная таблица определяет новую функцию $g(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$.

Говорят, что функция g получена из функции f путем удаления фиктивной переменной x_i , а также, что функция f получается из функции g путем введения фиктивной переменной x_i .

Определение 3.40. Функции $f(x_1, \dots, x_n)$ и $g(x_1, \dots, x_m)$ называют *равными*, если одна из них получается из другой с помощью введения или вычеркивания некоторых фиктивных переменных.

Пример 3.49. Пусть функция $f(x_1, x_2, x_3)$ задана таблицей значений (таблица 3.26). Определим ее существенные и фиктивные переменные.

Таблица 3.26.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	1	1	0	1
0	1	1	1	1	1	1	1

Как и для формул логики высказываний, в цепочке равенств булевых функций будем ставить номера использованных равносильностей из теоремы 3.4.

Неравенство (3.22) выполняется только для x_2 : $f(0, 0, 0) \neq f(0, 1, 0)$. Переменная x_2 — существенная, а переменные x_1, x_3 — фиктивные.

Действительно, если, используя таблицу 3.26, по алгоритму 3.3 построить СДНФ для $f(x_1, x_2, x_3)$, то получим

$$\begin{aligned} f(x_1, x_2, x_3) &= \bar{x}_1 x_2 \bar{x}_3 \vee \bar{x}_1 x_2 x_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 x_2 x_3 \stackrel{7}{=} \\ &\stackrel{7}{=} \bar{x}_1 x_2 (\bar{x}_3 \vee x_3) \vee x_1 x_2 (\bar{x}_3 \vee x_3) \stackrel{10}{=} \bar{x}_1 x_2 \vee x_1 x_2 \stackrel{7}{=} \\ &\stackrel{7}{=} x_2 (\bar{x}_1 \vee x_1) \stackrel{10}{=} x_2. \end{aligned}$$

Таким образом, $f(x_1, x_2, x_3) = g(x_2) = x_2$.

Упражнение 3.4. Докажите, что для любой булевой функции, отличной от константы 0 или 1, существует равная ей, у которой все переменные существенные.

Замечание 3.16

При лексикографическом упорядочении строк таблицы истинности булеву функцию иногда задают вектором значений функции. Например, булеву функцию из примера 3.49 можно записать как $f(x_1, x_2, x_3) = (00110011)$.

Задача 3.9. Определите существенные и фиктивные переменные функции $f(x_1, x_2, x_3) = (11110011)$.

Задача 3.10. Найдите все булевы функции двух и трех переменных, у которых все переменные существенны.

С табличным заданием функции непосредственно связан такой ее параметр, как *вес*.

Определение 3.41. Множество двоичных наборов $\{a_1, \dots, a_n\}$, на которых булева функция $f(x_1, \dots, x_n)$ принимает значение 1, называют *областью истинности* функции f и обозначают E_f .

Мощность области истинности функции f называют *весом* функции f и обозначают $\|f\|$.

Очевидно, что $0 \leq \|f\| \leq 2^n$, причем равенства достигаются лишь для функций-констант 0 и 1.

Определение 3.42. Булеву функцию от $2l + 1$ переменных, равную 1, если $m \geq l + 1$ переменных принимают значение 1, и 0 — в противном случае, называют *мажоритарной* или *функцией голосования* и обозначают Ma_{j2l+1} .

Пример 3.50. Покажем, что $\|Maj_{2l+1}\| = 2^{2l}$.

Действительно, рассмотрим двоичный набор α :

$$\alpha = (\alpha_1, \dots, \alpha_{2l+1}), \alpha_1 + \dots + \alpha_{2l+1} \geq l + 1, Maj_{2l+1}(\alpha_1, \dots, \alpha_{2l+1}) = 1.$$

Каждому такому набору α однозначно соответствует набор $\bar{\alpha}$:

$$\bar{\alpha} = (\bar{\alpha}_1, \dots, \bar{\alpha}_{2l+1}), \bar{\alpha}_1 + \dots + \bar{\alpha}_{2l+1} \leq l, Maj_{2l+1}(\bar{\alpha}_1, \dots, \bar{\alpha}_{2l+1}) = 0.$$

Таким образом, $Maj_{2l+1}(x_1, \dots, x_{2l+1})$ принимает единичные значения на половине наборов.

Для булевых функций используются и другие способы задания. Рассмотрим *геометрический способ*.

Под геометрическим способом задания булевой функции $f(x_1, \dots, x_n)$ понимается выделение тех вершин n -мерного двоичного куба, на наборах координат которых функция принимает единичное значение.

На рисунке 3.11 показаны геометрические представления для функций $f(x_1) = \bar{x}_1$ (рисунок 3.11a), $g(x_1, x_2) = x_1 \oplus x_2$ (рисунок 3.11b) и мажоритарной функции $Maj_3(x_1, x_2, x_3) = x_1x_2 \vee x_1x_3 \vee x_2x_3$ (рисунок 3.11c).

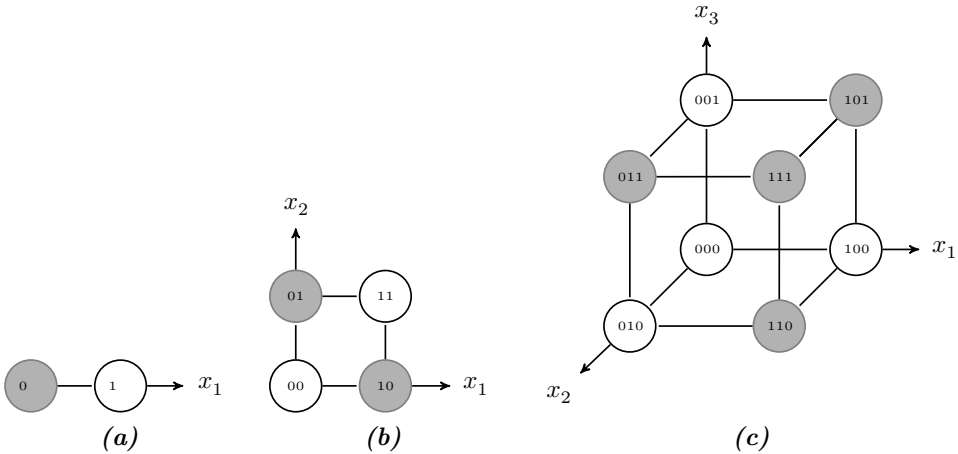
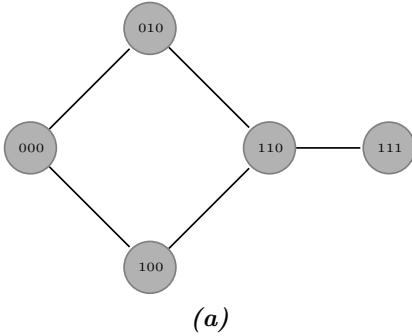


Рис. 3.11.

Как видно из рисунка 3.11, геометрический способ представления булевых функций n переменных эффективен для небольших размерностей $n = 1, 2, 3$.

Часто по геометрическому заданию функции строят *граф связности* вершин n -мерного куба, соответствующий данной функции. Для этого сначала отмечают те ребра, у которых оба конца выделены, т. е. соответствующие вершины лежат в области истинности. Затем все остальные ребра и вершины, не лежащие в области истинности, отбрасывают.

На рисунке 3.12а изображен граф связности для булевой функции $f(x_1, x_2, x_3)$, заданной таблицей значений (рисунок 3.12b).



x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

(b)

Рис. 3.12.

3.2.2. Замкнутость и полнота

Определение 3.43. Пусть имеется некоторый набор булевых функций K . *Суперпозицией функций* из этого набора называют булевы функции, полученные с помощью конечного числа применения двух операций:

- 1) переименование любой переменной, входящей в функцию из K ;
- 2) подстановка функции из K или полученной ранее суперпозиции вместо любой переменной.

Суперпозицию иначе называют *сложной функцией*.

Пример 3.51. Пусть $x \oplus y \in K$. Переименованием переменной y получаем функцию $0 = x \oplus x$.

Пусть $x \vee y, \bar{x} \in K$. Тогда подстановкой получаем функцию xy , поскольку

$$xy = \overline{\bar{x} \vee \bar{y}}. \quad (3.23)$$

Определение 3.44. *Замыканием* набора функций из K (обозначают $[K]$) называют множество всех суперпозиций этого набора.

Класс функций K называют *замкнутым*, если его замыкание совпадает с ним самим.

Лемма 3.9. Справедливы следующие свойства замыкания.

1. $K \subseteq [K]$.
2. $[[K]] = [K]$.
3. Если $F \subseteq G$, то $[F] \subseteq [G]$.
4. $[F \cap G] \subseteq [F] \cap [G]$.
5. $[F] \cup [G] \subseteq [F \cup G]$.

Доказательство очевидным образом следует из определения 3.44.

Определение 3.45. Функцию $f(x_1, \dots, x_n)$ называют *сохраняющей ноль*, если $f(0, \dots, 0) = 0$.

Через T_0 обозначим класс всех функций, сохраняющих 0, т. е.

$$T_0 = \{f(x_1, \dots, x_n) \mid f(0, \dots, 0) = 0\}.$$

Лемма 3.10. Класс T_0 не пуст и не совпадает со всем множеством булевых функций B_F .

Доказательство. Функции xy , $x \vee y$, $x \oplus y \in T_0$, следовательно $T_0 \neq \emptyset$, а функции $x \leftrightarrow y$, $x \mid y$, $x \downarrow y \notin T_0$, т. е. $T_0 \neq B_F$. ■

Справедлива следующая теорема.

Теорема 3.22. Класс T_0 замкнут.

Доказательство. Если $f(x_1, \dots, x_n)$ сохраняет 0 и y_1, \dots, y_n — новые переменные, то очевидно, что $f(y_1, \dots, y_n)$ также сохраняет ноль. Это означает, что T_0 замкнут относительно операции переименования переменных. Пусть $f(x_1, \dots, x_k)$, $g_1(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n)$ сохраняют 0. Тогда

$$f(g_1(0, \dots, 0), \dots, g_k(0, \dots, 0)) = f(0, \dots, 0) = 0.$$

Это означает, что T_0 замкнут относительно операции подстановки. Таким образом доказано, что T_0 — замкнутый класс. ■

Упражнение 3.5. Обозначим через T_0^n множество всех булевых функций от n переменных, сохраняющих 0. Докажите, что

$$|T_0^n| = 2^{2^n - 1}.$$

Определение 3.46. Функцию $f(x_1, \dots, x_n)$ называют *сохраняющей единицу*, если $f(1, \dots, 1) = 1$.

Через T_1 обозначим класс всех функций, сохраняющих 1, т. е.

$$T_1 = \{f(x_1, \dots, x_n) \mid f(1, \dots, 1) = 1\}.$$

Лемма 3.11. Класс T_1 не пуст и не совпадает со всем множеством булевых функций B_F .

Доказательство. Функции $xy, x \vee y, x \rightarrow y \in T_1$, т. е. $T_1 \neq \emptyset$, а функции $x \oplus y, x | y, x \downarrow y \notin T_1$, следовательно, $T_1 \neq B_F$. ■

Теорема 3.23. Класс T_1 является замкнутым.

Доказательство теоремы 3.23 аналогично приведенному в теореме 3.22.

Упражнение 3.6. Обозначим через T_1^n множество всех булевых функций от n переменных, сохраняющих 1. Докажите, что

$$|T_1^n| = 2^{2^n - 1}.$$

Обозначим через B_F класс всех булевых функций.

Определение 3.47. Класс булевых функций K называют *полным*, если $[K] = B_F$.

Другими словами, класс K полон, если любую булеву функцию можно получить из K конечным числом операций подстановки и переименования переменных.

Очевидно следующее утверждение.

Лемма 3.12 (о моделировании). Если K — полный класс и любая функция класса K выражается (с помощью операций подстановки и переименования переменных) через функции класса L , то L также полный класс.

Теорема 3.24. Следующие классы функций являются полными:

- 1) $K_1 = \{xy, x \vee y, \bar{x}\}$;
- 2) $K_2 = \{xy, \bar{x}\}$;
- 3) $K_3 = \{x \vee y, \bar{x}\}$;
- 4) $K_4 = \{xy, x \oplus y, 1, 0\}$.

Доказательство. Докажем сначала полноту класса K_1 . Рассмотрим произвольную булеву функцию $f \in B_F$. По таблице значений функции f (используя алгоритм 3.3) построим равную ей функцию g в виде СДНФ.

Но согласно определению СДНФ в записи функции g использованы только функции из класса K_1 . Следовательно, f может быть получена из функций класса K_1 операциями подстановки и переименования переменных, т. е. $f \in [K_1]$. Следовательно, любая булева функция принадлежит замыканию $[K_1]$. Таким образом, K_1 — полный класс.

Из равносильности 12 теоремы 3.4 получаем равенство

$$x \vee y = \overline{\overline{x} \overline{y}}. \quad (3.24)$$

Таким образом, функции K_2 выражаются через функции K_1 . По лемме 3.12 K_2 — полный класс.

Аналогично, используя (3.23) и применяя лемму 3.12, получаем полноту класса K_3 . Полнота класса K_4 следует из полноты класса K_2 и очевидного равенства $\overline{x} = x \oplus 1$. ■

3.2.3. Полиномы Жегалкина и линейные функции

Определение 3.48. Рассмотрим полный класс $K_4 = \{xy, x \oplus y, 1, 0\}$ из теоремы 3.24. Очевидно, что его замыкание содержит многочлены (от любого числа переменных) над полем \mathbb{Z}_2 .

Эти многочлены называют *полиномами Жегалкина*. Поскольку для элементов $a \in \mathbb{Z}_2$ справедливо тождество $a^2 = a$, то полином Жегалкина можно записать в виде

$$\sum_{\{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}} \bigoplus a_{i_1, \dots, i_k} x_{i_1} \cdots x_{i_k}, \quad a_{i_1, \dots, i_k} \in B, \quad (3.25)$$

и суммирование ведется по всем подмножествам множества $\{1, \dots, n\}$.

Число переменных в самом длинном слагаемом представления (3.25) называют *степенью нелинейности* булевой функции $\deg(f)$.

Историческая справка

Иван Иванович Жегалкин — российский и советский математик, профессор Московского университета. Полином Жегалкина был предложен им в 1927 г. [16] как удобное средство для представления булевых функций.

Пример 3.52. Рассмотрим булеву функцию от двух переменных

$$f(x_1, x_2) = 1 \cdot x_1 x_2 \oplus 0 \cdot x_1 \oplus 1 \cdot x_2 \oplus 1.$$

Это полином Жегалкина от двух переменных x_1 и x_2 с коэффициентами $a_{12} = 1$, $a_2 = 1$, $a_1 = 0$, $a_0 = 1$. Данный полином можно записать и так:

$$f(x_1, x_2) = x_1 x_2 \oplus x_2 \oplus 1, \quad \deg(f) = 2.$$

Теорема 3.25. Для любой булевой функции $f(x_1, \dots, x_n)$ существует единственное представление в виде полинома Жегалкина.

Доказательство. Существование полинома Жегалкина следует из полноты класса K_4 . Докажем единственность.

В сумме (3.25) для полинома Жегалкина от n переменных всего 2^n слагаемых (число всех подмножеств множества из n элементов), т. е. каждый полином определяется булевым набором 2^n коэффициентов.

Отсюда следует, что всего полиномов от n переменных 2^{2^n} столько же, сколько и булевых функций от n переменных (см. теорему 3.21). Но разным булевым функциям (с различными таблицами значений), очевидно, соответствуют разные полиномы Жегалкина.

Получаем взаимно однозначное соответствие между B_F и полиномами Жегалкина. ■

Замечание 3.17

В англоязычной литературе представление булевой функции в виде (3.25) иногда называют алгебраической нормальной формой.

Пример 3.53. Для функций $x_1 \rightarrow x_2$ и $x_1 \vee x_2$ справедливы равенства:

$$x_1 \rightarrow x_2 = x_1 x_2 \oplus x_1 \oplus 1, \quad x_1 \vee x_2 = x_1 x_2 \oplus x_1 \oplus x_2.$$

Упражнение 3.7. Докажите, что функция, представленная полиномом Жегалкина, существенно зависит от всех входящих в него переменных.

Рассмотрим три алгоритма представления булевой функции полиномом Жегалкина.

Первый из них можно было бы определить как *метод равносильных преобразований*. Используем соотношения:

$$\bar{x} = x \oplus 1, \quad x \vee y = \overline{\bar{x}\bar{y}} = (x \oplus 1)(y \oplus 1) \oplus 1 = xy \oplus x \oplus y, \quad (3.26)$$

Замечание 3.18

Для замены дизъюнкции $x \vee y$ полезно также использовать следующее равенство: если $xy = 0$, то $x \vee y = x \oplus y$.

Алгоритм 3.10. Метод равносильных преобразований для полинома Жегалкина

- 1: По таблице значений функции f (используя алгоритм 3.3) строим равную ей функцию g в виде СДНФ.
 - 2: Используя соотношения 3.26, заменяем отрицание и дизъюнцию в функции g на «умножение» (конъюнкцию) и «сложение» (\oplus).
 - 3: Приводим подобные члены согласно равенству $x \oplus x = 0$.
-

Пример 3.54. Пусть функция $f(x_1, x_2, x_3)$ задана таблицей 3.27.

Таблица 3.27.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	1	0	0	0
0	0	0	1	1	0	1	1
0	0	1	0	1	1	0	1
0	1	1	0	1	1	1	0

После шага 1 получаем представление

$$f(x_1, x_2, x_3) = g(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 x_2 \bar{x}_3.$$

Продолжим преобразования функции $g(x_1, x_2, x_3)$, используя (3.26). С учетом замечания 3.18 имеем:

$$\begin{aligned} g(x_1, x_2, x_3) &= \bar{x}_1 \bar{x}_2 x_3 \oplus x_1 \bar{x}_2 x_3 \oplus x_1 x_2 \bar{x}_3 = \\ &= (x_1 \oplus 1)(x_2 \oplus 1)x_3 \oplus x_1(x_2 \oplus 1)x_3 \oplus x_1 x_2(x_3 \oplus 1) = \\ &= x_1 x_2 x_3 \oplus x_1 x_2 \oplus x_2 x_3 \oplus x_3. \end{aligned}$$

Второй способ можно назвать *методом неопределенных коэффициентов*. Изложим этот способ для функции $f(x_1, x_2, x_3)$ из примера 3.54.

Пример 3.55. Рассмотрим функцию $f(x_1, x_2, x_3)$, заданную таблицей 3.27.

Последовательно находим:

- 1) $a_0 = f(0, 0, 0) = 0$;
- 2) $a_0 \oplus a_1 = f(1, 0, 0) = 0$, $a_1 = 0$;
- 3) $a_0 \oplus a_2 = f(0, 1, 0) = 0$, $a_2 = 0$;
- 4) $a_0 \oplus a_3 = f(0, 0, 1) = 1$, $a_3 = 1$;
- 5) $a_0 \oplus a_1 \oplus a_2 \oplus a_{12} = f(1, 1, 0) = 1$, $a_{12} = 1$;
- 6) $a_0 \oplus a_1 \oplus a_3 \oplus a_{13} = f(1, 0, 1) = 1$, $a_{13} = 0$;
- 7) $a_0 \oplus a_2 \oplus a_3 \oplus a_{23} = f(0, 1, 1) = 0$, $a_{23} = 1$;
- 8) $a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_{12} \oplus a_{13} \oplus a_{23} \oplus a_{123} = f(1, 1, 1) = 0$, $a_{123} = 1$.

Подставив найденные коэффициенты в представление (3.25), получим

$$f(x_1, x_2, x_3) = x_1x_2x_3 \oplus x_1x_2 \oplus x_2x_3 \oplus x_3.$$

Пример 3.56. Методом неопределенных коэффициентов найдем полином Жегалкина для булевой функции трех переменных $f(x_1, x_2, x_3)$, заданной лексикографически упорядоченным столбцом значений таблицы истинности: $(0, 1, 1, 0, 1, 0, 1, 1)$.

Согласно представлению (3.25) для функции от трех переменных полином Жегалкина с неопределенными коэффициентами имеет вид:

$$f((x_1, x_2, x_3) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus a_{12} \oplus x_1x_2 \oplus a_{13}x_1x_3 \oplus \\ \oplus a_{23}x_2x_3 \oplus a_{123}x_1x_2x_3.$$

Выпишем систему уравнений для неизвестных коэффициентов и последовательно решим ее:

- 1) $a_0 = f(0, 0, 0) = 0;$
- 2) $a_0 \oplus a_1 = f(1, 0, 0) = 1, a_1 = 1;$
- 3) $a_0 \oplus a_2 = f(0, 1, 0) = 1, a_2 = 1;$
- 4) $a_0 \oplus a_3 = f(0, 0, 1) = 1, a_3 = 1;$
- 5) $a_0 \oplus a_1 \oplus a_2 \oplus a_{12} = f(1, 1, 0) = 1, a_{12} = 1;$
- 6) $a_0 \oplus a_1 \oplus a_3 \oplus a_{13} = f(1, 0, 1) = 0, a_{13} = 0;$
- 7) $a_0 \oplus a_2 \oplus a_3 \oplus a_{23} = f(0, 1, 1) = 0, a_{23} = 0;$
- 8) $a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_{12} \oplus a_{13} \oplus a_{23} \oplus a_{123} = f(1, 1, 1) = 1, a_{123} = 1.$

Подставив найденные коэффициенты в представление (3.25), получим

$$f(x_1, x_2, x_3) = x_1x_2x_3 \oplus x_1x_2 \oplus x_3 \oplus x_2 \oplus x_1. \quad (3.27)$$

Третий алгоритм часто называют *методом треугольника*. Впервые был предложен в [54], более подробно описан в [55].

Как обычно, будем считать, что строки в таблице истинности идут в порядке возрастания двоичных кодов.

Алгоритм 3.11. Метод треугольника Паскаля для полинома Жегалкина

- 1: Строим таблицу значений функции. Вектор значений функции выписываем напротив первой строки таблицы.

- 2: Заполняем треугольник Паскаля, складывая попарно соседние булевы значения текущей строки $(\alpha_i \oplus \alpha_{i+1})$. Результат сложения выписываем в следующей строке. Продолжаем вычисления, пока в очередной строке не останется лишь одно число.
- 3: Строим конъюнкции по булевым наборам в левой части таблицы: если напротив переменной x_i стоит 1, то переменная входит в конъюнкцию; в противном случае переменная отсутствует в конъюнкции. Набору $(0, 0, 0)$ соответствует константа 1.
- 4: В состав полинома Жегалкина входят только конъюнкции из строк с единицами на левой стороне треугольника.

Пример 3.57. Рассмотрим работу алгоритма 3.11 для функции из примера 3.54. Протокол работы записан в таблице 3.28.

Строки с единичными значениями функции выделены серым цветом, значения левой стороны треугольника — полужирным шрифтом.

Таблица 3.28.

x_1	x_2	x_3	f																
0	0	0	0	0		1		0		0		0		1		1		0	
0	0	1	1		1		1		0		0		1		0		1		x_3
0	1	0	0			0		1		0		1		1		1			
0	1	1	0				1		1		1		0		0				x_2x_3
1	0	0	0					0		0		1		0					
1	0	1	1						0		1		1						
1	1	0	1							1		0							x_1x_2
1	1	1	0								1								$x_1x_2x_3$

Получаем представление функции $f(x_1, x_2, x_3)$ полиномом Жегалкина, совпадающее с (3.27).


Приведем обоснование алгоритма 3.11 для $n = 2$. В общем случае доказательство можно провести аналогично.

Запишем общий вид полинома Жегалкина для функции двух переменных. Имеем

$$f(x_1, x_2) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_{12}x_1x_2.$$

Используя метод неопределенных коэффициентов, получаем:

$$\begin{cases} a_0 = f(0, 0), \\ a_1 = f(0, 0) \oplus f(1, 0), \\ a_2 = f(0, 0) \oplus f(0, 1), \\ a_{12} = f(0, 0) \oplus f(0, 1) \oplus f(1, 0) \oplus f(1, 1). \end{cases} \quad (3.28)$$

 **Замечание 3.19**

В следующем разделе «Быстрое вычисление полинома Жегалкина» будет подробно рассмотрен еще один, пожалуй, наиболее эффективный алгоритм для построения полинома Жегалкина — метод быстрого преобразования Фурье.

Значения в правой части (3.28) совпадают со значениями левой стороны треугольника Паскаля, построенного для случая $n = 2$.

Определение 3.49. Функцию $f(x_1, \dots, x_n)$ называют *линейной*, если она представима в виде линейного полинома Жегалкина, т. е. если существуют $a_0, a_1, \dots, a_n \in B$ такие, что

$$f(x_1, \dots, x_n) = a_0 \oplus a_1 x_1 \oplus \dots \oplus a_n x_n.$$

Класс всех линейных функций обозначим буквой L .

Лемма 3.13. Класс L не пуст и не совпадает со всем множеством булевых функций B_F .

Доказательство. Функции $0, 1, \bar{x}, x \oplus y \in L$, следовательно, $L \neq \emptyset$, а функции $xy, x \vee y, x \rightarrow y \notin L$, т. е. $L \neq B_F$. ■

 **Замечание 3.20**

Любая булева функция от одной переменной линейна.

Очевидна следующая теорема.

Теорема 3.26. Класс L является замкнутым.

Упражнение 3.8. Обозначим через L^n множество всех линейных булевых функций от n переменных. Докажите, что

$$|L^n| = 2^{n+1}.$$

Следующее утверждение называют *леммой о нелинейной функции*.

Лемма 3.14. Пусть $f(x_1, \dots, x_n)$ — нелинейная функция. Тогда замыкание класса $K = \{f(x_1, \dots, x_n), 0, 1, \bar{x}\}$ содержит произведение xy .

Доказательство. Так как $f(x_1, \dots, x_n)$ — нелинейная функция, то ее полином Жегалкина содержит хотя бы одно нелинейное слагаемое вида:

$$x_{i_1} \wedge x_{i_2} \wedge \dots \wedge x_{i_k}, \quad k \geq 2.$$

Среди них выберем самое короткое. Для переменных, входящих в конъюнкцию, положим $x_{i_3} = \dots = x_{i_k} = 1$, а те, которые в данное слагаемое не входят, положим равными нулю. Введем новые обозначения: $x = x_{i_1}$, $y = x_{i_2}$. Тогда функция f примет вид

$$f(x, y) = xy \oplus \alpha x \oplus \beta y \oplus \gamma, \quad \alpha, \beta, \gamma \in B.$$

Составим следующую таблицу (таблица 3.29).

Таблица 3.29.

α	β	γ	$f(x, y)$	xy
0	0	0	xy	$xy = f(x, y)$
0	0	1	$xy \oplus 1 = \overline{xy}$	$xy = \overline{f(x, y)}$
0	1	0	$xy \oplus y = \overline{xy}$	$xy = f(\overline{x}, y)$
0	1	1	$xy \oplus y \oplus 1 = \overline{\overline{xy}}$	$xy = \overline{f(\overline{x}, y)}$
1	0	0	$xy \oplus x = \overline{xy}$	$xy = f(x, \overline{y})$
1	0	1	$xy \oplus x \oplus 1 = \overline{\overline{xy}}$	$xy = \overline{f(x, \overline{y})}$
1	1	0	$xy \oplus x \oplus y = \overline{\overline{xy}}$	$xy = \overline{f(\overline{x}, \overline{y})}$
1	1	1	$xy \oplus x \oplus y \oplus 1 = \overline{\overline{\overline{xy}}}$	$xy = f(\overline{x}, \overline{y})$

Последний столбец таблицы 3.29 содержит представление функции xy через отрицание, нелинейную функцию $f(x_1, \dots, x_n)$ и подстановку в нее констант 0, 1. ■

Пример 3.58. Получим функцию xy с помощью констант, отрицания и нелинейной функции $f(x_1, x_2, x_3) = x_1 x_2 x_3 \oplus x_1 x_2 \oplus x_1$. Воспользуемся леммой 3.14 о нелинейной функции.

Заметим, что требование выбора самого короткого нелинейного слагаемого является существенным.

Возьмем, например, слагаемое $x_1x_2x_3$, тогда (полагая $x_3 = 1$) получим $f(x_1, x_2, 1) = x_1x_2 \oplus x_1x_2 \oplus x_1 = x_1$ и лемма 3.14 неприменима.

Сделаем правильный выбор и рассмотрим слагаемое x_1x_2 . Полагая $x_3 = 0$, получаем $f(x_1, x_2, 0) = x_1x_2 \oplus x_1$. Тогда $xy = f(x, \bar{y}, 0)$.

Задача 3.11. Какие из следующих функций линейны:

- 1) $x \downarrow y$;
- 2) $\overline{x \leftrightarrow \bar{y}}$;
- 3) $(x \leftrightarrow y) \leftrightarrow z$;
- 4) $x \rightarrow (y \rightarrow x)$.

3.2.4. Быстрое вычисление полинома Жегалкина

Рассмотрим алгоритм быстрого вычисления полинома Жегалкина на основе идей быстрого преобразования Фурье для размерности 2^n , анонсированный в предыдущем разделе [34].

Обозначим значения булевой функции $\{f[0], \dots, f[2^n - 1]\}$. Алгоритм вычислений коэффициентов полинома Жегалкина выглядит следующим образом.

Алгоритм 3.12. Быстрое вычисление полинома Жегалкина

```
// Инициализация
for k ← 0, 2n - 1 do
    a[k] = f[k]
end for

// Основной цикл
for i ← 1, n do
    for s ← 0, 2n-i - 1 do
        for l ← 0, 2i-1 - 1 do
            a[2is + 2i-1 + l] = a[2is + l] ⊕ a[2is + 2i - 1 + l]
        end for
    end for
end for
```

После работы алгоритма (3.12) в векторе $a[0 : 2^n - 1]$ будут находиться коэффициенты полинома Жегалкина.

Разберемся в алгоритме (3.12) более детально. Рассмотрим внешний цикл.

1 При $i = 1, l = 0$ вычисляем

$$a[2s + 1] = a[2s] \oplus a[2s + 1], \quad 0 \leq s < 2^{n-1}.$$

2 При $i = 2$

$$a[4s + 2 + l] = a[4s + l] \oplus a[4s + 2 + l], \quad l = 0, 1, \quad 0 \leq s < 2^{n-2}$$

и т. д.

n При $i = n, s = 0$

$$a[2^{n-1} + l] = a[l] \oplus a[2^{n-1} + l], \quad 0 \leq l < 2^{n-1}.$$

В алгоритме (3.12) используется только операция сложения по модулю 2 в количестве

$$\sum_{i=1}^n 2^{n-i} 2^{i-1} = n2^{n-1} = \frac{N \log_2 N}{2}$$

операций.

В таблице (3.30) приведены результаты работы алгоритма (3.12) для булевой функции трех переменных. В последнем столбце перечислены все мономы от трех переменных, а в предпоследнем — найденные значения коэффициентов при этих мономах.

Таблица 3.30.

№	x_1	x_2	x_3	$f(x_1, x_2, x_3)$	$i = 1$	$i = 2$	$i = 3$	МОНОМЫ
0	0	0	0	0	1	1	1	1
1	1	0	0	0	1	1	1	x_1
2	0	1	0	1	1	0	0	x_2
3	1	1	0	0	1	0	0	$x_1 x_2$
4	0	0	1	0	0	0	1	x_3
5	1	0	1	1	1	1	0	$x_1 x_3$
6	0	1	1	1	1	1	1	$x_2 x_3$
7	1	1	1	0	1	0	0	$x_1 x_2 x_3$

Получаем представление

$$f(x_1, x_2, x_3) = 1 \oplus x_1 \oplus x_3 \oplus x_2 x_3. \quad (3.29)$$

Рассмотрим обратную задачу — быстрое вычисление значений полинома Жегалкина по его коэффициентам.

Можно заметить, что эти значения можно найти, если воспользоваться алгоритмом (3.12), поменяв в нем местами идентификаторы f и a .

В таблице (3.31) приведены результаты вычисления значений полинома (3.29) по модифицированному таким образом алгоритму (3.12). Значения полинома (3.29) указаны в последнем столбце таблицы.

Таблица 3.31.

№	x_1	x_2	x_3	a	$i = 1$	$i = 2$	$i = 3$
0	0	0	0	1	1	1	1
1	1	0	0	1	0	0	0
2	0	1	0	0	0	1	1
3	1	1	0	0	0	0	0
4	0	0	1	1	1	1	0
5	1	0	1	0	1	1	1
6	0	1	1	1	1	0	1
7	1	1	1	0	1	0	0

3.2.5. Булевы функции в криптографии

В традиционных системах шифрования, переводящих открытое сообщение в зашифрованное с помощью секретного ключа, важную роль играют булевы функции. Рассмотрим схему поточного шифрования, когда каждый поступающий символ тут же преобразуется в символ шифртекста.

Исходный текст $\{x_i\}$, ключ $\{y_i\}$, шифртекст $\{z_i\}$ — бинарные наборы одинаковой длины. Схема поточного шифрования следующая:

$$z_i = x_i \oplus y_i, \quad i \in 0 : n.$$

При дешифровании схема та же, что и при шифровании, только исходный текст и шифртекст меняются местами. Шифр носит название *шифр Вернама* или *одноразовый шифрблокнот (one time pad)*.

Обычно дважды такой шифр не используют: при сложении двух шифртекстов, соответствующих одному ключу, получается сумма исходных текстов, что дает много информации об исходных текстах и даже часто поз-

воляет их прочесть. На практике шифровальщик при личной встрече снабжается блокнотом, каждая страница которого содержит ключ. Такой же блокнот есть и у принимающей стороны. Использованные страницы уничтожаются.

Историческая справка

Шифр назван в честь телеграфиста Гильберта Вернама, создавшего в 1917 г. телеграфный аппарат, который выполнял эту операцию автоматически — надо было только подать на него ленту с ключом. Не будучи шифровальщиком, тем не менее Вернам верно заметил важное свойство своего шифра — каждая лента должна использоваться только один раз и после этого уничтожаться.

В американском кинофильме 2001 г. «Пароль „Рыба-меч“» хакер (которого играет Хью Джекман) взламывает именно шифр Вернама.

В 1949 г. Клод Шеннон опубликовал работу, в которой доказал, что при совершенно случайном ключе, используемом один раз, шифр Вернама является абсолютно стойкой криптосистемой, то есть перехват шифртекста не дает никакой информации о переданном сообщении. Других шифров с этим свойством не существует. Это по сути означает, что шифр Вернама является самой безопасной криптосистемой из всех возможных.

Пример 3.59. Рассмотрим шифрование исходного текста $\{0, 1, 0, 1, 1, 1, 0, 1\}$ с ключом $\{1, 1, 0, 1, 0, 1, 1, 1\}$ — таблица 3.32.

Таблица 3.32.

Исходное	0 1 0 1 1 1	Шифрованное	0 1 1 0 0 1
Ключ	0 0 1 1 1 0	Ключ	0 0 1 1 1 0
Шифрованное	0 1 1 0 0 1	Исходное	0 1 0 1 1 1

(a) шифрование

(b) дешифрование

3.2.6. Самодвойственные функции

Перенесем принцип двойственности логики высказываний на булевы функции. Для определения воспользуемся соотношением (3.6).

Определение 3.50. Булеву функцию $g(x_1, \dots, x_n)$ называют *двойственной* к $f(x_1, \dots, x_n)$, если выполняется равенство

$$g(x_1, \dots, x_n) = \overline{f(\bar{x}_1, \dots, \bar{x}_n)}.$$

Пример 3.60. Функция xy двойственна $x \vee y$ и наоборот. Это следует из равенств (3.24) и (3.23).

Функция $x \rightarrow y$ двойственна функции $\bar{x}y$, поскольку

$$\overline{\overline{x} \rightarrow \bar{y}} \stackrel{14}{=} \overline{\overline{\bar{x}} \vee \bar{y}} \stackrel{13}{=} \overline{x \vee \bar{y}} \stackrel{12}{=} \bar{x}y.$$

Двойственной к функции $\max(x_1, \dots, x_n) = x_1 \vee \dots \vee x_n$ является функция

$$\overline{\overline{x_1} \vee \dots \vee \overline{x_n}} = x_1 \wedge \dots \wedge x_n = \min(x_1, \dots, x_n).$$

Задача 3.12. Определите, является ли функция f двойственной к функции g :

- 1) $f = x \leftrightarrow y$, $g = \bar{x}y \vee x\bar{y}$;
- 2) $f = (x \oplus y)z$, $g = (x \oplus y)(z \oplus 1)$.

Определение 3.51. Булеву функцию $f(x_1, \dots, x_n)$ называют *самодвойственной*, если выполняется равенство

$$f(x_1, \dots, x_n) = \overline{f(\bar{x}_1, \dots, \bar{x}_n)}. \quad (3.30)$$

Другими словами, функция самодвойственна, если она совпадает со своей двойственной. Заметим, что равенство (3.30) для определения самодвойственности равносильно равенству

$$\overline{f(x_1, \dots, x_n)} = f(\bar{x}_1, \dots, \bar{x}_n). \quad (3.31)$$

Пример 3.61. Покажем, что $f(x_1, x_2, x_3) = x_1x_2 \vee x_1x_3 \vee x_2x_3$ есть самодвойственная функция. Имеем

$$\begin{aligned} \overline{f(\bar{x}_1, \bar{x}_2, \bar{x}_3)} &= \overline{\bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_3} \stackrel{11}{=} \overline{\overline{x_1 \vee x_2} \vee \overline{x_1 \vee x_3} \vee \overline{x_2 \vee x_3}} \stackrel{12,13}{=} \\ &\stackrel{12,13}{=} (x_1 \vee x_2)(x_1 \vee x_3)(x_2 \vee x_3) \stackrel{7,1}{=} x_1x_2 \vee x_1x_3 \vee x_2x_3 = \\ &= f(x_1, x_2, x_3). \end{aligned}$$

Из последнего примера видно, что определять самодвойственность булевой функции на основании определения иногда достаточно сложно. Эта задача легко решается с помощью следующей теоремы.

Теорема 3.27. При лексикографическом упорядочивании строк таблицы значений самодвойственной функции последний столбец (значения функции) антисимметричен относительно середины таблицы.

Доказательство. Пусть $(\alpha_1, \dots, \alpha_n)$ — произвольная строка таблицы значений самодвойственной булевой функции, а $N(\alpha)$ ее номер, определенный согласно (3.3) в теореме 3.1.

Рассмотрим соответствующую двойственную строку $(\bar{\alpha}_1, \dots, \bar{\alpha}_n)$. Номер этой строки обозначим через $\bar{N}(\alpha)$. Легко показать, что

$$\bar{N}(\alpha) = 2^n - 1 - N(\alpha).$$

■

Следствие 3.4. Самодвойственная функция равна 1 на половине наборов своей таблицы значений.

Следствие 3.5. Самодвойственную функцию полностью определяет первая (вторая) половина столбца значений таблицы истинности.

Упражнение 3.9. Докажите, что среди булевых функций двух переменных нет самодвойственных, существенно зависящих от обеих переменных.

Класс всех самодвойственных функций обозначим буквой S .

Лемма 3.15. Класс S не пуст и не совпадает со всем множеством булевых функций B_F .

Доказательство. Функции $x, x_1x_2 \vee x_1x_3 \vee x_2x_3 \in S$, т. е. $S \neq \emptyset$, а функции $xy, x \rightarrow y, x \oplus y \notin S$, следовательно, $S \neq B_F$. ■

Справедлива теорема.

Теорема 3.28. S — замкнутый класс.

Доказательство. Поскольку равенство (3.30) выполняется для всех значений переменных, класс S замкнут относительно операции переименования переменных.

Пусть функции $f(x_1, \dots, x_k), g_1(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n)$ принадлежат S . Тогда, используя равенство (3.31), получаем, что

$$\begin{aligned} \overline{f(x_1, \dots, x_k)} &= f(\bar{x}_1, \dots, \bar{x}_k), \\ \overline{g_1(x_1, \dots, x_n)} &= g_1(\bar{x}_1, \dots, \bar{x}_n), \dots, \overline{g_k(x_1, \dots, x_n)} = g_k(\bar{x}_1, \dots, \bar{x}_n). \end{aligned}$$

Тогда, если $h(x_1, \dots, x_n) = f(g_1(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n))$, то

$$\begin{aligned} h(\bar{x}_1, \dots, \bar{x}_n) &= f(g_1(\bar{x}_1, \dots, \bar{x}_n), \dots, g_k(\bar{x}_1, \dots, \bar{x}_n)) = \\ &= f(\overline{g_1(x_1, \dots, x_n)}, \dots, \overline{g_k(x_1, \dots, x_n)}) = \\ &= \overline{f(g_1(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n))} = \overline{h(x_1, \dots, x_n)}. \end{aligned}$$

В силу равенства (3.31) $h(x_1, \dots, x_n)$ — самодвойственная функция. Следовательно, класс S замкнут относительно операции подстановки. ■

Упражнение 3.10. Обозначим через S^n множество всех самодвойственных булевых функций от n переменных. Докажите, что

$$|S^n| = 2^{2^{n-1}}.$$

Лемма 3.16 (о несамодвойственной функции). Рассмотрим несамодвойственную функцию $f(x_1, \dots, x_n)$. Тогда замыкание класса

$$K = \{f(x_1, \dots, x_n), \bar{x}\}$$

содержит константы 0, 1.

Доказательство. Так как $f(x_1, \dots, x_n)$ — несамодвойственная функция, существуют $a_1, \dots, a_n \in B$ такие, что

$$\overline{f(a_1, \dots, a_n)} \neq f(\bar{a}_1, \dots, \bar{a}_n).$$

Множество B содержит только два элемента. Поэтому из этого неравенства следует равенство

$$f(a_1, \dots, a_n) = f(\bar{a}_1, \dots, \bar{a}_n).$$

Для удобства обозначений (не умаляя общности рассуждений) предположим, что $a_1 = \dots = a_k = 0$, $a_{k+1} = \dots = a_n = 1$. Тогда последнее равенство можно записать так:

$$f(\underbrace{0, \dots, 0}_k, 1, \dots, 1) = f(\underbrace{1, \dots, 1}_k, 0, \dots, 0),$$

Рассмотрим функцию $g(x) = f(\underbrace{x, \dots, x}_k, \bar{x}, \dots, \bar{x})$. Заметим, что $g(x)$ принадлежит $[K]$. Имеем равенства:

$$\begin{aligned} g(0) &= f(\underbrace{0, \dots, 0}_k, \bar{0}, \dots, \bar{0}) = f(\underbrace{0, \dots, 0}_k, 1, \dots, 1) = \\ &= f(\underbrace{1, \dots, 1}_k, 0, \dots, 0) = f(\underbrace{1, \dots, 1}_k, \bar{1}, \dots, \bar{1}) = g(1). \end{aligned}$$

Следовательно, $g(x)$ — одна из констант, принадлежащая $[K]$. Поскольку K содержит отрицание, то и другая константа принадлежит $[K]$. ■

Пример 3.62. Является ли функция $f(x_1, x_2) = x_2(x_2 \rightarrow x_1)$ самодвойственной? Для получения ответа на вопрос согласно теореме 3.27 составим ее таблицу значений (таблица 3.33).

Заметим, что столбец значений функции не является антисимметричным относительно середины: равенство (3.30) не выполняется, например,

Таблица 3.33.

x_1	x_2	$f(x_1, x_2)$	x_1	x_2	$f(x_1, x_2)$
0	0	0	1	0	0
0	1	0	1	0	0

при $x_1 = 0, x_2 = 1$. Следовательно, функция $f(x_1, x_2)$ самодвойственной не является.

Пример 3.63. Определить, можно ли получить константы из функции $f(x_1, x_2, x_3) = x_3 \rightarrow x_1 x_2$ и функции отрицания.

Составим ее таблицу значений (таблица 3.34). Заметим, что $f \notin S$, так как нарушается условие самодвойственности на первом и последнем наборах: $f(0, 0, 0) = f(1, 1, 1) = 1$. Таким образом:

$$f(x, x, x) = x \rightarrow xx = \bar{x} \vee x = 1; \quad f(\bar{x}, \bar{x}, \bar{x}) = \bar{x} \rightarrow \bar{x}\bar{x} = x \vee \bar{x} = 1.$$

Используя отрицание, получаем вторую константу.

Таблица 3.34.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1	1	0	0	1
0	0	1	0	1	0	1	0
0	1	0	1	1	1	0	1
0	1	1	0	1	1	1	1

Задача 3.13. Определите, самодвойственны или нет следующие функции:

$$f(x_1, x_2, x_3) = (x_1 \oplus x_2)(x_2 \oplus x_3); \quad g(x_1, x_2, x_3) = (x_1 \vee x_2)(x_1 \vee x_3)(x_2 \vee x_3).$$

Упражнение 3.11. Докажите, что булева функция четности

$$p_n(x_1, x_2, \dots, x_n) = x_1 \oplus x_2 \oplus \dots \oplus x_n$$

является самодвойственной тогда и только тогда, когда n нечетно.

Упражнение 3.12. Определите, какие из линейных функций являются самодвойственными.

3.2.7. Монотонные функции

Введем отношение частичного нестрогого порядка (см. определение 2.77 в разделе «Отношения предпорядка и порядка») на множестве булевых наборов длиной n .

Определение 3.52. Будем считать, что

$$\alpha = (\alpha_1, \dots, \alpha_n) \preceq \beta = (\beta_1, \dots, \beta_n), \text{ если } \alpha_1 \leq \beta_1, \dots, \alpha_n \leq \beta_n, \alpha_i, \beta_i \in B.$$

Например, $(1, 0, 0, 1) \preceq (1, 0, 1, 1)$, $(1, 0, 1, 0) \preceq (1, 0, 1, 1)$. В то же время неверно, что $(1, 0, 0, 1) \preceq (1, 0, 1, 0)$.

Определение 3.53. Булеву функцию $f(x_1, \dots, x_n)$ называют *монотонной*, если для любых двух наборов $\alpha = (\alpha_1, \dots, \alpha_n)$, $\beta = (\beta_1, \dots, \beta_n)$ из условия $\alpha \preceq \beta$ следует, что $f(\alpha) \leq f(\beta)$.

Класс всех монотонных функций обозначим буквой M .

Лемма 3.17. Класс M не пуст и не совпадает со всем множеством булевых функций B_F .

Доказательство. Функции $1, 0, xy, x \vee y \in M$, т. е. $M \neq \emptyset$, а функции $\bar{x}, x \rightarrow y, x \oplus y \notin M$, следовательно, $M \neq B_F$. ■

Проверка монотонности булевой функции по определению достаточно трудоемка. Используем для этого рассмотренный ранее подход графического представления упорядоченных множеств — диаграмму Хассе (раздел 2.3.7).

Пример 3.64. Проверим монотонность функции

$$f(x_1, x_2, x_3) = x_1 \oplus x_2 x_3.$$

Составим таблицу значений функции (таблица 3.35).

Таблица 3.35.

x_1	x_2	x_3	$x_1 \oplus x_2 x_3$	x_1	x_2	x_3	$x_1 \oplus x_2 x_3$
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	1
0	1	1	1	1	1	1	0

Составим диаграмму Хассе частично упорядоченного множества булевых наборов длиной 3 (рисунок 3.13).

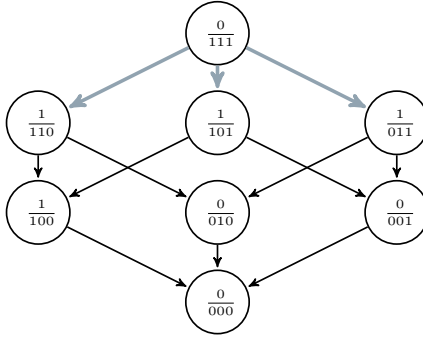


Рис. 3.13.

В вершинах этого графа проставим значения исследуемой функции согласно таблице 3.35 и соответствующие наборы. Монотонность функции означает, что при движении по любому ориентированному пути в диаграмме Хассе значение функции не должно увеличиваться.

Это свойство нарушается при переходе от вершины $(1, 1, 1)$ к вершинам $(1, 1, 0)$, $(1, 0, 1)$ и $(0, 1, 1)$. Соответствующие дуги в графе на рисунке 3.13 показаны линиями серого цвета.

Таким образом, функция f немонотонна.

Задача 3.14. Определите монотонность следующих функций:

$$a) xy \oplus y \oplus x; \quad b) x \rightarrow (x \rightarrow y); \quad c) xy \vee xz \vee yz.$$

Упражнение 3.13. Докажите монотонность функций:

$$\max(x_1, \dots, x_n) \text{ и } \min(x_1, \dots, x_n),$$

определенных в примере 3.60.

Упражнение 3.14. Определите, какие из линейных функций являются монотонными.

Теорема 3.29. Класс M — замкнутый класс.

Доказательство. Пусть $f(x_1, \dots, x_n) \in M$ и y_1, \dots, y_n — новые переменные. Возьмем два набора значений переменных y_1, \dots, y_n :

$$\alpha = (\alpha_1, \dots, \alpha_n) \preceq \beta = (\beta_1, \dots, \beta_n). \quad (3.32)$$

Но эти векторы будут наборами значений переменных x_1, \dots, x_n , и поэтому выполняется неравенство $f(\alpha) \leq f(\beta)$. Это означает, что класс M замкнут относительно операции переименования переменных.

Пусть $f(x_1, \dots, x_k)$, $g_1(x_1, \dots, x_n), \dots, g_n(x_1, \dots, x_n)$ — монотонные функции и α и β — два набора значений переменных x_1, \dots, x_n , удовлетворяющих (3.32). Рассмотрим подстановку

$$h(x_1, \dots, x_n) = f(g_1(x_1, \dots, x_n), \dots, g_n(x_1, \dots, x_n)).$$

Введем обозначения:

$$\gamma_1 = g_1(\alpha), \dots, \gamma_k = g_k(\alpha), \quad \zeta_1 = g_1(\beta), \dots, \zeta_k = g_k(\beta).$$

В силу монотонности функций g_1, \dots, g_k имеем, что $\gamma_1 \leq \zeta_1, \dots, \gamma_k \leq \zeta_k$. Тогда

$$\begin{aligned} h(\alpha_1, \dots, \alpha_n) &= f(g_1(\alpha_1, \dots, \alpha_n), \dots, g_k(\alpha_1, \dots, \alpha_n)) = f(\gamma_1, \dots, \gamma_k) \leq \\ &\leq f(\zeta_1, \dots, \zeta_k) = f(g_1(\beta_1, \dots, \beta_n), \dots, g_k(\beta_1, \dots, \beta_n)) = \\ &= h(\beta_1, \dots, \beta_n). \end{aligned}$$

Знак неравенства поставлен в силу монотонности функции f . Таким образом, доказано, что класс M замкнут относительно операции подстановки. ■

Замечание 3.21

В случаях изучения классов T_0, T_1, L, S достаточно легко можно было определить число функций класса от n переменных (см. упр. 3.5, 3.6, 3.8, 3.10). Вопрос о числе монотонных функций от n переменных не решен до настоящего времени. Существуют лишь оценки этого числа.

Упражнение 3.15. Докажите, что функция, двойственная монотонной, сама монотонна.

Упражнение 3.16. Докажите, что монотонными являются те и только те функции, которые или являются константами, или допускают представление в КНФ или ДНФ, не содержащей отрицаний.

Упражнение 3.17. Докажите, что монотонная функция, не сохраняющая ноль (единицу), равна тождественно единице (нулю).

Лемма 3.18 (о немонотонной функции). Пусть $f(x_1, \dots, x_n) \notin M$. Тогда замыкание класса

$$K = \{f(x_1, \dots, x_n), 0, 1\}$$

содержит \bar{x} .

Доказательство. Поскольку $f(x_1, \dots, x_n)$ немонотонна, то существуют наборы значений переменных $\alpha = (\alpha_1, \dots, \alpha_n)$, $\beta = (\beta_1, \dots, \beta_n)$ такие, что $\alpha < \beta$, но $f(\alpha) > f(\beta)$. Неравенство $f(\alpha) > f(\beta)$ означает, что $f(\alpha) = 1$ и $f(\beta) = 0$. Пусть наборы α и β отличаются k компонентами ($k \geq 1$). Тогда в α на этих местах стоят нули, а в β — единицы. Заменяя указанные компоненты по одной (по старшинству), получим цепочку наборов:

$$\alpha < \alpha^1 < \alpha^2 < \dots < \alpha^{k-1} < \beta,$$

где соседние наборы различаются только одной компонентой. Очевидно, что найдутся соседние наборы α^j , α^{j+1} такие, что $f(\alpha^j) = 1$, $f(\alpha^{j+1}) = 0$. Пусть они различаются i -й компонентой. Тогда $\alpha_i^j = 0$, $\alpha_i^{j+1} = 1$, а остальные компоненты одинаковы. Введем функцию

$$g(x) = f(\alpha_1^j, \alpha_2^j, \dots, \alpha_{i-1}^j, x, \alpha_{i+1}^j, \alpha_{i+2}^j, \dots, \alpha_n^j).$$

По условиям леммы $g(x) \in [K]$. Тогда

$$g(0) = g(\alpha_i^j) = f(\alpha^j) = 1, \quad g(1) = g(\alpha_i^{j+1}) = f(\alpha^{j+1}) = 0,$$

т. е. $g(x) = \bar{x}$. ■

Пример 3.65. Построим функцию отрицания из немонотонной функции и констант. Пусть функция $f(x_1, x_2, x_3)$ задана своей таблицей значений (таблица 3.36).

Таблица 3.36.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	0	0	1	1
0	1	0	0	0	1	1	1
1	0	0	1	1	0	1	0
1	1	0	0	1	1	1	1

Очевидно, что функция $f(x_1, x_2, x_3)$ не является монотонной. Действительно, монотонность нарушается на наборах $(0, 0, 1)$ и $(1, 0, 1)$. Эти наборы различаются только первой компонентой. Тогда согласно доказательству леммы 3.18 определяем функцию $g(x) = f(x, 0, 1)$. Нетрудно убедиться в том, что $g(0) = 1$, а $g(1) = 0$, т. е. функция $g(x)$ является отрицанием.

Пример 3.66. Определим, можно ли построить функцию \bar{x} из функции $f(x_1, x_2, x_3) = x_1 \oplus x_2x_3$ и констант.

Функция f немонотонна, так как $f(1, 0, 0) = 1$, $f(1, 1, 1) = 0$, тогда по лемме о немонотонной функции из этой функции можно получить функцию отрицания.

Выберем пару соседних наборов, на которых нарушено условие монотонности. Имеем $f(1, 1, 0) = 1$, $f(1, 1, 1) = 0$. Выбирая соответствующую замену переменных, получаем, что

$$f(1, 1, x) = 1 \oplus x = \bar{x}.$$

Замечание 3.22

Монотонные булевы функции были определены как монотонно неубывающие (определение 3.53). При этом определении класс M является замкнутым (см. теорему 3.29). Возникает вопрос: можно ли было определить этот класс как множество монотонно невозрастающих булевых функций и при этом сохранить свойство замкнутости?

Отрицательный ответ на этот вопрос дает рассмотрение функции $f(x) = \bar{x}$. Отрицание \bar{x} — монотонно убывающая функция. Подстановка функции в себя $f(f(x)) = \bar{\bar{x}} = x$ монотонно возрастает.

3.2.8. Теорема Поста

Установим необходимый и достаточный критерий полноты класса булевых функций, который называют теоремой Поста.

Определение 3.54. Классы T_0, T_1, L, S, M называют основными замкнутыми классами (классами Поста) булевых функций.

Теорема 3.30 (Пост). Класс булевых функций K является полным тогда и только тогда, когда он не содержится ни в одном из основных замкнутых классов.

Доказательство. Установим необходимость. Пусть K — полный класс. Предположим, что K содержится в одном из основных замкнутых классов, например $K \subseteq T_0$. Тогда из леммы 3.12 следует, что класс T_0 также полный, т. е. $[T_0] = B_F$.

Но согласно теореме 3.22 справедливо равенство $T_0 = [T_0]$, а по лемме 3.10 имеем $T_0 \neq B_F$. Противоречие показывает, что K не содержится в T_0 . Аналогично доказывается, что K не содержится и в других основных замкнутых классах.

Перейдем к доказательству достаточности. Пусть K не содержится ни в одном из классов T_0, T_1, S, M и L . Тогда класс K содержит функции

$$f_0 \notin T_0, f_1 \notin T_1, f_2 \notin S, f_3 \notin M, f_4 \notin L. \quad (3.33)$$

Докажем, что $[K]$ содержит \bar{x} и xy . Так как $f_0 \notin T_0$, то $f_0(0, \dots, 0) = 1$. Относительно значения $f_0(1, \dots, 1)$ рассмотрим два случая.

1. Пусть $f_0(1, \dots, 1) = 0$. Рассмотрим функцию $g(x) = f_0(x, \dots, x)$. Тогда $g(0) = f_0(0, \dots, 0) = 1$ и $g(1) = f_0(1, \dots, 1) = 0$, т. е. $g(x) = \bar{x}$. Это означает, что замыкание класса K содержит отрицание. Классу K принадлежит несамодвойственная функция f_2 . По лемме о несамодвойственной функции замыкание класса K содержит константы. Поскольку K содержит нелинейную функцию f_4 , то в силу леммы о нелинейной функции $[K]$ содержит произведение xy . Мы доказали, что в первом случае $[K]$ содержит \bar{x} и xy .

2. Предположим, что $f_0(1, \dots, 1) = 1$. Тогда, если $g(x) = f_0(x, \dots, x)$ и $h(x) = f_1(g(x), \dots, g(x))$, то

$$\begin{aligned} g(0) &= f_0(0, \dots, 0) = 1 = f_0(1, \dots, 1) = g(1), \\ h(0) &= f_1(g_1(0), \dots, g_1(0)) = f_1(1, \dots, 1) = 0 = f_1(g(1), \dots, g(1)) = h(1), \end{aligned}$$

поскольку f_1 не сохраняет 1. Это означает, что константы принадлежат $[K]$.

Так как K содержит немонотонную функцию, то по лемме о немонотонной функции $[K]$ принадлежит отрицание. Далее, f_4 — нелинейная функция из K , поэтому по лемме о нелинейной функции $[K]$ содержит и xy .

Итак, в обоих случаях замыканию $[K]$ принадлежат \bar{x} и xy . Поскольку по теореме 3.24 $\{\bar{x}, xy\}$ — полный класс, то K согласно лемме 3.10 также полный класс. ■

Приведенное доказательство теоремы Поста иллюстрирует рисунок 3.14. Линиями светло-серого цвета показан первый, рассмотренный при доказательстве случай, линии темно-серого цвета соответствуют второму случаю.

Следствие 3.6. Каждый полный класс содержит полный подкласс, состоящий не более чем из четырех функций.

Доказательство. Пусть K — полный класс. Тогда по теореме 3.30 класс K не содержится ни в одном из основных замкнутых классов. Согласно доказательству теоремы 3.30 в K найдутся функции f_0, f_1, f_2, f_3, f_4 , удовлетворяющие (3.33).

При рассмотрении первого случая были использованы три функции f_0, f_2 и f_4 , т. е. $\{f_0, f_2, f_4\}$ — полный подкласс класса K . Во втором случае использованы четыре функции: f_0, f_1, f_3 и f_4 . Тогда $\{f_0, f_2, f_3, f_4\}$ — полный подкласс класса K . Итак, в обоих случаях в K найдется полный подкласс, содержащий не более четырех функций. ■

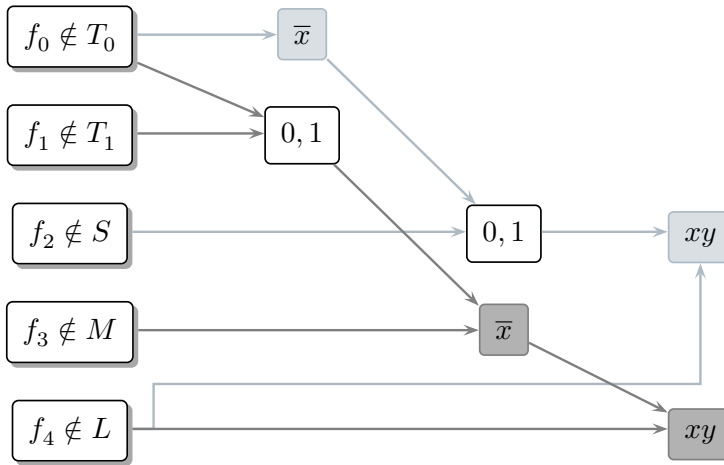


Рис. 3.14.

Пример 3.67. Существует полный подкласс, состоящий точно из четырех функций. В качестве примера рассмотрим класс

$$K = \{0, 1, xy, x \oplus y \oplus z\}.$$

Очевидно, что $0 \notin T_1$, $1 \notin T_0$. Функция xy не является ни самодвойственной, ни линейной. Нетрудно проверить, что $x \oplus y \oplus z$ — немонотонная функция. Следовательно, по теореме Поста K — полный класс.

В то же время любой его собственный подкласс полным не является. Действительно, если удалить функцию 0 , то оставшиеся функции будут сохранять 1 , если удалить 1 , то будут сохранять 0 . Класс функций K без функции xy состоит из линейных функций, а без последней функции — из монотонных.

Определение 3.55. Замкнутый класс K называют *предполным* если K — неполный класс, но для любой функции $f \notin K$ класс $K \cup \{f\}$ является полным.

Теорема 3.31. Предполными являются классы T_0 , T_1 , S , M и L и только они.

Доказательство. Необходимость докажем на примере класса T_0 . В нем содержатся несамодвойственная и нелинейная функция xy , немонотонная функция $x \oplus y$ и функция 0 , не сохраняющая 1 . Пусть $f \notin T_0$. Рассмотрим класс $\widehat{T}_0 = \{0, xy, x \oplus y\} \cup \{f\}$. Легко показать, что \widehat{T}_0 не содержится ни в одном из основных замкнутых классов и, следовательно, является полным.

Поскольку $\widehat{T}_0 \subseteq T_0 \cup \{f\}$, то по лемме 3.12 класс $T_0 \cup \{f\}$ также является полным.

Докажем достаточность. Пусть K — предполный класс. Поскольку класс K не является полным, то K содержится в одном из основных замкнутых классов. Для определенности предположим, что $K \subseteq T_0$. Если $K \neq T_0$, то существует функция f такая, что $f \notin K$ и $f \in T_0$. Тогда $K \cup \{f\} \subseteq T_0$ и класс $K \cup \{f\}$ не может быть полным. Следовательно, $K = T_0$. ■

Упражнение 3.18. Булеву функцию называют *полной (обобщенной функцией Шеффера)*, если класс, единственным элементом которого является эта функция, будет полным. Докажите, что штрих Шеффера и стрелка Пирса — единственные полные функции среди всех булевых функций двух переменных.

Определение 3.56. Систему функций Ψ называют *базисом* замкнутого класса K , если $[\Psi] = K$, но замыкание любой собственной подсистемы Ψ уже не совпадает с K .

Пример 3.68. Рассмотрим полный класс $K_4 = \{xy, x \oplus y, 1, 0\}$ из теоремы 3.24. Согласно определению 3.48 это полиномы Жегалкина. Но это не базис, так как $1 \oplus 1 = 0$ и базис K_4 состоит из трех функций $\{xy, x \oplus y, 1\}$.

Упражнение 3.19. Докажите, что система $\Psi = \{0, 1, xy, x \vee y\}$ есть базис класса M .

Упражнение 3.20. Докажите, что если система булевых функций полна, то будет полной и система, состоящая из двойственных функций.

Применим теорему Поста для исследования полноты системы булевых функций $K = \{f_1, \dots, f_n\}$. Для проверки условий теоремы составим таблицу, которую называют *таблицей Поста* (таблица 3.37).

Таблица 3.37.

	T_0	T_1	L	S	M
f_1					
...
f_n					

В клетки таблицы ставятся знаки: «+», если функция, стоящая в строке принадлежит классу Поста, стоящему в данном столбце, «−» — в противном случае. Для полноты системы K необходимо и достаточно (в силу теоремы Поста), чтобы в каждом столбце стоял хотя бы один знак «−».

Пример 3.69. Исследуем на полноту системы булевых функций $K_1 = \{x \oplus y, xy, 1, 0\}$ и $K_2 = \{x \oplus y \oplus z, xy, x \rightarrow y\}$.

Составим таблицы Поста для системы K_1 (таблица 3.38a) и системы K_2 (таблица 3.38b).

Таблица 3.38.

	T_0	T_1	L	S	M
$x \oplus y$	+	-	+	-	-
xy	+	+	-	-	+
1	-	+	+	-	+
0	+	-	+	-	+

(a)

	T_0	T_1	L	S	M
$x \oplus y \oplus z$	+	+	+	+	-
xy	+	+	-	-	+
$x \rightarrow y$	-	+	-	-	-

(b)

Система K_1 полна, но не является базисом. Базис образует подсистема $\{x \oplus y, xy, 1\}$.

Система K_2 не является полной, так как $K_2 \subset T_1$. Для получения полной системы нужно добавить любую функцию, не сохраняющую 1 (например, \bar{x}).

Пример 3.70. Покажем полноту системы функций

$$G = \{f(x, y, z) = xy \rightarrow z, g(x, y) = x \oplus y\}.$$

Проиллюстрируем поэтапное доказательство теоремы Поста, т. е. выразим константы, отрицание и конъюнкцию через функции системы G .

Построим таблицу Поста (таблица 3.39) для системы функций G . Полином Жегалкина для $f(x, y, z)$ имеет вид

$$xy \rightarrow z = xyz \oplus xy \oplus 1. \quad (3.34)$$

Принадлежность функций f, g остальным классам Поста достаточно очевидна.

Таблица 3.39.

	T_0	T_1	L	S	M
$xy \rightarrow z$	-	+	-	-	-
$x \oplus y$	+	-	+	-	-

По теореме Поста делаем вывод, что система функций G полна.

1. Получение констант. Имеем:

$$f(0, 0, 0) = f(1, 1, 1) = 1 : f(x, x, x) = xx \rightarrow x = 1, \quad (3.35)$$

Вторую константу 0 получим из функции

$$g(x, y) \notin T_1 : g(f(x, x, x), f(x, x, x)) = (xx \rightarrow x) \oplus (xx \rightarrow x) = 0. \quad (3.36)$$

2. Получение отрицания. По лемме о немонотонной функции

$$g(1, x) = 1 \oplus x = \bar{x}. \quad (3.37)$$

Подставим $f(x, x, x)$ вместо 1 в (3.37), согласно (3.35) получим

$$g(f(x, x, x), x) = (xx \rightarrow x) \oplus x = \bar{x}.$$

3. Получение конъюнкции. По лемме о нелинейной функции, используя (3.34) для функции f , имеем

$$xy = \overline{f(x, y, 0)}.$$

Далее, используя (3.35)–(3.37), получаем цепочку равенств:

$$\begin{aligned} xy &= f(x, y, g(f(x, x, x), f(x, x, x))) \oplus 1 = \\ &= (xy \rightarrow ((xx \rightarrow x) \oplus (xx \rightarrow x))) \oplus (xx \rightarrow x) \end{aligned}$$

Таким образом, константы, отрицание и конъюнкцию выразили через функции системы G .

Задача 3.15. Выразить функцию $h = x \downarrow y$ через функции системы G из примера 3.70.

Выразим функцию h через конъюнкцию и отрицание: $h = \overline{xy}$. Используя выражения для отрицания и конъюнкции через функции системы G , которые были получены в примере 3.70, получаем

$$\begin{aligned} h &= g(f(x, x, x), f(g(f(x, x, x), x), y, g(f(x, x, x), f(x, x, x)))) = \\ &= (xx \rightarrow x) \oplus (((xx \rightarrow x) \oplus x)y \rightarrow ((xx \rightarrow x) \oplus (xx \rightarrow x))). \end{aligned}$$

Задача 3.16. Значения булевых функций $f(x_1, x_2, x_3)$ и $g(x_1, x_2, x_3)$ заданы таблицей 3.40. Исследовать на полноту систему булевых функций $\{f, g\}$, в случае полноты найти все базисы данной системы.

Таблица 3.40.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$	$g(x_1, x_2, x_3)$
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	0

Составим таблицу Поста для системы $\{f, g\}$ (таблица 3.41).

Таблица 3.41.

	T_0	T_1	L	S	M
$f(x_1, x_2, x_3)$	—	+	—	—	—
$g(x_1, x_2, x_3)$	+	—	—	—	—

Таким образом, система $\{f, g\}$ полна и является базисом. Ни одну из функций нельзя удалить из системы, не теряя при этом свойство полноты.

Задача 3.17. Исследовать на полноту систему булевых функций

$$\begin{cases} f_1(x, y) &= x \oplus y, \\ f_2(x, y, z) &= xy \oplus z, \\ f_3(x, y, z) &= x \oplus y \oplus z \oplus 1, \\ f_4(x, y, z) &= xy \oplus yz \oplus xz \end{cases}$$

и в случае полноты найти все базисы данной системы.

Составим таблицу Поста для данного случая (таблица 3.42).

Из анализа таблицы получаем, что система полна и имеет два собственных базиса: $\{f_2, f_3\}$ и $\{f_1, f_3, f_4\}$.

Задача 3.18. Будут ли полными следующие системы функций:

Таблица 3.42.

	T_0	T_1	L	S	M
f_1	+	-	+	-	-
f_2	+	-	-	-	-
f_3	-	-	+	+	-
f_4	+	+	-	+	+

- 1) $K_1 = \{x \oplus y, x \vee y \vee \bar{z}\}$;
- 2) $K_2 = \{\bar{x}, xy \vee xz \vee yz\}$?

3.2.9. Разложение Шеннона

Определение 3.57. *Разложением Шеннона, или декомпозицией Шеннона, по переменной x_i называется метод представления булевой функции от n переменных в виде суммы двух подфункций от $(n - 1)$ остальных переменных:*

$$f(x_1, \dots, x_n) = x_i f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) \vee \bar{x}_i f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n). \quad (3.38)$$

Тождество (3.38) остается справедливым и при замене дизъюнкции на \oplus , так как оба слагаемых ортогональны — никогда не принимают истинное значение одновременно.

$$f(x_1, \dots, x_n) = x_i f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) \oplus \bar{x}_i f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n).$$

Для каждой из оставшихся функций от меньшего числа переменных можно продолжить разложение (3.38) по одной из оставшихся переменных.

Функция $f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$ в (3.38) называется положительным дополнением, а функция $f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$ — отрицательным дополнением.

Историческая справка

Клод Элюд Шеннон (1916–2001) — американский инженер и математик. Является основателем теории информации, нашедшей применение в современных высокотехнологических системах связи. Шеннон внес огромный вклад в теорию вероятностных схем, теорию автоматов и теорию систем управления — области наук, входящие в понятие «кибернетика». В 1948 г. предложил использовать слово «бит» для обозначения наименьшей единицы информации.

Представление (3.38) носит имя Шеннона, однако гораздо раньше оно было получено Джорджем Булем, а сама возможность такого разложения по любой из переменных непосредственно вытекает из возможности определения любой булевой функции с помощью таблицы истинности.

Декомпозиция (3.38) по переменной x_i основана на том, что таблицу истинности для булевой функции от n переменных можно разбить на две части таким образом, чтобы в первой части оказались только те булевы наборы, на которых переменная x_i всегда принимает значение 1, а во второй части остались только наборы, на которых переменная x_i всегда принимает значение 0.

Пример 3.71. Получим разложение Шеннона для булевой функции, заданной лексикографически упорядоченным вектором значений

$$f(x_1, x_2, x_3) = \{1, 1, 0, 1, 0, 1, 0, 1\}$$

по переменной x_1 .

Запишем булеву функцию $f(x_1, x_2, x_3)$ в виде СДНФ:

$$f(x_1, x_2, x_3) = \overline{x_1} \overline{x_2} \overline{x_3} \vee \overline{x_1} \overline{x_2} x_3 \vee \overline{x_1} x_2 x_3 \vee x_1 \overline{x_2} x_3 \vee x_1 x_2 x_3.$$

Из последнего равенства, применяя дистрибутивность по x_1 и $\overline{x_1}$, получаем требуемое разложение:

$$\begin{aligned} f(x_1, x_2, x_3) &= x_1 (\overline{x_2} x_3 \vee x_2 x_3) \vee \overline{x_1} (\overline{x_2} \overline{x_3} \vee \overline{x_2} x_3 \vee x_2 x_3) = \\ &= x_1 f(1, x_2, x_3) \vee \overline{x_1} f(0, x_2, x_3). \end{aligned}$$

Аналогично выполняется разложение функции $f(x_1, x_2, x_3)$ по переменной x_2 или x_3 :

$$\begin{aligned} f(x_1, x_2, x_3) &= x_2 (\overline{x_1} x_3 \vee x_1 x_3) \vee \overline{x_2} (\overline{x_1} \overline{x_3} \vee \overline{x_1} x_3 \vee x_1 x_3) = \\ &= x_3 (\overline{x_1} x_2 \vee x_1 \overline{x_2} \vee x_1 x_2) \vee \overline{x_3} (\overline{x_1} \overline{x_2} \vee \overline{x_1} x_3). \end{aligned}$$

3.2.10. Бинарные диаграммы решений

В современных задачах, решаемых с применением булевой алгебры, применяются все более сложные булевы функции. Отсюда возникает задача удобного представления булевой функции. В связи с чем возникают следующие требования к такому представлению:

- 1) оно должно быть каноническим;
- 2) оно должно быть минимальным (по числу переменных);

3) этим представлением должно быть легко манипулировать.

Из курса нам известно несколько канонических представлений булевой функции (например, ДНФ и КНФ, таблица истинности). Но данные представления перестали удовлетворять вышеизложенным требованиям в связи со сложностью самих булевых функций. Рассмотрим новый достаточно эффективный метод представления булевой функции — бинарную диаграмму решений.

Основной идеей для создания такой структуры данных послужило рассмотренное ранее разложение Шеннона. Любую булеву функцию по одной из входных переменных можно разделить на две подфункции, называемые положительным и отрицательным дополнением, из которых по принципу *if-then-else* выбирается только одна подфункция в зависимости от значения входной переменной (по которой выполнялось разложение Шеннона). Представляя каждую такую подфункцию в виде поддерева и продолжая разложение по оставшимся входным переменным, можно получить бинарную диаграмму решений.

Определение 3.58. *Бинарная диаграмма решений*¹ — это представление булевой функции в виде корневого ориентированного дерева.

Из каждого узла дерева идут по два ребра: одно соответствует нулевому значению соответствующей переменной (будем изображать его штриховой линией), а другое — единичному значению этой переменной (оно изображается сплошной линией). Таким образом, с каждой ветвью от корня до листа сопоставляется набор значений переменных, а листья помечаются соответствующими значениями функции.

Введем следующие обозначения.

Каждая нетерминальная (невисячая) вершина графа v помечена переменной $index(v)$ (обычно это переменные заголовка булевой функции x_i) и имеет две исходящие дуги к двум сыновьям: левую — $left(v)$ (будем ее изображать штриховой линией) в случае, когда переменной x_i присваивается значение 0, и правую — $right(v)$ (показана сплошной линией) в случае, когда переменной x_i присваивается значение 1. Каждая терминальная (висячая) вершина графа помечена либо 0, либо 1.

Для заданной таблицы истинности значение функции, реализованной бинарной диаграммой решений, определяется с помощью продвижения по пути от корня к терминальной вершине по ветвям, соответствующим присваиваемым переменным значениям. В качестве значения функции берется метка найденной терминальной вершины.

¹ Binary Decision Diagrams, BDD.

Дуги на диаграмме обычно упорядочены таким образом, что значения терминальных вершин (слева направо) соответствуют их вхождениям в таблицу истинности (сверху вниз).

Обычно рассматривают *упорядоченные бинарные диаграммы решений*¹, где задан линейный порядок на множество переменных и требуется, чтобы для любой вершины u и любого ее преемника v , не являющегося терминальной вершиной, соответствующие им переменные были упорядочены по возрастанию индексов переменных булевой функции $f(x_1, \dots, x_n)$, например $x_1 \prec x_2 \prec \dots \prec x_n$.

Замечание 3.23

Дональд Кнут назвал бинарные диаграммы решений одной из действительно фундаментальных структур данных, которые появились за последние двадцать пять лет. На **YouTube** можно посмотреть его замечательную лекцию **Fun With Binary Decision Diagrams (BDDs)**, прочитанную в Стэнфордском университете (июнь 2008 г.).

В настоящее время OBDD широко используются в системах автоматизированного проектирования (САПР) для синтеза логических схем, формальной верификации, задач распознавания и классификации текстов.

Поддержка OBDD присутствует во всех современных языках высокого уровня.

На рисунке 3.15 изображено представление булевой функции, заданное лексикографически упорядоченным вектором значений:

$$f(x_1, x_2, x_3) = \{0, 0, 0, 1, 0, 1, 0, 1\}.$$

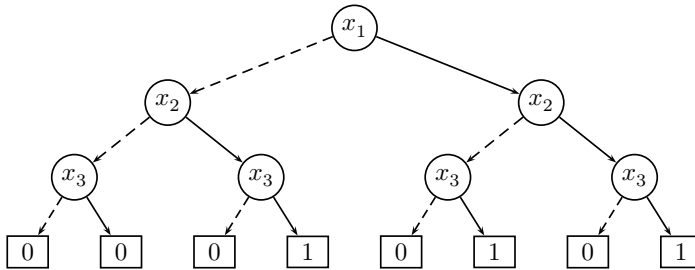


Рис. 3.15.

Определяются три правила преобразования OBDD (редукции графа), не изменяющих значение представляемой булевой функции.

1 Удаление дублирующих терминальных переменных.

Удаляются все терминальные вершины с заданной меткой (значением булевой функции), кроме одной, а все дуги, ведущие к удаленным вершинам, перенаправляются в оставшуюся вершину.

¹ В англоязычной литературе такое представление называется Ordinary Binary Decision Diagrams, сокращенно OBDD.

Рисунок 3.16 иллюстрирует применение первого правила к дереву решений, представленному на рисунке 3.15. Применение правила преобразования сокращает количество терминальных вершин с 8 до 2.

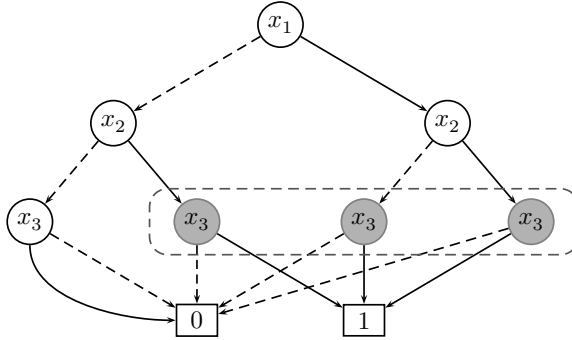


Рис. 3.16.

2 Удаление дублирующих нетерминальных вершин. Если для нетерминальных вершин u и v выполняются следующие условия:

$$\text{index}(u) = \text{index}(v), \text{left}(u) = \text{left}(v), \text{right}(u) = \text{right}(v),$$

то одна из этих вершин удаляется, а все дуги, входящие в нее, перенаправляются во вторую вершину.

На рисунке 3.17а показано применение второго правила к дереву решений после шага 1 (рисунок 3.16). Применение правила удаляет две вершины с переменной x_3 (три вершины с меткой x_3 , помеченные серым цветом на рисунке 3.16, *сливаются в одну*) и дуги к терминальным вершинам с метками 0 и 1.

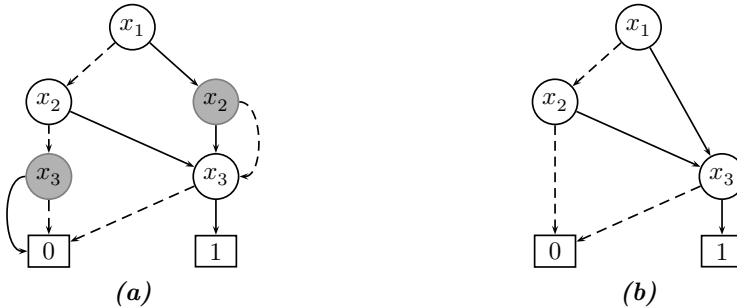


Рис. 3.17.

3 Удаление избыточных проверок. Если для нетерминальной вершины v верно $\text{left}(v) = \text{right}(v)$, то вершина v удаляется, а все дуги, входящие в нее, перенаправляются в $\text{left}(v)$.

На рисунке 3.17*b* показано применение третьего правила к дереву решений после шага 2 (рисунок 3.17*a*).

Применение третьего правила удаляет две вершины: одну с переменной x_3 и одну с переменной x_2 (помеченные серым цветом на рисунке 3.17*b*).

Замечание 3.24

В общем случае правила преобразования 2 и 3 должны применяться неоднократно, поскольку каждое применение преобразования может привести к появлению новых возможностей для других преобразований.

Теорема 3.32. Относительно OBDD-представления очевидны следующие утверждения.

1. Любая булева функция при любом заданном порядке на переменных имеет OBDD-представление.
2. Для любого заданного порядка на переменных две упорядоченные бинарные диаграммы решений одной и той же функции изоморфны.

Следствие 3.7. Из теоремы 3.32 вытекает несколько важных следствий.

1. Легко может быть проверена выполнимость. Функция является выполнимой в том и только том случае, когда ее OBDD-представление не сводится к одиночной терминальной вершине с меткой 0.
2. Любая тавтологическая функция должна иметь терминальную вершину с меткой 1 в качестве своего OBDD-представления.
3. Если функция не зависит от переменной x , то ее OBDD-представление не может иметь вершин, помеченных x .

Таким образом, как только построены представления функций в виде OBDD, многие их характеристики становятся легко проверяемыми. Многие логические операции (конъюнкция, дизъюнкция, отрицание) могут быть выполнены непосредственно над OBDD с помощью алгоритмов, выполняющих манипуляции над графами за полиномиальное время.

Замечание 3.25

Размер OBDD определяется как булевой функцией, так и выбором порядка входных переменных. При составлении графа для булевой функции $f(x_1, \dots, x_n)$ количество узлов в лучшем случае может линейно зависеть от n , а в худшем случае зависимость может быть экспоненциальной при неудачном выборе порядка входных переменных.

Выбор порядка переменных является критически важным при использовании таких структур данных на практике. Проблема нахождения лучшего порядка переменных является NP-полной задачей (см. определение 3.114).

Пример 3.72. Построим OBDD-представление булевой функции, заданное лексикографически упорядоченным вектором значений:

$$f(x_1, x_2, x_3) = \{0, 1, 1, 1, 0, 1, 0, 0\}.$$

На рисунке 3.18 представлена исходная бинарная диаграмма решений данной функции.

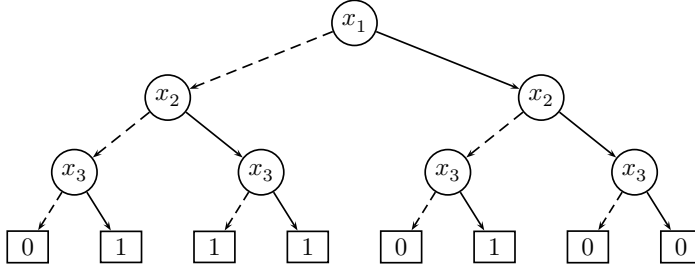


Рис. 3.18.

После применения первого правила получаем следующее дерево (рисунок 3.19).

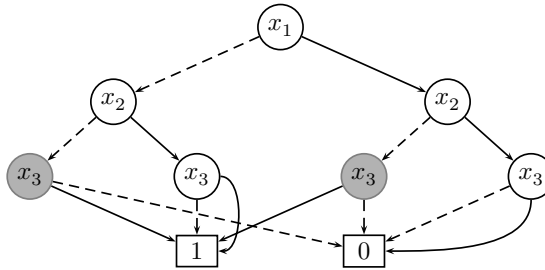


Рис. 3.19.

Применение второго правила показано на рисунке 3.20a. Две вершины с меткой x_3 , помеченные серым цветом на рисунке 3.19, сливаются в одну.

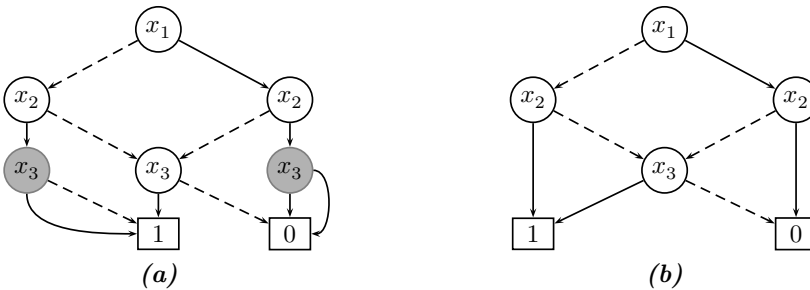


Рис. 3.20.

Применение третьего правила (рисунок 3.20b) удаляет две вершины с

переменной x_3 (помеченные серым цветом на рисунке 3.20a).

Пример 3.73. Построим OBDD-представление булевой функции, заданное лексикографически упорядоченным вектором значений:

$$f(x_1, x_2, x_3) = \{1, 0, 0, 1, 1, 0, 1, 0\}.$$

На рисунке 3.21 представлена исходная бинарная диаграмма решений данной функции.

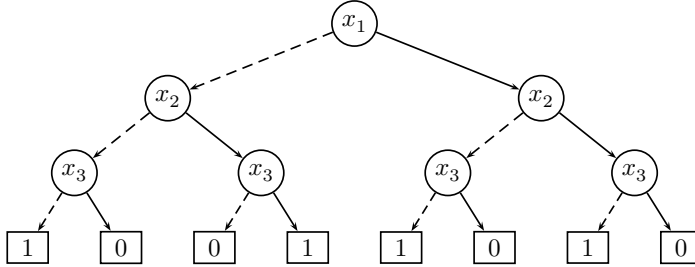


Рис. 3.21.

После применения первого правила получаем следующее дерево (рисунок 3.22).

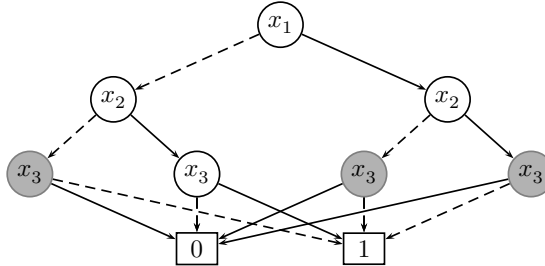


Рис. 3.22.

Применение второго правила показано на рисунке 3.23a. Три вершины с меткой x_3 , помеченные серым цветом на рисунке 3.22, сливаются в одну левую вершину с меткой x_3 на рисунке 3.23a.

Все эти вершины имеют на рисунке 3.22 в качестве левого потомка терминальную вершину 1, а в качестве правого — терминальную вершину 0.

Задача 3.19. Построить OBDD-представление булевой функции

$$f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3.$$

для следующего порядка переменных: $x_1 \prec x_2 \prec x_3$.

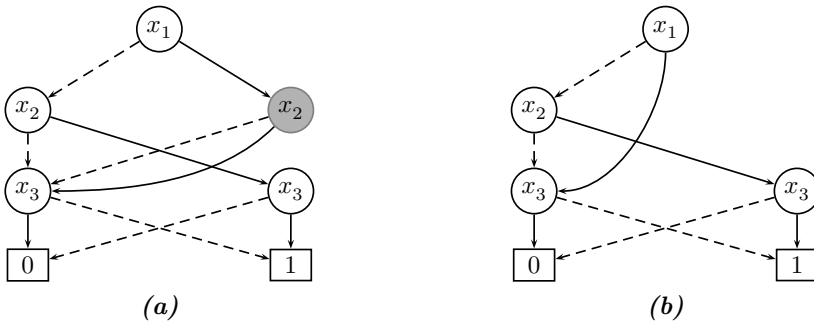


Рис. 3.23.

Задача 3.20. Построить OBDD-представления булевой функции

$$f(x_1, x_2, x_3, x_4) = (x_1 \leftrightarrow x_2)(x_3 \leftrightarrow x_4)$$

для следующих порядков переменных:

- 1) $x_1 \prec x_2 \prec x_3 \prec x_4$;
- 2) $x_1 \prec x_3 \prec x_2 \prec x_4$.

Сравнить получившиеся результаты.

Замечание 3.26

Учебной литературы на русском языке по этой теме почти нет. Хороший обзор и обсуждение можно найти в книге Дональда Кнута [23].

Задачи и вопросы

1. Построить булеву функцию от трех переменных, для которой

$$f(1, 0, 0) = 1, \quad f(0, 1, 1) = 0.$$

2. Найти все функционально полные системы, состоящие из одной булевой функции двух переменных.
3. Опишите классы Поста T_0, T_1, M, S, L .
4. Опишите булеву функцию $f(x, y, z)$ со следующими свойствами:
 - а) $f \in M, f(1, 1, 1) = 0$;
 - б) $f \in M, f(0, 0, 0) = 1$.

5. Про булеву функцию f известно, что она монотонная и непостоянная. Справедливо ли утверждение: f сохраняет ноль.

6. Сколько различных булевых функций от n переменных:

- а) самодвойственных;
- б) сохраняющих ноль;
- в) линейных?

7. Как построить разложение Шеннона для заданной булевой функции?

8. Покажите, что $\{\leftrightarrow, \oplus\}$ не составляют функционально полной системы функций. Предложите способ сделать эту систему полной добавлением одной, не более чем 2-местной функции.

9. Выяснить функциональную полноту следующей системы функций:

$$\{xy, x \oplus y, x \leftrightarrow (xy)\}.$$

10. Можно ли получить с помощью операций суперпозиции над множеством:

- а) $F = \{\rightarrow\}$ функцию $f(x, y) = x \oplus y$;
- б) $F = \{\wedge, \vee\}$ функцию $f(x, y) = x \rightarrow y$?

11. Может ли суперпозиция нелинейных функций дать функцию линейную?

12. Докажите полноту систем булевых функций:

- а) $\{x \rightarrow y, \overline{y \rightarrow x}\}$;
- б) $\{x \rightarrow y, 0\}$.

13. Выразите с помощью суперпозиции:

- а) \rightarrow через $1, \oplus$ и \wedge ;
- б) \wedge и \vee через \rightarrow, \neg ;
- в) \wedge, \vee и \neg через стрелку Пирса $f(x, y) = x \downarrow y = \overline{x \vee y}$.

14. Найдите все существенные переменные следующих булевых функций:

- а) $f(x, y, z) = xy \vee \overline{y}z$;
- б) $f(x, y, z) = (x \rightarrow (y \rightarrow z)) \rightarrow ((x \rightarrow y) \rightarrow (x \rightarrow z))$.

15. Докажите, что класс булевых функций, сохраняющих ноль, но не сохраняющих единицу не является замкнутым.

16. Докажите, что множество булевых функций, удовлетворяющих условию

$$f(x_1, \dots, x_n) \geq x_n,$$

образует замкнутый класс.

17. Докажите, что множество булевых функций вида $a_0(a_1 \vee x_1) \cdots (a_n \vee x_n)$ образует замкнутый класс.

18. Булеву функцию $f(x_1, x_2, x_3) = x_1 \bar{x}_2 \vee \bar{x}_1 x_3$ представьте в виде полинома Жегалкина.

19. Сформулируйте алгоритм, проверяющий булеву функцию на монотонность, используя метод перебора с возвратом.

3.3. Логика предикатов

Средствами логики нельзя выразить даже очень простые с математической точки зрения рассуждения.

Рассмотрим, например, следующее умозаключение. *Любой человек смертен, Сократ — человек, следовательно, Сократ смертен.* Все эти утверждения с точки зрения логики высказываний являются простыми, их структура не может быть проанализирована, поэтому нельзя доказать логичность этого рассуждения в рамках логики высказываний.

Рассмотрим расширение логики высказываний, которое называют *логикой предикатов*.

3.3.1. Основные определения

В естественной речи часто встречаются сложные высказывания, истинность которых может изменяться при изменении объектов, о которых идет речь, хотя форма самого высказывания остается прежней. Например, высказывание «У кошки четыре ноги» истинно, «У слона четыре ноги» тоже истинно, а высказывание «У человека четыре ноги» — ложно. Все эти высказывания имеют одну форму «У объекта x четыре ноги», где x — некоторый объект. Такое переменное высказывание, истинностное значение которого зависит от параметра, и называют предикатом.

Таким образом, предикат — функция, определенная на некотором множестве параметров и со значениями в $\{0, 1\}$. Этот подход естественным образом можно обобщить для случая n параметров.

Дадим формальное определение.

Определение 3.59. Пусть M — непустое множество. Тогда n -местным¹ предикатом² $P(x_1, \dots, x_n)$ называют функцию $P : M^n \rightarrow \{0, 1\}$.

Множество M называют *предметным множеством*, а аргументы $x_1, \dots, x_n \in M$ — *предметными переменными*.

Замечание 3.27

Можно считать, что высказывание — это нульместный предикат, т. е. предикат, в котором нет переменных для замены.


¹ Иногда местность предиката или функции будем обозначать верхним индексом P^n .

² От лат. *praedicatum* — сказанное.

Пример 3.74. Пусть множество M есть класс млекопитающих. Рассмотрим одноместный предикат $P(x) : U$ объекта x четыре ноги. Тогда

$$P(\text{слон}) = 1, P(\text{кошка}) = 1, P(\text{человек}) = 0.$$

Пример 3.75. Пусть M — множество натуральных чисел \mathbb{N} . Рассмотрим двухместные предикаты $Q(x, y) : D(x, y) = 1, G(x, y) : x < y$. Тогда, например, $Q(5, 8) = 1, Q(6, 8) = 0$, а $G(1, 3) = 1, G(8, 5) = 0$.

 **Замечание 3.28**

Булева функция от n переменных есть n -местный предикат, заданный на предметном множестве $M = B^n$.

Предикаты, заданные на множестве M , могут быть связаны логически связками — конъюнкцией, дизъюнкцией, отрицанием, импликацией и эквиваленцией. Эти операции вводятся очевидным образом согласно таблице 3.1. Приведем в качестве примера определение конъюнкции предикатов.

Определение 3.60. Предикат $R(x_1, \dots, x_n)$ называют конъюнкцией предикатов $P(x_1, \dots, x_n)$ и $Q(x_1, \dots, x_n)$, заданных на множестве M , если на любом наборе значений предметных переменных $(\alpha_1, \dots, \alpha_n)$ из M

$$R(\alpha_1, \dots, \alpha_n) = P(\alpha_1, \dots, \alpha_n) \wedge Q(\alpha_1, \dots, \alpha_n).$$

Для предикатов вводятся две новые операции — квантор¹ общности и квантор существования.

Определение 3.61. Пусть $P(x_1, \dots, x_n, y)$ — предикат, заданный на множестве M .

Выражение $(\forall y)P(x_1, \dots, x_n, y)$ (читается: для всякого y выполняется $P(x_1, \dots, x_n, y)$) есть предикат $Q(x_1, \dots, x_n)$, полученный из P навешиванием квантора общности на переменную y .

$$Q(\alpha_1, \dots, \alpha_n) = 1 \Leftrightarrow P(\alpha_1, \dots, \alpha_n, \beta) = 1 \text{ для любого } \beta \in M.$$

Выражение $(\exists y)P(x_1, \dots, x_n, y)$ (читается: существует y такой, что выполняется $P(x_1, \dots, x_n, y)$) есть предикат $R(x_1, \dots, x_n)$, полученный из P навешиванием квантора существования на переменную y .

$$R(\alpha_1, \dots, \alpha_n) = 1 \Leftrightarrow P(\alpha_1, \dots, \alpha_n, \beta) = 1 \text{ для некоторого } \beta \in M.$$

Операцию навешивания квантора \forall или \exists на переменную x называют еще квантификацией переменной x .

¹ От лат. *quantum* — сколько.

Пример 3.76. В обозначениях примера 3.75 рассмотрим предикаты:

$$R(x) = (\forall y)G(x, y), \quad W(y) = (\exists x)G(x, y).$$

Тогда $R(x) = 0$ для всех x , а $W(1) = 0$, $W(y) = 1$ для всех $y > 1$.

Введем понятие формулы логики предикатов. Сделаем это так же, как и в логике высказываний (см. определение 3.3), т. е. сначала введем понятие атомарной формулы, а затем — формулы.

В отличие от логики высказываний, где переменные (атомарные формулы) не анализируются, атомарная формула для логики предикатов имеет структуру.

Определение 3.62. *Сигнатурой* называют множество

$$\Sigma = \Sigma_{\mathcal{F}} \cup \Sigma_{\mathcal{P}} \cup \Sigma_{\mathcal{C}},$$

где $\Sigma_{\mathcal{F}} = \{f_i^{n_i}\}_{i \in I}$ — множество функциональных символов (функторов);

$\Sigma_{\mathcal{P}} = \{P_j^{n_j}\}_{j \in J}$ — множество предикатных символов;

$\Sigma_{\mathcal{C}} = \{c_k\}_{k \in K}$ — множество предметных констант (нульместных символов функций).

В определении сигнатуры можно оставить два множества $\Sigma_{\mathcal{F}}$ и $\Sigma_{\mathcal{P}}$, считая, что $\Sigma_{\mathcal{C}} \subseteq \Sigma_{\mathcal{F}}$.

Определение 3.63. *Термом* называют:

- 1) переменную или предметную константу;
- 2) выражение $f(t_1, \dots, t_n)$, где t_1, \dots, t_n — термы, а $f \in \Sigma_{\mathcal{F}}$.

Определение 3.64. *Атомарной формулой* называют выражение вида $P(t_1, \dots, t_n)$, где t_1, \dots, t_n — термы, а $P \in \Sigma_{\mathcal{P}}$.

Определение 3.65. *Формулой логики предикатов* называют:

- 1) атомарную формулу;
- 2) выражение вида

$$(F) \wedge (G), (F) \vee (G), \neg(F), (F) \rightarrow (G), (F) \leftrightarrow (G), (\forall y)F, (\exists y)F,$$

где F и G — формулы логики предикатов, переменная $y \in M$.

Формулу F в двух последних выражениях называют *областью действия квантора* по переменной y .

Замечание 3.29

К соглашениям о скобках и приоритете операций, принятом в логике высказываний (см. раздел 3.1.1), добавим следующее: кванторы имеют одинаковый приоритет, который больше приоритета всех связок.

Как и ранее, вместо $\neg(F)$ будем писать \overline{F} , а $(F) \wedge (G)$ иногда заменять на FG .

Определение 3.66. Вхождение переменной в формулу называют *связанным*, если переменная стоит непосредственно за квантором или входит в область действия квантора по этой переменной. В противном случае вхождение называют *свободным*.

Переменную называют *свободной переменной* формулы, если существует хотя бы одно свободное вхождение переменной в формулу, и *связной переменной* формулы, если имеется связанное вхождение.

Формулу называют *замкнутой формулой* или *предложением*, если любое вхождение переменной в формулу является связным.

Пример 3.77. Рассмотрим формулу

$$(\exists x)Q(x, y) \rightarrow \overline{P(g(x, y)) \wedge (\forall z)P(z)}.$$

Первое вхождение переменной x связано, второе — свободно. Все вхождения переменной y свободны, вхождение переменной z связано.

Формула $(\forall x)(P(x) \rightarrow Q(x) \wedge R(x))$ является предложением, так как все вхождения переменной x связаны.

3.3.2. Семантика логики предикатов

Формула логики предикатов есть только абстрактная схема высказывания. Введем понятие значения (смысла) формулы, т. е. семантику логики предикатов. В логике высказываний это было сделано с помощью интерпретации переменных логическими значениями 0, 1. Используем аналогичный подход для формул логики предикатов.

Определение 3.67. *Алгебраической системой* $\mathfrak{M} = \langle M, \Sigma^{\mathfrak{M}} \rangle$ сигнатуры Σ называют непустое предметное множество M , для которого задана *интерпретация сигнатурных символов* или просто *интерпретация*, т. е. функция, которая:

1) каждой предметной константе $c \in \Sigma_c$ ставит в соответствие выделенный элемент $c^{\mathfrak{M}} \in M$;

2) каждому символу n -местной функции $f^n \in \Sigma_{\mathcal{F}}$ ставит в соответствие n -местную функцию $f^{\mathfrak{M}}$, определенную на множестве M ;

3) каждому символу n -местного предиката $P^n \in \Sigma_{\mathcal{P}}$ ставит в соответствие n -местный предикат $P^{\mathfrak{M}}$, заданный на M .

Если сигнатура не содержит функциональных символов, то такую алгебраическую систему называют *моделью*.

В результате любая формула F получает в соответствие предикат, местность которого равна числу свободных переменных формулы F .

Замечание 3.30

Обычно, когда алгебраическая система известна и задана интерпретация, при обозначении предикатов, функций и выделенных элементов индекс \mathfrak{M} опускается.

Пример 3.78. Пусть $\mathfrak{M} = \langle \mathbb{N}, \Sigma^{\mathfrak{M}} = \{S(x, y, z), P(x, y, z)\} \rangle$, где

$$S(x, y, z) : x + y = z, \quad P(x, y, z) : xy = z.$$

1. Формула $F(x) = (\forall y)S(x, y, y)$ истинна в \mathfrak{M} только при $x = 0$.
2. Формула $Q(x) = (\forall y)P(x, y, y)$ истинна в \mathfrak{M} только при $x = 1$.
3. Формула $R(x, y) = (\forall z)(\forall u)(S(x, z, u) \rightarrow S(y, z, u))$ истинна в \mathfrak{M} только при $x = y$.
4. Формула $W(x, y) = (\exists z)S(x, z, y)$ истинна в \mathfrak{M} только при $x < y$.

Пример 3.79. Пусть $\mathfrak{M} = \langle 2^A, \Sigma^{\mathfrak{M}} = \{Q(x, y)\} \rangle$, где A — некоторое множество, $Q(x, y) : x \subseteq y$.

1. Формула

$$F(x, y, z) = Q(x, y) \wedge Q(x, z) \wedge (\forall u)(Q(u, y) \wedge Q(u, z) \rightarrow Q(u, x))$$

истинна в \mathfrak{M} только при $x = y \cap z$.

2. Формула $G(x) = (\forall y)Q(x, y)$ истинна в \mathfrak{M} только при $x = \emptyset$.

Задача 3.21. Пусть $\mathfrak{M} = \langle M, \Sigma^{\mathfrak{M}} = \{P(x), L(x), F(x), B(x, y)\} \rangle$, где M — множество точек, прямых и плоскостей трехмерного евклидова пространства, а $P(x)$, $L(x)$, $F(x)$ и $B(x, y)$ — следующие предикаты:

$$P(x) : x - \text{точка}; \quad L(x) : x - \text{прямая}; \quad F(x) : x - \text{плоскость}; \\ B(x, y) : x \text{ принадлежит } y.$$

Запишите следующие формулы:

- 1) *через две различные точки можно провести единственную прямую;*

2) через три точки, не лежащие на одной прямой, можно провести единственную плоскость.

Проиллюстрируем использование выразительных возможностей языка логики предикатов.

Пример 3.80. Рассмотрим задачу перевода на язык логики предикатов следующего рассуждения.

Любой радикал является сторонником общественного прогресса. Некоторые консерваторы недолюбливают всех сторонников общественного прогресса. Значит, некоторые консерваторы недолюбливают всех радикалов.

Выберем сигнатуру:

$R(x) : x$ — радикал; $P(x) : x$ — сторонник общественного прогресса;

$K(x) : x$ — консерватор; $L(x, y)$ — x недолюбливает y .

Тогда рассуждение запишется в виде следующей последовательности формул:

$$F_1 = (\forall x)(R(x) \rightarrow P(x)); F_2 = (\exists x)(K(x) \wedge (\forall y)(P(y) \rightarrow L(x, y)));$$

$$F_3 = (\exists x)(K(x) \wedge (\forall y)(R(y) \rightarrow L(x, y))).$$

Рассмотрение примера 3.80 показывает, что выразительных средств логики предикатов достаточно, чтобы записать рассуждение о радикалах и консерваторах. Естественно далее поставить вопрос, логично ли оно? Будет ли третье предложение следствием первых двух? Ответ на этот вопрос будет получен в примере 3.87 (см. раздел 3.3.5).

3.3.3. Примеры и задачи на выразительность логики предикатов

Рассмотрим еще несколько примеров формулировки утверждений с помощью предикатов.

Пример 3.81. На предметном множестве натуральных чисел задана сигнатура:

1) $E(x)$ — число x — четное;

2) $P(x)$ — число x — простое;

3) $S(x, y, z)$ — число x является суммой чисел y и z .

Запишем в виде формулы логики предикатов теорему Гольдбаха: *каждое четное натуральное число является суммой двух простых чисел.*

$$F = (\forall x) [E(x) \rightarrow (\exists y)(\exists z) (P(y) \wedge P(z) \rightarrow S(y, z, x))]$$

Пример 3.82. Пусть на предметном множестве $M = \mathbb{N}$ определены предикаты:

- 1) $P(x)$ — число x — простое;
- 2) $Q(x, y)$ — число x больше числа y .

Выразим с помощью формулы логики предикатов утверждение (Евклид): *простых чисел бесконечно много.*

$$G = (\forall x)(\exists y) [P(y) \wedge Q(y, x)].$$

Пример 3.83. Пусть на предметном множестве геометрических фигур заданы следующие предикаты:

- 1) $C(x)$ — x — квадрат;
- 2) $S(x)$ — x — шар;
- 3) $B(x)$ — x — фигура черного цвета;
- 4) $W(x)$ — x — фигура белого цвета.

Запишем в виде формулы логики предикатов утверждение: *если все шары черные, то белых квадратов нет.*

$$F = (\forall x) (S(x) \rightarrow B(x)) \rightarrow \overline{(\exists y) (C(y) \wedge W(y))}.$$

Задача 3.22. Пусть определена сигнатура:

- 1) $S(x, t)$ — я вижу предмет x в момент времени t ;
- 2) $T(x, t)$ — я беру предмет x в момент времени t .

Записать в виде формулы логики предикатов следующие утверждения:

1. Я беру всякую вещь, которую я никогда не вижу.
2. Я никогда не беру того, что я всегда вижу.
3. Всегда существуют вещи, которые я не вижу и не беру.

Задача 3.23. Подберите сигнатуру и запишите следующие рассуждения в виде последовательности формул логики предикатов.

1. *Имеются прилежные студенты. Ни один студент не лишен способностей. Значит, некоторые студенты, лишённые способностей, не прилежны.*

2. Тот, кто распускает этот слух, должен быть и ловким, и беспринципным. Браун не ловок. Джонсон не беспринципен. Значит, ни Браун, ни Джонсон не распускают этот слух.

3. Если бы кто-нибудь мог решить эту задачу, то и какой-нибудь математик мог бы. Смит — математик, а не может ее решить. Значит, задача неразрешима.

3.3.4. Равносильность формул логики предикатов

Определение 3.68. Формулы F и G называют *равносильными*, если для любой алгебраической системы $\mathfrak{M} = \langle M, \Sigma^{\mathfrak{M}} \rangle$ они одновременно истинны или одновременно ложны в \mathfrak{M} при любых значениях свободных переменных.

Как и в логике высказываний, равносильность формул будем символически обозначать $F = G$.

В отличие от логики высказываний, равносильность формул логики предикатов нельзя определить на основе построения таблиц истинности на всевозможных интерпретациях. Их может быть бесконечно много.

Первый способ определения равносильности основан на рассмотрении значений истинности обеих формул для произвольной алгебраической системы согласно определению 3.68.

Пример 3.84. Установим равносильность формул $F(x) = \overline{(\forall y)P(x, y)}$ и $G(x) = (\exists y)\overline{P(x, y)}$

Рассмотрим произвольную алгебраическую систему $\mathfrak{M} = \langle M, \Sigma^{\mathfrak{M}} \rangle$. Пусть $F^{\mathfrak{M}}(a) = 1$ для произвольного $a \in M$. Тогда найдется $b \in M$ такой, что $P^{\mathfrak{M}}(a, b) = 0$. Следовательно $\overline{P^{\mathfrak{M}}(a, b)} = 1$. Из последнего равенства получаем, что $G^{\mathfrak{M}}(a) = 1$.

Итак, доказано, что если $F^{\mathfrak{M}}(a)$ истинно, то $G^{\mathfrak{M}}(a)$ тоже истинно для произвольного $a \in M$. Обратная импликация доказывается аналогично. Следовательно, формулы $F(x)$ и $G(x)$ равносильны.

Определение 3.69. Формулу F называют *тождественно истинной* (*общезначимой* или *тавтологией*), если она истинна в любой алгебраической системе при любых значениях свободных переменных.

Прием, использованный в примере 3.84, позволяет устанавливать равносильность для достаточно простых формул, и его сложно применять в общем случае. Основным способом состоит в преобразовании формул на основе установленных равносильностей. Имеет место следующая теорема.

Теорема 3.33. Пусть F, G, H — произвольные формулы логики предикатов. Тогда законы 1–15 логики высказываний, установленные в теореме 3.4, справедливы и для формул логики предикатов.

Доказательство. Доказательство каждого закона проводится так же, как в примере 3.84 с использованием таблицы 3.1 истинности логических связок. ■

Дополним полученный список законами, специфичными для логики предикатов.

Теорема 3.34. Для произвольных формул F и G логики предикатов справедливы равносильности (законы):

$$16) (\forall x)(F(x) \wedge G(x)) = (\forall x)F(x) \wedge (\forall x)G(x);$$

$$17) (\exists x)(F(x) \vee G(x)) = (\exists x)F(x) \vee (\exists x)G(x);$$

$$18) (\forall x)(\forall y)F(x, y) = (\forall y)(\forall x)F(x, y);$$

$$19) (\exists x)(\exists y)F(x, y) = (\exists y)(\exists x)F(x, y);$$

$$20) \overline{(\forall x)F(x)} = (\exists x)\overline{F(x)};$$

$$21) \overline{(\exists x)F(x)} = (\forall x)\overline{F(x)}.$$

Доказательство. В примере 3.84 была установлена равносильность 20. Остальные доказываются аналогично. ■

Согласно законам 16 и 17 квантор общности можно переносить через конъюнкцию, а квантор существования — через дизъюнкцию. Однако поменять местами \wedge и \vee в этих законах нельзя.

Доказательство того, что формулы неравносильны, будем осуществлять построением алгебраической системы \mathfrak{M} , в которой одна формула истинна, а другая — ложна.

Следствие 3.8. Существует алгебраическая система \mathfrak{M} , в которой

$$(\forall x)(F(x) \vee G(x)) \neq (\forall x)F(x) \vee (\forall x)G(x);$$

$$(\exists x)(F(x) \wedge G(x)) \neq (\exists x)F(x) \wedge (\exists x)G(x).$$

Доказательство. Рассмотрим $\mathfrak{M} = \langle \mathbb{N}, \Sigma^{\mathfrak{M}} = \{F(x), G(x)\} \rangle$, где

$$F(x) : x - \text{четное число}; \quad G(x) : x - \text{нечетное число}.$$

Для данной алгебраической системы \mathfrak{M} получаем неравенства:

$$\begin{aligned} (\forall x)(F(x) \vee G(x)) = 1 &\neq (\forall x)F(x) \vee (\forall x)G(x) = 0, \\ (\exists x)(F(x) \wedge G(x)) = 0 &\neq (\exists x)F(x) \wedge (\exists x)G(x) = 1, \end{aligned}$$

которые и доказывают следствие. \blacksquare

Равносильности 18 и 19 позволяют менять местами одноименные кванторы. Однако для разноименных кванторов эта операция не сохраняет равносильность.

Следствие 3.9. Существует алгебраическая система \mathfrak{M} , в которой

$$(\forall x)(\exists y)(F(x, y)) \neq (\exists y)(\forall x)F(x, y). \quad (3.39)$$

Доказательство. Рассмотрим систему

$$\mathfrak{M} = \langle \mathbb{N}, \Sigma^{\mathfrak{M}} = \{F(x, y)\} \rangle, \text{ где } F(x, y) : x < y.$$

Тогда в (3.39) левая формула будет истинной, а правая — ложной. \blacksquare

В следствии 3.8 было доказано, что квантор существования нельзя переносить через конъюнкцию. Однако если одна из формул F или G не содержит переменной x , то легко показать, что это делать можно.

Следствие 3.10. Пусть формула G не содержит x . Тогда справедливы равносильности:

$$\begin{aligned} 22) \quad (\forall x)(F(x) \vee G) &= (\forall x)F(x) \vee G; \\ 23) \quad (\exists x)(F(x) \wedge G) &= (\exists x)F(x) \wedge G. \end{aligned}$$

Результаты теоремы 3.34 и следствия 3.10 обобщает следующая лемма.

Лемма 3.19. Равносильности 16, 17, и 22, 23 можно записать в общем виде:

$$\begin{aligned} 24) \quad (Q_1x)(Q_2y)(F(x) \vee G(y)) &= (Q_1x)F(x) \vee (Q_2y)G(y); \\ 25) \quad (Q_1x)(Q_2y)(F(x) \wedge G(y)) &= (Q_1x)F(x) \wedge (Q_2y)G(y), \end{aligned}$$

где Q_1, Q_2 — кванторы \exists или \forall ; переменная y не входит в $F(x)$; переменная x не входит в $G(y)$.

Рассмотрим еще два очевидных закона логики предикатов.

Лемма 3.20 (о переименовании связанных переменных). Пусть переменная z не входит в $F(x)$, а переменная x не входит в $F(z)$. Тогда имеют место следующие равносильности:

$$26) (\forall x)F(x) = (\forall z)F(z);$$

$$27) (\exists x)(F(x) = (\exists z)F(z).$$

Проиллюстрируем применение полученных равносильностей логики предикатов на примере.

Замечание 3.31

Как и ранее для логики высказываний и булевых функций, в цепочке равенств формул логики предикатов будем ставить номера использованных равносильностей из теоремы 3.34, ее следствий и леммы 3.20.

Пример 3.85. Докажем равносильность формул F и G , где:

$$F = \overline{(\exists x) ((\forall y)P(x, y) \rightarrow (\forall z)(P(z, z) \vee Q(z)))};$$

$$G = (\forall x)(\forall y)(\exists z) (P(x, y) \wedge \overline{P(z, z)} \wedge \overline{Q(z)}).$$

Запишем последовательность равносильных преобразований.

$$\begin{aligned} F &\stackrel{21}{=} (\forall x) \overline{((\forall y)P(x, y) \rightarrow (\forall z)(P(z, z) \vee Q(z)))} \stackrel{14}{=} \\ &\stackrel{14}{=} (\forall x) \overline{((\forall y)P(x, y) \vee (\forall z)(P(z, z) \vee Q(z)))} \stackrel{12,13}{=} \\ &\stackrel{12,13}{=} (\forall x) ((\forall y)P(x, y) \wedge \overline{(\forall z)(P(z, z) \vee Q(z))}) \stackrel{20,12}{=} \\ &\stackrel{20,12}{=} (\forall x) ((\forall y)P(x, y) \wedge (\exists z)(\overline{P(z, z)} \wedge \overline{Q(z)})) \stackrel{25}{=} \\ &\stackrel{25}{=} (\forall x)(\forall y)(\exists z) (P(x, y) \wedge \overline{P(z, z)} \wedge \overline{Q(z)}) = G. \end{aligned}$$

Задача 3.24. Докажите равносильности:

- 1) $(\forall x) ((\forall y)P(x, y) \rightarrow (\exists z)(P(x, z) \wedge Q(z))) = (\forall x)(\exists u) (P(x, u) \rightarrow Q(u));$
- 2) $(\exists x)F(x) \rightarrow (\exists x)G(x) = (\forall x)(\exists y)(F(x) \rightarrow G(y)).$

3.3.5. Логическое следствие

Определение 3.70. Формулу G называют *логическим следствием*¹ множества формул $K = \{F_1, \dots, F_k\}$, если для любой алгебраической системы \mathfrak{M} из истинности в \mathfrak{M} всех формул из K при некоторых значениях переменных следует истинность G в \mathfrak{M} при тех же значениях переменных (символически обозначают $K \models G$).

Пример 3.86. Покажем, что формула $G = Q(c)$ есть логическое следствие формул $F_1 = (\forall x)(P(x) \rightarrow Q(x) \wedge R(x))$ и $F_2 = P(c)$.

¹ Иногда говорят, что формула G *семантически следует* из множества формул K .

Рассмотрим произвольную алгебраическую систему $\mathfrak{M} = \langle M, \Sigma^{\mathfrak{M}} \rangle$. Пусть $F_1^{\mathfrak{M}} = F_2^{\mathfrak{M}} = 1$. Из истинности $F_2^{\mathfrak{M}}$ следует, что $P^{\mathfrak{M}}(c^{\mathfrak{M}}) = 1$. Истинность $F_1^{\mathfrak{M}}$ означает, что $P^{\mathfrak{M}}(\alpha) \rightarrow Q^{\mathfrak{M}}(\alpha) \wedge R^{\mathfrak{M}}(\alpha) = 1$ для любого элемента $\alpha \in M$, в частности и для $\alpha = c^{\mathfrak{M}}$.

Таким образом для $\alpha = c^{\mathfrak{M}}$ истинна импликация и ее посылка, следовательно, истинно и заключение, т. е. $Q^{\mathfrak{M}}(c^{\mathfrak{M}}) \wedge R^{\mathfrak{M}}(c^{\mathfrak{M}}) = 1$. Отсюда $G^{\mathfrak{M}} = Q^{\mathfrak{M}}(c^{\mathfrak{M}}) = 1$ и в силу произвольности алгебраической системы \mathfrak{M} формула G есть логическое следствие формул F_1 и F_2 .

Пример 3.87. Докажем логичность рассуждения о радикалах и консерваторах (см. пример 3.80). Это рассуждение записано в виде последовательности формул F_1, F_2 и F_3 .

Рассмотрим произвольную алгебраическую систему $\mathfrak{M} = \langle M, \Sigma^{\mathfrak{M}} \rangle$. Пусть $F_1^{\mathfrak{M}}$ и $F_2^{\mathfrak{M}}$ истинны. Покажем, что $F_3^{\mathfrak{M}}$ также истинна. Из истинности $F_2^{\mathfrak{M}}$ следует, что существует $\alpha \in M, K^{\mathfrak{M}}(\alpha) = 1$.

Возьмем произвольный элемент $\beta \in M$. Если $R^{\mathfrak{M}}(\beta) = 0$, то

$$K^{\mathfrak{M}}(\alpha) \wedge (R^{\mathfrak{M}}(\beta) \rightarrow L^{\mathfrak{M}}(\alpha, \beta)) = 1.$$

Пусть $R^{\mathfrak{M}}(\beta) = 1$. Тогда из истинности $F_1^{\mathfrak{M}}$ получаем $P^{\mathfrak{M}}(\beta) = 1$, а из истинности $F_2^{\mathfrak{M}}$ следует, что и $L^{\mathfrak{M}}(\alpha, \beta) = 1$.

В силу произвольности $\beta \in M$ формула $F_3^{\mathfrak{M}}$ истинна. Таким образом рассуждение о радикалах и консерваторах логично.

Определение 3.71. Множество формул логики предикатов

$$K = \{F_1(x_1, \dots, x_l), \dots, F_n(x_1, \dots, x_l)\}$$

выполнимо, если существует алгебраическая система $\mathfrak{M} = \langle M, \Sigma^{\mathfrak{M}} \rangle$ и набор значений свободных переменных $\alpha_1, \dots, \alpha_l \in M$ таких, что

$$F_1^{\mathfrak{M}}(\alpha_1, \dots, \alpha_l) = 1, \dots, F_n^{\mathfrak{M}}(\alpha_1, \dots, \alpha_l) = 1.$$

Пример 3.88. Множество формул

$$K = \{F_1 = (\forall x)(\exists y)(P(y) \wedge Q(x, y)), F_2 = (\forall y)Q(c, y), F_3 = \overline{P(c)}\}$$

выполнимо.

Пусть $\mathfrak{M} = \langle \mathbb{N}, \Sigma^{\mathfrak{M}} = \{P(x), Q(x, y), c\} \rangle$, где

$$P(x) : x - \text{простое число}, Q(x, y) : x \leq y, c = 1.$$

Тогда высказывание $F_1^{\mathfrak{M}}$ означает, что для любого натурального числа x найдется простое число y , которое не меньше x ; высказывание $F_2^{\mathfrak{M}}$

означает, что 1 — *наименьшее натуральное число*; высказывание $F_3^{\text{н}}$ означает, что 1 — *не простое число*. Ясно, что все эти высказывания истинны, поэтому множество формул K выполнимо.

Понятия логического следствия и выполнимости в логике первого порядка связаны точно так же, как и в логике высказываний.

Теорема 3.35. Формула G является логическим следствием формул F_1, \dots, F_k тогда и только тогда, когда множество формул $\{F_1, \dots, F_k, \bar{G}\}$ невыполнимо.

Доказательство. Для доказательства нужно провести рассуждения, аналогичные использованным при доказательстве теоремы 3.6. ■

3.3.6. Нормальные формы

Как и в логике высказываний, в логике предикатов рассматриваются формулы в определенной канонической форме. Приведем две из них: предваренную нормальную и сколемовскую нормальную формы.

Определение 3.72. Формула G имеет *предваренную* или *пренексную нормальную форму* (ПНФ), если

$$G = (Q_1x_1) \dots (Q_nx_n)F, \quad (3.40)$$

где Q_1, \dots, Q_n — кванторы \exists или \forall , а F — бескванторная формула.

Выражение $(Q_1x_1) \dots (Q_nx_n)$ называют *префиксом* формулы G , а F — *матрицей* формулы.

Если все Q_i равны \forall , то формулу называют \forall -формулой или *универсальной* формулой.

Пример 3.89. Формула $(\forall x)(\exists y)(P(x, y) \wedge \overline{Q(y)})$ имеет ПНФ, а формула $(\exists x)(T(x) \wedge (\forall y)(S(y) \rightarrow L(x, y)))$ — не имеет.

Формула $(\forall x)(\forall y)(\forall z)(F(x, x) \wedge F(x, z) \rightarrow (F(x, y) \vee F(y, z)))$ имеет универсальную форму.

Теорема 3.36. Для любой формулы F существует формула G , равносильная F и имеющая ПНФ.

Доказательство. Теорема следует из рассмотрения алгоритма 3.13, который по данной формуле F выдает формулу G в ПНФ. ■

Алгоритм 3.13. Приведение к предваренной нормальной форме

- 1: Исключаем из исходной формулы эквиваленции и импликации // Используем законы 14 и 15 теоремы 3.4

- 2: Переносим отрицание к атомарным формулам // Используем законы 11 и 12 теоремы 3.4 и 20 и 21 теоремы 3.34
- 3: Выносим кванторы вперед // Используем законы 24 и 25 теоремы 3.34 и при необходимости переименование связанных переменных (равносильности 26 и 27)

Пример 3.90. Пусть $F = (\forall x)P(x) \rightarrow (\forall y)(Q(y) \rightarrow (\forall z)R(y, z))$.

- 1 Выполним шаг 1.

$$F \stackrel{14}{=} \overline{(\forall x)P(x)} \vee (\forall y) (\overline{Q(y)} \vee (\forall z)R(y, z)).$$

- 2 Перейдем к шагу 2.

$$F \stackrel{20}{=} (\exists x)\overline{P(x)} \vee (\forall y) (\overline{Q(y)} \vee (\forall z)R(y, z)).$$

- 3 Выполнение шага 3 запишем цепочкой равносильностей.

$$\begin{aligned} F &\stackrel{24}{=} (\exists x)(\forall y) (\overline{P(x)} \vee \overline{Q(y)} \vee (\forall z)R(y, z)) \stackrel{22}{=} \\ &\stackrel{22}{=} (\exists x)(\forall y)(\forall z) (\overline{P(x)} \vee \overline{Q(y)} \vee R(y, z)). \end{aligned}$$

Задача 3.25. Приведите к ПНФ следующие формулы:

- 1) $\overline{(\exists x)P(x)} \vee (\forall x)Q(x, y)$;
- 2) $(\forall x) (R(x) \rightarrow \overline{(\exists y)S(y)})$.

Определение 3.73. Литералом в логике предикатов называют атомарную формулу или ее отрицание.

Замечание 3.32

С учетом определения 3.73, как и в логике высказываний, для бескванторной формулы логики предикатов можно ввести понятие конъюнктивной нормальной формы. При таком определении полностью сохраняется утверждение теоремы 3.11 и алгоритм приведения к КНФ, полученный при доказательстве теоремы 3.11.

Определение 3.74. Говорят, что \forall -формула имеет сколемовскую нормальную форму (СНФ), если ее матрица представлена в конъюнктивной нормальной форме.

Пример 3.91. Формула $(\forall x)(P(x) \wedge (\overline{R(y)} \vee Q(x, y)))$ имеет СНФ, а формулы $(\forall x)(\exists y)(Q(x, y) \vee G(y))$ и $(\forall x)(Q(x, y) \vee (P(y) \rightarrow G(x)))$ — нет.

Вторая формула не является \forall -формулой, а у третьей матрица не имеет КНФ.

Рассмотрим алгоритм приведения к СНФ. Такой процесс называют *сколемизацией*.

Историческая справка

Туральф Альберт Сколем (Скулем, 1887–1963) — норвежский математик, специалист в области математической логики.

Алгоритм 3.14. Приведение к сколемовской нормальной форме

С помощью алгоритма 3.13 приводим формулу к ПНФ

По теореме 3.11 матрицу формулы приводим к КНФ // формула будет иметь вид (3.40)

// Исключаем кванторы существования из префикса $(Q_1x_1) \dots (Q_nx_n)$

for each $Q_k \mid (Q_kx_k) = (\exists x_k)$ **do**

// Q_k — самый левый квантор существования в префиксе

if $k = 1$ **then** // квантор существования стоит первым в префиксе

$\alpha \leftarrow$ предметная константа, не входящая в матрицу формулы

Все вхождения переменной x_k в матрице заменяем на α

else

$f(x_1, \dots, x_{k-1}) \leftarrow (k-1)$ -местный функциональный символ, не входящий в матрицу формулы

Все вхождения переменной x_k в матрице заменяем на $f(x_1, \dots, x_{k-1})$

end if

(Q_kx_k) удаляем из префикса

end for

Замечание 3.33

При исключении кванторов в алгоритме 3.14 можно не выделять отдельно случай $k = 1$, так как при этом 0-местный функциональный символ есть предметная константа.

Пример 3.92. С помощью алгоритма 3.14 привести к СНФ формулу

$$F = (\exists x)(\forall y)(P(x, y) \rightarrow (\exists z)(Q(x, z) \wedge G(y))).$$

Приводим формулу к ПНФ:

$$\begin{aligned} F &\stackrel{14}{=} (\exists x)(\forall y) (\overline{P(x, y)} \vee (\exists z)(Q(x, z) \wedge G(y))) \stackrel{17}{=} \\ &\stackrel{17}{=} (\exists x)(\forall y)(\exists z) (\overline{P(x, y)} \vee (Q(x, z) \wedge G(y))). \end{aligned}$$

Преобразуем матрицу к КНФ:

$$F \stackrel{8}{=} (\exists x)(\forall y)(\exists z) \left((\overline{P(x, y)} \vee (Q(x, z) \wedge \overline{P(x, y)} \vee G(y))) \right).$$

Проводим процедуру сколемизации. Заменяв сначала x на α , а затем z на $f(y)$, получаем СНФ формулы F :

$$F \xrightarrow{x=\alpha} (\forall y)(\exists z) \left(\left(\overline{P(\alpha, y)} \vee Q(\alpha, z) \right) \wedge \left(\overline{P(\alpha, y)} \vee G(y) \right) \right) \xrightarrow{z=f(y)} \\ \xrightarrow{z=f(y)} (\forall y) \left(\left(\overline{P(a, y)} \vee Q(a, f(y)) \right) \wedge \left(\overline{P(a, y)} \vee G(y) \right) \right).$$

Справедлива следующая теорема.

Теорема 3.37. Для любой формулы F алгоритм 3.14 строит формулу G , имеющую СНФ и одновременно выполнимую (или невыполнимую) с F .

Доказательство. Из описания алгоритма ясно, что результатом его работы является формула в СНФ. Очевидно, преобразования, проводимые на первых двух шагах алгоритма, являются равносильными, в частности сохраняют выполнимость (или невыполнимость) исходной формулы.

Проведем анализ исключения кванторов существования. Пусть \widehat{F} — формула после приведения к КНФ. Покажем, что \widehat{F} невыполнима тогда и только тогда, когда невыполнима G .

Рассмотрим случай, когда \widehat{F} содержит только один квантор существования:

$$\widehat{F} = (\forall x_1) \dots (\forall x_{k-1})(\exists x_k)(\forall x_{k+1}) \dots (\forall x_n)Q(x_1, \dots, x_n), \\ G = (\forall x_1) \dots (\forall x_{k-1})(\forall x_{k+1}) \dots (\forall x_n)Q(x_1, \dots, x_{k-1}, y, x_{k+1}, \dots, x_n), \\ \text{где } y = f(x_1, \dots, x_{k-1}).$$

Пусть формула \widehat{F} невыполнима. Допустим, что формула G выполнима, т. е. существует интерпретация, в которой

$$(\forall x_1) \dots (\forall x_{k-1})(\forall x_{k+1}) \dots (\forall x_n)Q(x_1, \dots, x_{k-1}, \alpha, x_{k+1}, \dots, x_n) = 1, \\ \alpha = f(x_1, \dots, x_{k-1}).$$

Очевидно, что \widehat{F} также выполнима в данной интерпретации.

Наоборот, пусть формула G невыполнима. Допустим, что \widehat{F} выполнима. Введем функцию $f(x_1, \dots, x_{k-1}) = x_k$. Тогда очевидно, что $G = 1$, полученное противоречие доказывает теорему.

Если в префиксе имеется $m > 1$ кванторов существования, то доказательство проводится аналогично. ■

3.3.7. Примеры и задачи на приведение к ПНФ и СНФ

Пример 3.93. Приведем к сколемовской нормальной форме следующую формулу:

$$F = (\exists x)(\forall y) (P(x) \rightarrow Q(y, z)) \rightarrow (\exists x)(\forall z) (Q(x, z) \wedge P(y)).$$

Сначала приведем формулу F к предваренной нормальной форме. Запишем последовательность равносильных преобразований.

$$\begin{aligned} F &\stackrel{14}{=} \overline{(\exists x)(\forall y) (\overline{P(x)} \vee Q(y, z))} \vee (\exists x)(\forall z) (Q(x, z) \wedge P(y)) \stackrel{20,21}{=} \\ &\stackrel{20,21}{=} (\forall x)(\exists y) (\overline{P(x)} \vee Q(y, z)) \vee (\exists x)(\forall z) (Q(x, z) \wedge P(y)) \stackrel{12,13}{=} \\ &\stackrel{12,13}{=} (\forall x)(\exists y) (P(x) \wedge \overline{Q(y, z)}) \vee (\exists x)(\forall z) (Q(x, z) \wedge P(y)) \stackrel{27}{=} \\ &\stackrel{27}{=} (\forall x)(\exists y) (P(x) \wedge \overline{Q(y, z)}) \vee (\exists w)(\forall z) (Q(w, z) \wedge P(y)) \stackrel{24}{=} \\ &\stackrel{24}{=} (\forall x)(\exists y)(\exists w) (P(x) \wedge \overline{Q(y, z)} \vee (\forall z) (Q(w, z) \wedge P(y))) \stackrel{26}{=} \\ &\stackrel{26}{=} (\forall x)(\exists y)(\exists w) (P(x) \wedge \overline{Q(y, z)} \vee (\forall u) (Q(w, u) \wedge P(y))) \stackrel{24}{=} \\ &\stackrel{24}{=} (\forall x)(\exists y)(\exists w)(\forall u) (P(x) \wedge \overline{Q(y, z)} \vee Q(w, u) \wedge P(y)). \end{aligned}$$

Сколемизируем полученную формулу.

$$\begin{aligned} F &\xrightarrow{y=f(x)} (\forall x)(\exists w)(\forall u) (P(x) \wedge \overline{Q(f(x), z)} \vee Q(w, u) \wedge P(f(x))) \xrightarrow{w=g(x)} \\ &\xrightarrow{w=g(x)} (\forall x)(\forall u) \underbrace{(P(x) \wedge \overline{Q(f(x), z)} \vee Q(g(x), u) \wedge P(f(x)))}_{M(x, u, z)}. \end{aligned}$$

В последнем выражении легко привести матрицу формулы в КНФ:

$$\begin{aligned} M(x, u, z) &= (P(x) \vee Q(g(x), u)) \wedge (P(x) \vee P(f(x))) \wedge \\ &\wedge (\overline{Q(f(x), z)} \vee Q(g(x), u)) \wedge (\overline{Q(f(x), z)} \vee P(f(x))). \end{aligned}$$

Пример 3.94. Приведем к сколемовской нормальной форме формулу, уже представленную в ПНФ:

$$F = (\exists x)(\forall y)(\forall z)(\exists u)(\exists v)(\exists w) (P(x, y) \vee \overline{R(z, u, v)}Q(u, w)).$$

Проводим процедуру сколемизации.

$$F \xrightarrow{x=\alpha} (\forall y)(\forall z)(\exists u)(\exists v)(\exists w) (P(\alpha, y) \vee \overline{R(z, u, v)}Q(u, w)) \xrightarrow{u=f(y, z)}$$

$$\begin{aligned}
& \xrightarrow{u=f(y,z)} (\forall y)(\forall z)(\exists v)(\exists w) \left(P(\alpha, y) \vee \overline{R(z, f(y, z), v)} Q(f(y, z), w) \right) \xrightarrow{v=g(y,z)} \\
& \xrightarrow{v=g(y,z)} (\forall y)(\forall z)(\exists w) \left(P(\alpha, y) \vee \overline{R(z, f(y, z), g(y, z))} Q(f(y, z), w) \right) \xrightarrow{w=h(y,z)} \\
& \xrightarrow{w=h(y,z)} (\forall y)(\forall z) \underbrace{\left(P(\alpha, y) \vee \overline{R(z, f(y, z), g(y, z))} Q(f(y, z), h(y, z)) \right)}_{M(y,z)}.
\end{aligned}$$

Аналогично примеру 3.93 приводим матрицу формулы в КНФ:

$$M(y, z) = \left(P(\alpha, y) \vee \overline{R(z, f(y, z), g(y, z))} \right) \left(P(\alpha, y) \vee Q(f(y, z), h(y, z)) \right).$$

Задача 3.26. Привести формулу логики предикатов сначала к предваренной нормальной форме, затем к сколемовской нормальной форме.

$$F = (\forall x)(\forall y) [(\exists z) (P(x, z) \wedge P(y, z)) \rightarrow (\exists y)Q(x, y, u)].$$

Пример 3.95. Равносильными преобразованиями преобразуем формулу логики предикатов, представленную в ПНФ:

$$F = (\forall x)(\forall y)(\exists z)(\exists u) (P(z) \vee Q(x, y, u))$$

так, чтобы в ней осталось три квантора.

Имеем:

$$\begin{aligned}
F & \stackrel{17}{=} (\forall x)(\forall y) ((\exists z)P(z) \vee (\exists u)Q(x, y, u)) \stackrel{23}{=} \\
& \stackrel{23}{=} (\forall x)(\forall y) ((\exists t)P(t) \vee (\exists t)Q(x, y, t)) \stackrel{19}{=} (\forall x)(\forall y)(\exists t) (P(t) \vee Q(x, y, t)).
\end{aligned}$$

Пример 3.96. Используя равносильности, преобразуем формулу логики предикатов, представленную в ПНФ:

$$G = (\exists x)(\exists y)(\forall z)(\forall u) (S(x, z, u) \vee L(y))$$

так, чтобы в ней осталось три квантора.

Запишем цепочку равносильных преобразований:

$$\begin{aligned}
G & \stackrel{24}{=} (\exists x)(\forall z)(\forall u)S(x, z, u) \vee (\exists y)L(y) \stackrel{27}{=} (\exists t)(\forall z)(\forall u)S(t, z, u) \vee (\exists t)L(t) \stackrel{24}{=} \\
& \stackrel{24}{=} (\exists t)(\forall z)(\forall u) (S(t, z, u) \vee L(t)).
\end{aligned}$$

3.3.8. Метод резолюций в логике предикатов

Обобщим метод резолюций для решения задачи логического следования в логике предикатов. Как и в логике высказываний, нужно преобразовать

множество формул логики предикатов к множеству *бескванторных дизъюнктов* литералов, сохранив при этом преобразовании выполнимость (или невыполнимость).

Рассмотрим алгоритм такого преобразования для множества формул логики предикатов $T = \{F_1, \dots, F_k\}$. Множество дизъюнктов обозначим S_T .

Алгоритм 3.15. Инициализация метода резолюций

// Инициализация

$S_T \leftarrow \emptyset$

for each $F_i \in T$ **do**

Алгоритмом 3.14 (сохраняя выполнимость) приводим F_i к формуле G_i в СНФ

if x_{i_1}, \dots, x_{i_m} — свободные переменные G_i **then** // G_i — незамкнутая формула

for each x_{i_n} **do**

// α_{i_n} — предметная константа, не входящая в матрицу формулы G_i

Заменяем все свободные вхождения x_{i_n} на предметную константу α_{i_n}

end for

Получаем замкнутую формулу H_i

// формула H_i , одновременно выполнима (или невыполнима) с G_i

else

$H_i \leftarrow G_i$

// Формула G_i уже замкнута

end if

В формуле H_i опускаем кванторы всеобщности: $H_i \leftarrow D_{i_1} \wedge \dots \wedge D_{i_p}$

// подразумевая, что каждая переменная связана соответствующим квантором

$S_T \leftarrow S_T \cup \{D_{i_1}, \dots, D_{i_p}\}$

// Очевидно, что выполнимость H_i равносильна выполнимости множества дизъюнктов $\{D_{i_1}, \dots, D_{i_p}\}$

end for

3.3.8.1. Подстановка и унификация

Так как литералы в логике предикатов имеют структуру, правило резолюций в прежнем виде неприменимо. Действительно, множество дизъюнктов $S = \{P(x), \overline{P(\alpha)}\}$ невыполнимо (переменная x связана квантором общности). Но по правилу резолюций для логики высказываний нельзя построить опровержение S .

Сделаем в множестве S замену $x \rightarrow \alpha$. Получим множество дизъюнктов $\tilde{S} = \{P(\alpha), \overline{P(\alpha)}\}$. Очевидно, что множества S и \tilde{S} одновременно выполнимы (или невыполнимы). Но из \tilde{S} с помощью правила резолюций легко получить пустой дизъюнкт \square .

Таким образом, в логике предикатов правило резолюций надо дополнить возможностью делать подстановку.

Определение 3.75. *Подстановкой* называют множество равенств

$$\delta = \{x_1 = t_1, \dots, x_n = t_n\},$$

где x_1, \dots, x_n — переменные; t_1, \dots, t_n — термы, причем терм t_i не содержит переменной x_i .

Действие подстановки $\delta = (x_1 = t_1, \dots, x_n = t_n)$ на множество дизъюнктов S , выражающееся в одновременной замене $x_1 \rightarrow t_1, \dots, x_n \rightarrow t_n$, обозначим $\delta(S)$; *пустую подстановку*, не содержащую равенств, — ε ; композицию (последовательное применение) подстановок — δ_1 и затем δ_2 — $\delta_2 \circ \delta_1$.

Пример 3.97. Пусть

$$S = \{R(x, y) \vee P(f(y), z), Q(y, z, z)\}, \quad \delta = \{x = g(y), y = \alpha, z = h(x)\},$$

тогда $\delta(S) = \{R(g(y), \alpha) \vee P(f(\alpha), h(x)), Q(\alpha, h(x), h(x))\}$.

Определение 3.76. Пусть $\{L_1, \dots, L_k\}$ — множество литералов. Подстановку δ называют *унификатором* этого множества, если выполнено равенство $\delta(L_1) = \dots = \delta(L_k)$.

Множество называют *унифицируемым*, если существует унификатор этого множества.

Пример 3.98. Множество $S = \{P(a, y), P(x, f(b))\}$ унифицируемо подстановкой $\delta = \{x = a, y = f(b)\}$.

Если множество унифицируемо, то существует, как правило, не один унификатор этого множества, а несколько. Среди всех унификаторов данного множества выделяют *наиболее общий унификатор*.

Определение 3.77. Унификатор δ множества литералов называют наиболее общим унификатором этого множества, если для любого унификатора σ того же множества существует подстановка λ такая, что $\sigma = \lambda \circ \delta$.

Пример 3.99. Подстановка в примере 3.98 — наиболее общий унификатор.

Рассмотрим алгоритм нахождения наиболее общего унификатора. Введем понятие множества рассогласований.

Определение 3.78. Пусть S — множество литералов. Определим первую слева позицию, в которой не для всех литералов стоит один и тот же символ. Затем в каждом литерале выпишем выражение, которое начинается символом, занимающим эту позицию. (Этими выражениями могут быть сам литерал, атомарная формула или терм.) Множество полученных выражений называют *множеством рассогласований* в S .

Пример 3.100. Пусть $S = \{P(x, f(y)), P(x, \alpha), P(x, g(h(x)))\}$, тогда первая слева позиция, в которой не все литералы имеют один и тот же символ, — пятая. Множество рассогласований есть $\{f(y), \alpha, g(h(x))\}$.

Алгоритм 3.16. Алгоритм унификации множества литералов

// Инициализация

$k \leftarrow 0$

$S_k \leftarrow S$

$\delta_k \leftarrow \varepsilon$

// Основной цикл

while S_k содержит более одного литерала **do**

$E_k \leftarrow$ множество рассогласований в S_k

if существуют переменная $x_i \in E_k$ и терм $t_j \in E_k$, не содержащий x_i **then**

$\delta_{k+1} \leftarrow \{x_i = t_j\} \circ \delta_k$;

$S_{k+1} \leftarrow \delta_{k+1}(S_k)$ // в множестве S_k выполняем замену $x_i = t_j$

$k \leftarrow k + 1$

else

return множество S неунифицируемо

end if

end while

return δ_k — наиболее общий унификатор

Пример 3.101. Пусть $S = \{P(x, f(y), f(g(x))), P(\alpha, u, z)\}$. Протокол работы алгоритма 3.16 запишем в виде таблицы 3.43.

Таблица 3.43.

k	S_k	E_k	Подстановка
0	$\{P(x, f(y), f(g(x))), P(\alpha, u, z)\}$	$\{x, \alpha\}$	$x = \alpha$
1	$\{P(\alpha, f(y), f(g(\alpha))), P(\alpha, u, z)\}$	$\{u, f(y)\}$	$u = f(y)$
2	$\{P(\alpha, f(y), f(g(\alpha))), P(\alpha, f(y), z)\}$	$\{z, f(g(\alpha))\}$	$z = f(g(\alpha))$
3	$\{P(\alpha, f(y), f(g(\alpha)))\}$		

Множество S унифицируемо и наиболее общий унификатор:

$$\delta_3 = \{x = \alpha, u = f(y), z = f(g(\alpha))\}.$$

Пример 3.102. Пусть $S = \{Q(f(\alpha), g(x)), Q(y, y)\}$. Протокол работы алгоритма 3.16 запишем в виде таблицы 3.44.

Таблица 3.44.

k	S_k	E_k	Подстановка
0	$\{Q(f(\alpha), g(x)), Q(y, y)\}$	$\{y, f(\alpha)\}$	$y = f(\alpha)$
1	$\{Q(f(\alpha), g(x)), Q(f(\alpha), f(\alpha))\}$	$\{g(x), f(\alpha)\}$	

Множество S не унифицируемо, так как множество рассогласований

$$E_1 = \{g(x), f(\alpha)\}$$

не содержит переменной.

Замечание 3.34

Алгоритм 3.16 всегда заканчивает работу, так как S_{k+1} содержит на одну переменную меньше, чем S_k .

Обоснованием алгоритма 3.16 служит следующая теорема.

Теорема 3.38. Пусть S — множество литералов. Если S унифицируемо, то алгоритм 3.16 возвращает δ_k — наиболее общий унификатор множества S .

Доказательство. Рассмотрим σ — произвольный унификатор множества S . Покажем, что на каждом шаге k алгоритма 3.16 справедливо равенство

$$\sigma = \lambda_k \circ \delta_k, \quad (3.41)$$

где λ_k — некоторая подстановка. Доказательство проведем индукцией по k .

База индукции: $k = 0$. Тогда $\delta_0 = \varepsilon$ и $\sigma = \sigma \circ \varepsilon = \sigma \circ \delta_0$. В качестве подстановки λ_0 можно взять σ .

Шаг индукции: Предположим, что для $0 \leq k \leq t$ равенство (3.41) справедливо. Докажем его для $k = t + 1$.

Если S_m содержит один литерал, то на следующем проходе алгоритм закончит работу. Согласно (3.41) δ_m есть наиболее общий унификатор.

Пусть S_m содержит более одного литерала. Рассмотрим множество рассогласований E_m . Так как σ — унификатор множества S и по индукционному предположению равенство (3.41) справедливо для $k = t$, то подстановка λ_m должна унифицировать множество E_m .

Поскольку E_m — унифицируемое множество рассогласований, в E_m существует хотя бы одна переменная x . Пусть терм $t \in E_m$, $t \neq x$. Множество E_m унифицируется подстановкой λ_m , поэтому $\lambda_m(x) = \lambda_m(t)$. Отсюда следует, что t не содержит x .

Не умаляя общности, можно считать, что в алгоритме 3.16 для получения очередной подстановки δ_{m+1} использовано равенство $x = t$, т. е.

$$\delta_{m+1} = \{x = t\} \circ \delta_m. \quad (3.42)$$

Из равенства $\lambda_m(x) = \lambda_m(t)$ следует, что $\{x = \lambda_m(t)\} \in \lambda_m$. Тогда

$$\lambda_{m+1} = \lambda_m \setminus \{x = \lambda_m(t)\}.$$

Так как t не содержит x , то $\lambda_{m+1}(t) = \lambda_m(t)$. Таким образом,

$$\lambda_{m+1} \circ \{x = t\} = \lambda_{m+1} \cup \{x = \lambda_{m+1}(t)\} = \lambda_{m+1} \cup \{x = \lambda_m(t)\} = \lambda_m. \quad (3.43)$$

Подставив выражение для λ_m из (3.43) в (3.41) при $k = m$ с учетом (3.42), получаем

$$\sigma = \lambda_m \circ \delta_m = \lambda_{m+1} \circ \{x = t\} \circ \delta_m = \lambda_{m+1} \circ \delta_{m+1}.$$

Равенство (3.41) доказано для $k = m + 1$ и, следовательно, для любого k .

По условию теоремы множество S унифицируемо, следовательно, алгоритм должен закончить работу, возвратив последнюю подстановку δ_k . Очевидно, что это унификатор множества S , а в силу (3.41) — наиболее общий унификатор. ■

3.3.8.2. Метод резолюций

Определение 3.79. *Правилом резолюций в логике предикатов называют следующее условие: из дизъюнктов $\overline{Q(t_1, \dots, t_k)} \vee F$ и $Q(r_1, \dots, r_k) \vee G$ выводим дизъюнкт $\delta(F) \vee \delta(G)$, где δ — наиболее общий унификатор множества $\{Q(t_1, \dots, t_k), Q(r_1, \dots, r_k)\}$.*

Дизъюнкт $\delta(F) \vee \delta(G)$ называют *бинарной резольвентой*.

Пример 3.103. С помощью правила резолюций из пары дизъюнктов $P(x) \vee Q(x)$ и $\overline{P(\alpha)} \vee R(y)$ подстановкой $\{x = \alpha\}$ можно вывести дизъюнкт $Q(\alpha) \vee R(y)$.

В отличие от логики высказываний, в логике предикатов существует еще одно правило.

Определение 3.80. Пусть дан дизъюнкт D , в котором два и более литералов с одинаковым знаком (*наличие или отсутствие отрицания*) имеют унификатор δ . Тогда $\delta(D)$ называют *склеивкой* дизъюнкта D .

Пример 3.104. Склеивка дизъюнкта $\overline{P(x)} \vee \overline{P(f(y))} \vee Q(x, y)$ дает дизъюнкт $\overline{P(f(y))} \vee Q(f(y), y)$. При этом унификатор $\delta = \{x = f(y)\}$.

Модифицируем определение вывода в логике предикатов с учетом нового правила склейки.

Определение 3.81. Пусть S — множество дизъюнктов. *Выводом* из множества дизъюнктов S называется последовательность дизъюнктов $D_1 \dots, D_n$ такая, что каждый дизъюнкт D_i или принадлежит S , или выводим либо из предыдущих дизъюнктов по правилу резолюций, либо из предыдущего дизъюнкта по правилу склейки.

Как и в логике высказываний, дизъюнкт D выводим из S , если существует вывод из S , последним дизъюнктом которого является D .

Для логики предикатов также справедлива теорема о полноте. Ее формулировка совпадает с формулировкой теоремы 3.15.

Теорема 3.39. Множество дизъюнктов S логики предикатов невыполнимо тогда и только тогда, когда из S выводим пустой дизъюнкт.

Доказательство теоремы 3.39 можно найти в [65].

Для решения вопроса о логическом следствии формулы G из множества формул $T = \{F_1, \dots, F_k\}$ метод резолюций в логике предикатов применяется так же, как и в логике высказываний.

Алгоритм 3.17. Метод резолюций для логики предикатов

- 1: По алгоритму 3.15 из множества формул $T \cup \bar{G}$ получаем множество дизъюнктов S_T .
 - 2: Строим вывод пустого дизъюнкта из множества S_T .
-

Замечание 3.35

В отличие от логики высказываний, шаг 2 алгоритма 3.17 может никогда не завершиться при существовании бесконечного числа возможных резольвент, получаемых в процессе выполнения алгоритма. В 1936 г. А. Черчем и А. Тьюрингом было доказано, что не существует алгоритма для проверки невыполнимости множества формул логики предикатов.

Пример 3.105. Пусть

$$T = \{F_1 = (\forall x)(E(x) \rightarrow V(x) \wedge R(x)), F_2 = (\exists x)(E(x) \wedge Q(x))\},$$

$$G = (\exists x)(Q(x) \wedge R(x)).$$

Покажем, что $T \models G$. Используем алгоритм 3.15 для построения множества дизъюнктов. После сколемизации получаем:

$$\widehat{F}_1 = (\forall x) \left((\overline{E(x)} \vee V(x)) \wedge (\overline{E(x)} \vee R(x)) \right);$$

$$\widehat{F}_2 = E(\alpha) \wedge Q(\alpha), \quad \bar{G} = (\forall x) \left(\overline{Q(x)} \vee \overline{R(x)} \right).$$

Множество дизъюнктов S_T имеет вид

$$S_T = \{D_1 = \overline{E(x)} \vee V(x), D_2 = \overline{E(x)} \vee R(x), D_3 = E(\alpha), D_4 = Q(\alpha), \\ D_5 = \overline{Q(x)} \vee \overline{R(x)}\}.$$

Строим опровержение множества S_T .

$$D_4, D_5, D_6 \stackrel{D_4, D_5, \{x=\alpha\}}{=} \overline{R(\alpha)}, D_2, D_3, D_7 \stackrel{D_2, D_3, \{x=\alpha\}}{=} R(\alpha), D_8 \stackrel{D_6, D_7}{=} \square.$$

Пример 3.106. Установим логичность следующего рассуждения.

Таможенники обыскивают всякого, кто въезжает в страну, кроме высокопоставленных лиц. Некоторые люди, способствующие провозу наркотиков, въезжали в страну и были обысканы исключительно модями, также способствовавшими провозу наркотиков. Никто из высокопоставленных лиц не способствовал провозу наркотиков. Следовательно, некоторые из таможенников способствовали провозу наркотиков.

Представим рассуждение в виде последовательности формул логики предикатов. Введем следующую сигнатуру.

$$\begin{aligned} E(x) &— x \text{ въезжал в страну;} & S(x, y) &— x \text{ обыскивает } y; \\ V(x) &— x \text{ — высокопоставленное лицо,} & G(x) &— x \text{ — таможенник;} \\ P(x) &— x \text{ способствовал провозу наркотиков.} \end{aligned}$$

Тогда рассуждение может быть представлено следующими формулами логики предикатов:

$$\begin{aligned} F_1 &= (\forall x) (E(x) \wedge \overline{V(x)} \rightarrow (\exists y)(G(y) \wedge S(y, x))); \\ F_2 &= (\exists x)(E(x) \wedge P(x) \wedge (\forall y)(S(y, x) \rightarrow P(y))); \\ F_3 &= \overline{(\exists x)(V(x) \wedge P(x))}, G = (\exists x)(G(x) \wedge P(x)). \end{aligned}$$

С помощью алгоритма 3.15 строим множество дизъюнктов для формул $\{F_1, F_2, F_3, \overline{G}\}$. После сколемизации получаем:

$$\begin{aligned} \widehat{F}_1 &= (\forall x) ((\overline{E(x)} \vee V(x) \vee G(f(x))) \wedge (\overline{E(x)} \vee V(x) \vee S(f(x), x))); \\ \widehat{F}_2 &= (\forall y) (E(\alpha) \wedge P(\alpha) \wedge (\overline{S(y, \alpha)} \vee P(y))), \widehat{F}_3 = (\forall x) (\overline{V(x)} \vee \overline{P(x)}); \\ \widehat{\overline{G}} &= (\forall x) (\overline{G(x)} \vee \overline{P(x)}). \end{aligned}$$

Множество дизъюнктов S_T имеет вид:

$$S_T = \{D_1 = \overline{E(x)} \vee V(x) \vee G(f(x)), D_2 = \overline{E(x)} \vee V(x) \vee S(f(x), x), \\ D_3 = E(\alpha), D_4 = P(\alpha), D_5 = \overline{S(y, \alpha)} \vee P(y), D_6 = \overline{V(x)} \vee \overline{P(x)}, \\ D_7 = \overline{G(x)} \vee \overline{P(x)}\}.$$

Строим вывод пустого дизъюнкта в S_T .

$$D_2, D_3, D_8 \stackrel{D_2, D_3, \{x=\alpha\}}{=} V(\alpha) \vee S(f(\alpha), \alpha), D_1, D_9 \stackrel{D_1, D_3, \{x=\alpha\}}{=} V(\alpha) \vee G(f(\alpha)), \\ D_4, D_6, D_{10} \stackrel{D_4, D_6, \{x=\alpha\}}{=} \overline{V(\alpha)}, D_{11} \stackrel{D_9, D_{10}}{=} G(f(\alpha)), D_{12} \stackrel{D_8, D_{10}}{=} S(f(\alpha), \alpha), \\ D_5, D_{13} \stackrel{D_5, D_{12}, \{y=f(\alpha)\}}{=} P(f(\alpha)), D_7, D_{14} \stackrel{D_7, D_{13}, \{x=f(\alpha)\}}{=} \overline{G(f(\alpha))}, D_{15} \stackrel{D_{11}, D_{14}}{=} \square.$$

Задача 3.27. Доказать логичность следующих рассуждений используя метод резолюций.

1. Все студенты университета — болельщики «Зенита».
2. Некоторые студенты играют в футбол.
3. Следовательно, некоторые из болельщиков «Зенита» играют в футбол.

Введем следующие обозначения:

- а) $Z(x)$ — x — болельщик «Зенита»;
- б) $S(x)$ — x — студент университета;
- в) $F(x)$ — x играет в футбол.

Тогда вышеприведенные рассуждения можно представить формулами:

$$F_1 = (\forall x) (S(x) \rightarrow Z(x)), F_2 = (\exists y) (S(y) \wedge F(y)), \\ G = (\exists x) (Z(x) \wedge F(x)).$$

Составим множество формул $\{F_1, F_2, \overline{G}\}$ и каждую из них приведем к сколемовской нормальной форме. Получим формулы:

$$H_1 = (\forall x) (\overline{S(x)} \vee Z(x)), H_2 = (S(\alpha) \wedge F(\alpha)), \\ H_3 = (\forall x) (\overline{Z(x)} \vee \overline{F(x)}).$$

Множество дизъюнктов S_T имеет вид:

$$S_T = \{D_1 = \overline{S(x)} \vee Z(x), D_2 = S(\alpha), D_3 = F(\alpha), D_4 = \overline{Z(x)} \vee \overline{F(x)}\}.$$

Пустой дизъюнкт из множества S_T выводится очевидным образом.

Задача 3.28.

1. Существуют студенты, которые любят всех преподавателей.
2. Ни один из студентов не любит невежд.
3. Итак, ни один из преподавателей не является невеждой.

Введем сигнатуру:

- а) $S(x)$ — x студент;
- б) $T(x)$ — x преподаватель;
- в) $I(x)$ — x невежда;
- г) $L(x, y)$ — x любит y .

Запишем эти утверждения в виде формул логики предикатов:

$$\begin{aligned} F_1 &= (\exists x) (S(x) \wedge (\forall y) (T(y) \rightarrow L(x, y))), \\ F_2 &= (\forall x) (S(x) \rightarrow (\forall y) (I(y) \rightarrow \overline{L(x, y)})), \\ G &= (\forall x) (T(x) \rightarrow \overline{I(x)}). \end{aligned}$$

С помощью алгоритма 3.15 строим множество дизъюнктов для формул $\{F_1, F_2, G\}$. После сколемизации получаем:

$$\begin{aligned} \widehat{F}_1 &= (\forall y) (S(\alpha) \wedge (\overline{T(y)} \vee L(\alpha, y))), \\ \widehat{F}_2 &= (\forall x)(\forall y) (\overline{S(x)} \vee \overline{I(y)} \vee \overline{L(x, y)}), \\ \widehat{G} &= T(\alpha)I(\alpha). \end{aligned}$$

Множество дизъюнктов S_T имеет вид:

$$\begin{aligned} S_T &= \{D_1 = S(\alpha), D_2 = \overline{T(y)} \vee L(\alpha, y), D_3 = \overline{S(x)} \vee \overline{I(y)} \vee \overline{L(x, y)}, \\ &D_4 = T(\alpha), D_5 = I(\alpha)\}. \end{aligned}$$

Строим вывод пустого дизъюнкта в S_T .

$$\begin{aligned} &D_2, D_3, D_6 \stackrel{D_2, D_3, \{x=\alpha\}}{=} \overline{S(\alpha)} \vee \overline{T(y)} \vee \overline{I(y)}, \\ &D_1, D_7 \stackrel{D_1, D_6}{=} \overline{T(y)} \vee \overline{I(y)}, D_4, D_8 \stackrel{D_4, D_7, \{y=\alpha\}}{=} \overline{I(\alpha)}, \\ &D_5, D_9 \stackrel{D_5, D_8}{=} \square \end{aligned}$$

3.3.9. Метод резолюций и ПРОЛОГ

Историческая справка

В 1965 г. в работе «*A machine oriented logic based on the resolution principle*» Джон Алан Робинсон (J. A. Robinson) представил метод автоматического поиска доказательства теорем в логике предикатов, получивший название принципа резолюции. Идея данного метода была предложена Жаком Эрбраном в 1931 г., когда еще не было компьютеров.

Робинсон модифицировал этот метод так, что он стал пригоден для автоматического, компьютерного использования, и, кроме того, разработал эффективный алгоритм унификации, составляющий основу его метода.

Этот метод послужил отправной точкой для создания в 1973 г. группой искусственного интеллекта (Университет Марселя) во главе с Аленом Колмероз нового языка программирования со встроенной процедурой логического вывода — языка ПРОЛОГ (PROLOG). Название языка происходит от слов ЛОГическое ПРОграммирование (*PROgramming in LOGic*).

Определение 3.82. Дизъюнкт называют *хорновским*, если не более чем один литерал входит в него без отрицания. В языке ПРОЛОГ хорновские дизъюнкты называют обычно *предложениями* или *клизамми*.

Дизъюнкты, в которых ровно один литерал без отрицания, могут быть равносильным образом (использованные равносильности теоремы 3.4 указаны над равенством) преобразованы к виду

$$L \vee \bar{L}_1 \vee \dots \vee \bar{L}_n \stackrel{14,11}{=} L_1 \wedge \dots \wedge L_n \rightarrow L.$$

Такие дизъюнкты называют *правилами* и условно записывают так:

$$L_1, \dots, L_n \rightarrow L.$$

При этом дизъюнкт L называют *заголовком правила*.

Допустим случай, когда $n = 0$, тогда дизъюнкт имеет вид L . Такие дизъюнкты называют *фактами*.

Если ни один литерал не входит в дизъюнкт без отрицания, то его равносильным образом можно преобразовать к виду

$$\bar{L}_1 \vee \dots \vee \bar{L}_n \stackrel{11}{=} \overline{L_1 \wedge \dots \wedge L_n}.$$

Подобные дизъюнкты записывают так: $? - L_1, \dots, L_n$, и называют *запросами*.

Множество хорновских дизъюнктов, содержащих ровно один запрос, называют *ПРОЛОГ-программой*, а множество хорновских дизъюнктов без запросов — *базой знаний*.

Подобная постановка задачи, с одной стороны, позволяет существенно упростить процедуру вывода методом резолюций, а с другой — достаточна для реализации большинства алгоритмических процедур.

Замечание 3.36

ПРОЛОГ (наряду с SQL) представляет собой разновидность декларативного языка. В этом случае программа — описание предметной области как отношений между объектами и формулировка цели, которую требуется решить.

В отличие от, например, объектно-ориентированных языков, в ПРОЛОГ-программе нет явного описания последовательности действий, необходимых для решения задачи. ПРОЛОГ как система программирования состоит из языка хорновских дизъюнктов (в разных версиях синтаксис может быть различным) и интерпретатора, реализующего метод резолюций с той или иной стратегией выбора.

Наиболее часто используется линейная резолюция. Берется самый левый литерал цели (подцель) и первый унифицируемый с ним дизъюнкт. К ним применяется правило резолюции. Полученная резольвента добавляется в программу в качестве нового вопроса (основной дизъюнкт для линейной резолюции). И так до тех пор, пока не будет получен пустой дизъюнкт, что будет означать успех, или до тех пор, пока очередную подцель будет невозможно унифицировать ни с одним дизъюнктом программы, что будет означать неудачу.

В последнем случае включается так называемый *бектрекинг* — механизм возврата, который осуществляет откат программы к той точке, в которой выбирался унифицирующийся с последней подцелью дизъюнкт.

Пример 3.107. Рассмотрим следующую программу.

- 1: программирует(Иванов)
- 2: программирует(Петров)
- 3: читал(Иванов, «Алгоритмы: построение и анализ» [26])
- 4: читал(Петров, «Конек-Горбунок»)
- 5: книга(«Алгоритмы: построение и анализ» [26], программирование)
- 6: книга(«Конек-Горбунок», детская)
- 7: сдавал(Иванов)
- 8: сдавал(Петров)
- 9: сдавал(X), программирует(X), знает(X) \rightarrow экзамен(X , 5)
- 10: сдавал(X), программирует(X) \rightarrow экзамен(X , 4)
- 11: сдавал(X) \rightarrow экзамен(X , 3)
- 12: читал(X , Y), книга(Y , программирование) \rightarrow знает(X)
- 13: ? — экзамен(Петров, Z)

1 При вычислении ответа на запрос интерпретатор пытается унифицировать его с фактами. В нашем случае запрос не унифицируется ни с одним из фактов.

2 Тогда интерпретатор пытается унифицировать запрос с заголовком одного из правил. Это можно сделать с заголовком правила 9 с помощью подстановки $\{X = \text{Петров}, Z = 5\}$. Запрос принимает следующий вид:

14: ? – сдавал(Петров), программирует(Петров), знает(Петров)

3 Далее, первый литерал запроса 14 унифицируется с фактом 8, а второй с фактом 2. Последний дизъюнкт запроса 14 не унифицируется ни с одним из фактов. Унифицируем его с заголовком правила 12 следующей подстановкой $\{X = \text{Петров}\}$. Получаем запрос:

15: ? — читал(Петров, Y), книга(Y, программирование)

4 Первый литерал запроса 15 унифицируется с фактом 4 подстановкой $\{Y = \text{«Конек-Горбунок»}\}$, но получившийся второй литерал не унифицируется ни с одним из фактов, ни с заголовками правил. Возвращаемся назад к запросу 14. Ни один из его дизъюнктов не имеет альтернативных вариантов унификации, поэтому возвращаемся еще на шаг назад к запросу 13. Правило 9 «отработано».

5 Интерпретатор пытается унифицировать запрос с заголовком одного из правил, следующих за правилом 9. Это можно сделать с заголовком правила 10 с помощью подстановки $\{X = \text{Петров}, Z = 4\}$. Запрос принимает вид:

16: ? — сдавал(Петров), программирует(Петров)

6 Как было показано, первый литерал запроса 16 унифицируется с фактом 8, а второй — с фактом 2. Запрос пуст. Интерпретатор заканчивает работу. Ответ на запрос 13 положителен.

Интерес обычно представляет не столько сам факт успешного завершения программы, сколько конкретные значения переменных, при которых это возможно. В нашем случае $Z = 4$.

Пример 3.108. Рассмотрим следующие утверждения¹.

Ивана интересуют компьютеры, книги и автомобили. Петра интересует нечто, что интересует Ивана, но если это нечто является техникой и если это произведено в России. Известно, что компьютеры и автомобили — это техника. Кроме того, известно, что компьютеры производятся в Китае, а автомобили — в Америке и России. Вопрос: *Что интересует Петра?*

¹ Пример взят с сайта Олега Акимова (<http://sceptic-ratio.narod.ru/index.htm>).

Формализуем задачу на Прологе.

- 1: интерес(Иван, компьютеры)
- 2: интерес(Иван, книги)
- 3: интерес(Иван, автомобили)
- 4: интерес(Иван, X), техника(X), произведено(X , Россия) \rightarrow интерес(Петр, X)
- 5: техника(компьютеры)
- 6: техника(автомобили)
- 7: произведено(компьютеры, Китай)
- 8: произведено(автомобили, Америка)
- 9: произведено(автомобили, Россия)
- 10: ? — интерес(Петр, X)

Первые три факта несопоставимы с целью; далее унифицируем запрос с заголовком правила. Запрос принимает следующий вид:

- 11: ? — интерес(Иван, X), техника(X), произведено(X , Россия)

Первый литерал запроса 11 унифицируется с фактом 1, а второй — с фактом 5 подстановкой $\{X = \text{компьютеры}\}$, однако третий литерал при данной подстановке не унифицируется ни с одним из фактов, ни с заголовками правил.

Пробуем унифицировать первый литерал запроса с фактом 2 подстановкой $\{X = \text{книги}\}$, но с этой подстановкой не унифицируется второй литерал. Переходим к факту 3. С подстановкой $\{X = \text{автомобили}\}$ происходит унификация всех литералов запроса. Запрос пуст и программа заканчивает работу. Ответ на запрос 10 положителен, *Петра интересуют автомобили.*

Задачи и вопросы

1. Запишите утверждение, состоящее в том, что d есть НОД чисел a и b в терминах предиката $D(x, y)$, означающего « x делит y », и предиката $G(x, y)$, означающего « $x \geq y$ ».

2. Что такое *терм* в исчислении предикатов?

3. Что такое область действия квантора?

4. Объясните понятие связанного и свободного вхождения переменной в формулу.

5. Сколько свободных вхождений переменной y в формулу логики предикатов

$$F = T(x) \wedge (\forall x) [S(x, y) \rightarrow (\exists x) (R(x, y) \vee T(y))]?$$

6. Можно ли в формулах логики предикатов переставлять рядом стоящие *одноименные* кванторы?

7. Можно ли в формулах логики предикатов переставлять рядом стоящие *разноименные* кванторы?

8. Что такое *выполнимая* формула в исчислении предикатов?

9. Определите, выполнимы ли следующие формулы логики предикатов:

а) $(\exists x)(\forall y) (G(x, y) \wedge \overline{H(x, y)})$;

б) $(\exists x)(F(x) \rightarrow F(y))$;

в) $(\exists x)(\exists y) (P(x) \wedge \overline{P(y)})$;

г) $(\exists x)(\forall y) (Q(x, y) \rightarrow (\forall z)R(x, y, z))$.

10. Определите, общезначимы ли следующие формулы логики предикатов:

а) $Q(x) \rightarrow (\forall y)Q(y)$;

б) $(\forall x)Q(x) \rightarrow Q(y)$;

в) $\overline{(\exists x)P(x)} \rightarrow \overline{(\forall x)P(x)}$;

г) $(\exists x)P(x) \rightarrow (\forall x)P(x)$.

11. Что такое *интерпретация* в логике предикатов?

12. Приведите пример формулы логики предикатов в *предваренной нормальной* форме.

13. Приведите к предваренной нормальной форме формулы логики предикатов:

$$(\exists x)(\forall y)S(x, y) \rightarrow (\exists x)(\forall y)S(x, y).$$

14. Приведите пример формулы логики предикатов в *сколемовской нормальной* форме.

15. Приведите к сколемовской нормальной форме формулу логики предикатов:

а) $(\exists x)(\forall y)R(x, y) \rightarrow (\forall x)(\exists y)R(x, y)$;

б) $(\forall x)P(x) \rightarrow (\forall x)(\exists y)R(x, y)$.

16. Какое свойство формулы логики предикатов сохраняется при ее сколемизации?

17. Запишите бинарную резольвенту дизъюнктов:

$$D_1 = \neg Q(\alpha, f(x)) \vee R(x), D_2 = Q(u, z) \vee P(z).$$

18. Примените правило склейки к дизъюнкту

$$Q(x, y, \gamma) \vee Q(\alpha, y, z) \vee Q(x, \beta, z) \vee \neg R(x, y, z).$$

3.4. Нечеткая логика

Чем сложнее система, тем менее мы способны дать точные и в то же время имеющие практическое значение суждения об ее поведении.

Лотфи Заде

3.4.1. Нечеткие множества

3.4.1.1. Введение в нечеткие множества

Достаточно часто поиск оптимального решения практической задачи на основе классических методов математики затруднен. Причина заключается в проблеме осуществления корректного подбора приемлемого аналитического описания решаемой задачи. Но даже в случае успешной реализации аналитического описания поставленной задачи ее решение может потребовать слишком больших ресурсов.

Как заметил фон Нейман, стремление получить точную, исчерпывающую модель для достаточно сложного объекта (процесса) не имеет смысла, поскольку сложность такого описания становится соизмеримой со сложностью самого объекта. Однако существует другой подход к решению проблемы.

Историческая справка

В 1965 г. американский математик Лотфи Заде^а (L. Zade) опубликовал статью «Нечеткие множества» (Fuzzy sets). Было дано новое определение понятия множества, предназначенное для описания сложных плохо определенных систем, в которых наряду с количественными данными присутствуют неоднозначные, субъективные, качественные данные. Эта статья породила новое научное направление, появились нечеткие отношения, нечеткая логика и т. д. Эти понятия в настоящее время широко используются в экспертных системах, системах искусственного интеллекта. Нечеткая логика входит в качестве составной части в понятие «мягкие вычисления» (soft computing), введенное Заде в 1994 г.

Началом практического применения теории нечетких множеств можно считать 1975 г., когда Мамдани и Ассилиан построили первый нечеткий контроллер для управления простым паровым двигателем. Период с конца 1980-х гг. и до нашего времени характеризуется бумом практического применения теории нечеткой логики в разных сферах науки и техники. В начале 80-х европейские и американские инженеры и научные сообщества весьма скептически восприняли новую теорию^б. Зато на Востоке нечеткая логика пошла на ура. Для людей, воспитанных на восточной философии, с ее неоднозначными и расплывчатыми категориями, нечеткая

логика сразу стала своей, родной.

В бизнесе и финансах нечеткая логика получила признание после того, как в 1988 г. экспертная система на основе нечетких правил для прогнозирования финансовых индикаторов **единственная** предсказала биржевой крах. В 1990 г. Комитет по контролю экспорта США внес нечеткую логику в список критически важных оборонных технологий, не подлежащих экспорту потенциальному противнику.

За прошедшее время нечеткая логика прошла путь от почти антинаучной теории, практически отвергнутой в Европе и США, до ситуации конца 1990-х гг., когда в Японии в широком ассортименте появились нечеткие бритвы, пылесосы, фотокамеры.

Сегодня нечеткая логика рассматривается как стандартный метод моделирования и проектирования. В 1997 г. язык нечеткого управления (Fuzzy Control Language) внесен в Международный стандарт программируемых контроллеров IEC 1131-7. Системы на нечетких множествах разработаны и успешно внедрены в таких областях как: медицинская диагностика, техническая диагностика, финансовый менеджмент, управление персоналом, биржевое прогнозирование, распознавание образов, разведка ископаемых, выявление мошенничества, управление компьютерными сетями, управление технологическими процессами, управление транспортом, логистика, поиск информации в Интернете, радиосвязь и телевидение. Спектр приложений очень широкий — от бытовых видеокамер, пылесосов и стиральных машин до средств наведения ракет ПВО и управления боевыми вертолетами и самолетами.

В настоящее время в Японии это направление переживает настоящий бум. Здесь функционирует специально созданная лаборатория Laboratory for International Fuzzy Engineering Research (LIFE). Программой этой организации является создание более близких человеку вычислительных устройств. LIFE объединяет 48 компаний, в числе которых: Hitachi, Mitsubishi, NEC, Sharp, Sony, Honda, Mazda, Toyota. Из зарубежных участников LIFE можно выделить: IBM, Fuji Xerox, а также NASA^c.

^a Профессор технических наук Калифорнийского университета в Беркли.

^b В конце 1960-х гг. работы Л. Заде даже были рассмотрены в Конгрессе США как яркий пример бессмысленной траты государственных средств, выделяемых на развитие науки.

^c NASA стало использовать нечеткую логику в маневрах стыковки.

Как известно, одним из наиболее поразительных свойств человеческого интеллекта является способность принимать правильные решения в обстановке неполной и нечеткой информации. Построение моделей приближенных рассуждений человека и использование их в компьютерных системах будущих поколений представляет сегодня одну из важнейших проблем науки.

Нечеткая логика как раз и предназначена для формализации человеческих способностей к неточным или приближенным рассуждениям, которые позволяют более адекватно описывать ситуации с неопределенностью. Классическая логика по своей сути игнорирует проблему неопределенности, поскольку все высказывания и рассуждения в формальных логических системах могут иметь только два значения: 0 или 1.

В отличие от этого, в нечеткой логике истинность рассуждений оценивается в некоторой степени, которая может принимать и другие значения.

Чтобы иметь возможность выражать неопределенные знания, необходима такая логическая система, которая позволяет некоторому высказыванию иметь истинностное значение, отличающееся от бинарного 0 или 1. Один из подходов — расширить множество истинностных значений и позволить высказываниям принимать некоторые дополнительные значения.

Одним из первых такой вариант многозначной логической системы предложил в 1930 г. польский математик Ян Лукасевич.

В логике Лукасевича используется три истинностных значения: $\{0, 0.5, 1\}$, где значение 0 интерпретируется как *ложь*, 1 — как *истина*, а число 0.5 — как *возможно*. В качестве высказываний с истинностным значением «возможно» могут выступать такие, которые относятся к некоторому моменту времени в будущем. Так, например, высказывание *сборная России по футболу выйдет в 1/8 финала на предстоящем чемпионате мира* до начала чемпионата не может быть оценено ни как истинное, ни как ложное. Именно по этой причине более адекватным ответом на вопрос об его истинности будет использование трехзначной логики с соответствующей интерпретацией истинности в форме значения «возможно».

Благодаря такому подходу нечеткая логика обеспечивает разрешимость некоторых классически неразрешимых проблем. К примеру, хорошо известными являются древние парадоксы *sorites* (*куча*) и *falakros* (*лысый человек*).

- **Парадокс кучи:** одно пшеничное зерно не образует кучи. То же верно и для двух зерен, трех и т. д. Следовательно, куча не существует.

- **Парадокс лысого человека:** человек без волос или только с одним волосом — лысый. То же верно и для человека с двумя волосинами и т. д. Следовательно, все люди — лысые.

Указанные парадоксы возникают тогда, когда свойство «быть кучей» и «быть лысым» понимаются точно, т. е. исключая их нечеткость. Классическая двузначная логика не способна с ними справиться. В рамках нечеткой логики подобные парадоксы не имеют места.

Предлагаемое нечеткой логикой решение заключается в допущении, что импликация $S(x) \rightarrow S(x + 1)$, где $S(x)$ означает, например, высказывание « x зерен не образуют кучи», истинна только в некоторой степени, близкой к 1 в зависимости от числа зерен x , скажем, $1 - \Delta(x)$, где $\Delta(x) > 0$. При этом допущении парадокс кучи, так же как и парадокс лысого, исчезает.

Историческая справка

Ян Лукасевич — польский логик. Построил первую систему многозначной логики, а с ее помощью — систему модальной логики. Разработал оригинальный язык для формализации логических выражений — польская запись, послуживший основой для

более известной обратной польской записи, см. раздел «Префиксная и постфиксная записи формул».

Перейдем к рассмотрению основ нечеткой логики, которая использует основные понятия теории нечетких множеств для формализации неточных знаний и выполнения приближенных рассуждений в той или иной проблемной области.

3.4.1.2. Основные понятия

Пусть M — некоторое универсальное множество¹ и A его произвольное подмножество. Для любого $x \in M$ верно $x \in A$ или $x \notin A$. На множестве M вводится *характеристическая функция* подмножества A :

$$\mu_A(x) = \begin{cases} 1, & x \in A, \\ 0, & x \notin A. \end{cases}$$

Множества, для которых характеристическая функция принимает только значения 0 или 1, называются *четкими*.

Пример 3.109. Пусть $M = \{a, b, c, d, e, f, g, h\}$ и $A = \{a, b, c, d\}$. Тогда функция $\mu_A(x)$ описана таблицей 3.45.

Таблица 3.45.

x	a	b	c	d	e	f	g	h
$\mu_A(x)$	1	1	1	1	0	0	0	0

Очевидно, что для универсума $\mu_M(x) \equiv 1$, а для пустого множества $\mu_\emptyset(x) \equiv 0$. Для нечетких множеств характеристическая функция может принимать любые значения из промежутка $[0, 1]$.

Пример 3.110. В условиях примера 3.109 определим $\mu_A(x)$ по таблице 3.46.

Таблица 3.46.

x	a	b	c	d	e	f	g	h
$\mu_A(x)$	1.0	0.9	0.7	0.8	0.1	0.2	0.0	0.0

¹ В дальнейшем будем называть его «универсумом».

Для нечетких множеств характеристическую функцию называют *функцией принадлежности* (*Membership Function*).

Определение 3.83. *Нечетким множеством* A , заданным на универсуме M , будем называть упорядоченное множество пар

$$A = \{(x, \mu_A(x)) | \forall x \in M\}.$$

Используется также запись $x \in_{\mu_A(x)} A$. Например, $a \in_{1.0} A$, $b \in_{0.9} A$. Нечеткое множество A называется *пустым*, если

$$\mu_A(x) = 0, \forall x \in M.$$

Носителем нечеткого множества A , называется множество точек

$$S(A) = \{x \in M | \mu_A(x) > 0\}.$$

Высотой нечеткого множества A называется величина

$$h(A) = \sup_{x \in A} \mu_A(x).$$

Нечеткое множество называется *нормальным*, если его высота равна единице. В противном случае нечеткое множество называется *субнормальным*. Субнормальное нечеткое множество всегда можно нормализовать, поделив функцию принадлежности μ_A на величину $h(A)$.

Ядром нечеткого множества A называется его четкое подмножество, элементы которого имеют степени принадлежности, равные единице.

$$\text{core}(A) = \{x \in A | \mu_A(x) = 1\}.$$

Ядро субнормального нечеткого множества пустое.

Элементы, для которых $\mu_A(x) = 0.5$, называются *точками перехода* нечеткого множества A .

Четкое множество A^* , ближайшее к нечеткому множеству A , определяется следующим образом:

$$\mu_{A^*}(m) = \begin{cases} 0, & \text{если } \mu_A(m) < 0,5, \\ 1, & \text{если } \mu_A(m) > 0,5, \\ 0 \text{ или } 1, & \text{в противном случае.} \end{cases}$$

Кардинальным числом (*мощностью*) нечеткого n -элементного множе-

ства A называется величина

$$\text{card}(A) = |A| = \sum_{i=1}^n \mu_A(x_i). \quad (3.44)$$

Пример 3.111. Формализуем неточное определение «горячий чай». В качестве универсума M будет выступать шкала температуры в градусах Цельсия. Очевидно, что она будет изменяться от 0 до 100 градусов. Описание нечеткого множества для понятия «горячий чай» может выглядеть следующим образом (таблица 3.47).

Таблица 3.47.

x	0	10	20	30	40	50	60	70	80	90	100
$\mu_A(x)$	0.0	0.0	0.0	0.15	0.30	0.50	0.80	0.90	1.0	1.0	1.0

Так, чай с температурой 60° принадлежит к множеству «горячий чай» со степенью принадлежности 0.80. Для одного человека чай при температуре 60° может оказаться горячим, для другого — не слишком горячим. Именно в этом и проявляется нечеткость задания соответствующего множества.

Нечеткое множество в примере является нормальным, его точка перехода — 50° .

Определение 3.84. Множество A_α будем называть *нечетким множеством α -уровня*, если оно образует совокупность таких элементов $x \in M$, степень принадлежности которых множеству A больше или равна $\alpha \in (0, 1]$.

$$A_\alpha = \{x \mid \mu_A(x) \geq \alpha, x \in M\}.$$

Пример 3.112. Пусть

$$A = \{(a, 0.2); (b, 0.8); (c, 0.5); (d, 0.1); (e, 0.25)\}.$$

Тогда

$$A_{0.25} = \{(b, 0.8); (c, 0.5); (e, 0.25)\}.$$

Часто используется более компактная запись конечных или счетных нечетких множеств. Так, вместо приведенного выше в примере 3.112 представления множества A его можно записать следующим образом:

$$A = 0.2/a + 0.8/b + 0.5/c + 0.1/d + 0.25/e. \quad (3.45)$$

В дальнейшем для представления нечетких множеств будем использовать обе записи.

Очевидно, справедливо следующее свойство.

Лемма 3.21. $A_{\alpha_1} \subseteq A_{\alpha_2}$ тогда и только тогда, когда $\alpha_1 \leq \alpha_2$.

Справедлива теорема о декомпозиции.

Теорема 3.40 (о декомпозиции). Любое нечеткое множество $A \subseteq M$ можно представить в виде

$$A = \bigcup_{\alpha \in [0,1]} \alpha A_{\alpha}, \quad (3.46)$$

где A_{α} означает нечеткое множество со следующей функцией принадлежности:

$$\mu_{\alpha A_{\alpha}}(x) = \begin{cases} \alpha, & x \in A_{\alpha}, \\ 0, & x \notin A_{\alpha}. \end{cases} \quad (3.47)$$

Разложение нечеткого множества в виде (3.46) называется *декомпозицией* нечеткого множества A .

Пример 3.113. Пусть

$$A = \{(a, 0.1); (b, 0.3); (c, 0.7); (d, 0.8); (e, 1.0)\}.$$

Тогда

$$A = 0.1 \times A_{0.1} \cup 0.3 \times A_{0.3} \cup 0.7 \times A_{0.7} \cup 0.8 \times A_{0.8} \cup 1.0 \times A_{1.0},$$

где

$$\begin{cases} A_{0.1} = \{(a, 0.1); (b, 0.1); (c, 0.1); (d, 0.1); (e, 0.1)\}, \\ A_{0.3} = \{(b, 0.3); (c, 0.3); (d, 0.3); (e, 0.3)\}, \\ A_{0.7} = \{(c, 0.7); (d, 0.7); (e, 0.7)\}, \\ A_{0.8} = \{(d, 0.8); (e, 0.8)\}, \\ A_{1.0} = \{(e, 1.0)\}. \end{cases}$$

3.4.1.3. Операции над нечеткими множествами

Определение 3.85. Определим для нечетких множеств отношения включения¹ (\subset , \subseteq) и равенства ($=$) следующим образом:

$$A \subset B : \mu_A(x) < \mu_B(x), \forall x \in M,$$

$$A \subseteq B : \mu_A(x) \leq \mu_B(x), \forall x \in M,$$

$$A = B : \mu_A(x) = \mu_B(x), \forall x \in M.$$

Пример 3.114. Пусть

$$A = \{(a, 0.0); (b, 0.1); (c, 0.5); (d, 0.9); (e, 1.0)\}.$$

Элемент a не принадлежит множеству A , элемент b принадлежит ему в малой степени, элемент c более или менее принадлежит, элемент d принадлежит в значительной степени, e является элементом A .

Определим для нечетких множеств, как и для обычных, основные логические операции. Пусть A и B — нечеткие подмножества одного универсума M .

• **Объединение** множеств $A \cup B$ определяется функцией принадлежности

$$\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}, \forall x \in M.$$

Пример 3.115. Пусть нечеткие множества A и B определены следующим образом:

$$A = \{(a, 1.0); (b, 1.0); (c, 0.9); (d, 0.8); (e, 0.6); (f, 0.5); (g, 0.4); (h, 0.2)\},$$

$$B = \{(a, 0.5); (b, 1.0); (c, 0.6); (d, 0.4); (e, 0.2); (f, 0.0); (g, 0.0); (h, 0.0)\}.$$

Тогда нечеткое множество $C = A \cup B$ будет равно:

$$C = \{(a, 1.0); (b, 1.0); (c, 0.9); (d, 0.8); (e, 0.6); (f, 0.5); (g, 0.4); (h, 0.2)\}.$$

• **Пересечение** множеств $A \cap B$ определяется функцией принадлежности

$$\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}, \forall x \in M.$$

Пример 3.116. Рассмотрим нечеткие множества A и B из примера 3.115. Тогда нечеткое множество $D = A \cap B$ будет равно:

$$D = \{(a, 0.5); (b, 1.0); (c, 0.6); (d, 0.4); (e, 0.2); (f, 0.0); (g, 0.0); (h, 0.0)\}.$$

¹ Иногда говорят, что B «доминирует» над A .

- **Разность** множеств $A \setminus B$ определяется функцией принадлежности

$$\mu_{A \setminus B}(x) = \max\{\mu_A(x) - \mu_B(x), 0\}.$$

Пример 3.117. Рассмотрим нечеткие множества A и B из примера 3.115. Тогда нечеткое множество $E = A \setminus B$ будет равно:

$$E = \{(a, 0.5); (b, 0.0); (c, 0.3); (d, 0.4); (e, 0.4); (f, 0.5); (g, 0.4); (h, 0.2)\}.$$

- **Дополнение** $\bar{A} = M \setminus A$ множества A определяется функцией принадлежности

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x), \quad \forall x \in M.$$

- **Возведение в степень** α множества A определяется функцией принадлежности

$$\mu_{A^\alpha}(x) = \mu_A^\alpha(x), \quad \forall x \in M.$$

Частными случаями возведения в степень являются $\alpha = 2$ — операция **концентрирования**¹, обозначается $\text{CON}(A)$ и $\alpha = 0.5$ — операция **размывания**, обозначается $\text{DIL}(A)$.

В результате применения операции концентрирования к множеству A снижается степень нечеткости описания, причем для элементов с высокой степенью принадлежности это уменьшение относительно мало, а для элементов с малой степенью принадлежности относительно велико. Операция размывания увеличивает степень нечеткости исходного нечеткого множества.

Пример 3.118. Пусть $A = \{(a, 0.3); (b, 0.6); (c, 1.0)\}$. Тогда

$$\text{CON}(A) = \{(a, 0.09); (b, 0.36); (c, 1.0)\},$$

$$\text{DIL}(A) = \{(a, 0.548); (b, 0.775); (c, 1.0)\}.$$

Операция *контрастной интенсификации* обозначается $\text{INT}(A)$ и определяется следующим образом:

$$\mu_{\text{INT}(A)}(x) = \begin{cases} 2\mu_A(x)^2, & 0 \leq \mu_A(x) \leq 0.5, \\ 1 - 2(1 - \mu_A(x))^2, & 0.5 \leq \mu_A(x) \leq 1. \end{cases}$$

Эта операция отличается от концентрирования тем, что она увеличивает те значения $\mu_A(x)$, которые больше 0.5 и уменьшает те, которые меньше 0.5. Таким образом, контрастная интенсификация уменьшает нечеткость множества A .

¹ Аналоги в естественном языке «очень» и «не очень».

Пример 3.119. Определим нечеткое множество A — множество вещественных чисел, близких к числу π , характеристической функцией

$$\mu_A(x) = \frac{1}{(1 + |x - \pi|)^m}, \quad m \geq 1.$$

Например, для описания множества чисел, **не очень далеких** от π , положить $m = 2$. Для описания множества чисел, **очень близких** к π , можно провести концентрирование предыдущей функции принадлежности, то есть положить $m = 4$ (см. рисунок 3.24a). Увеличение степени нечеткости исходного нечеткого множества показано на рисунке 3.24b.

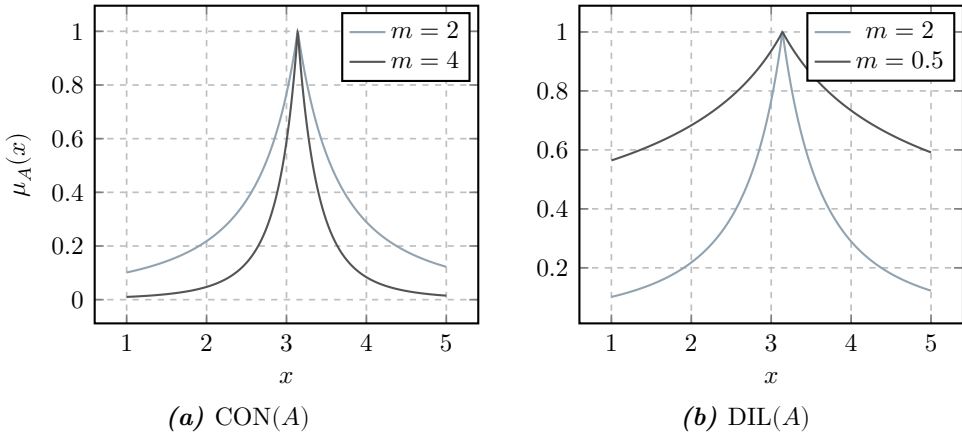


Рис. 3.24.

Для введенных операций над нечеткими множествами справедливы **почти все** законы классической теории множеств (см. таблицу 3.48).

Таблица 3.48.

$A \cup (A \cap B)$	поглощение
$A \cap (A \cup B)$	
$A \cup A = A$	идемпотентность
$A \cap A = A$	
$A \cup B = B \cup A$	коммутативность
$A \cap B = B \cap A$	
$A \cup (B \cup C) = (A \cup B) \cup C$	ассоциативность

Продолжение таблицы 3.48

$A \cap (B \cap C) = (A \cap B) \cap C$	
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	дистрибутивность
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	
$\overline{A \cup B} = \bar{A} \cap \bar{B}$	законы де Моргана
$\overline{A \cap B} = \bar{A} \cup \bar{B}$	
$\overline{\bar{A}} = A$	двойное дополнение

Для нечетких множеств **не справедливы** законы противоречия и исключенного третьего:

$$A \cap \bar{A} \neq \emptyset, \quad A \cup \bar{A} \neq M. \quad (3.48)$$

Для доказательства (3.48) рассмотрим пример.

Пример 3.120. Рассмотрим универсум $M = \{a, b, c, d, e, f\}$. Пусть нечеткое множество A имеет вид

$$A = \{(a, 1.0); (b, 0.8); (c, 0.6); (d, 0.4); (e, 0.2); (f, 0.0)\}.$$

Тогда по определению множество \bar{A} имеет следующий вид

$$\bar{A} = \{(a, 0.0); (b, 0.2); (c, 0.4); (d, 0.6); (e, 0.8); (f, 1.0)\}.$$

Находим пересечение нечетких множеств

$$A \cap \bar{A} = \{(a, 0.0); (b, 0.2); (c, 0.4); (d, 0.4); (e, 0.2); (f, 0.0)\} \neq \emptyset.$$

Аналогично находим объединение нечетких множеств

$$A \cup \bar{A} = \{(a, 1.0); (b, 0.8); (c, 0.6); (d, 0.6); (e, 0.8); (f, 1.0)\} \neq M.$$

Для нечетких множеств часто вводят понятие дизъюнктивной суммы.

Определение 3.86. *Дизъюнктивная сумма* нечетких множеств A и B , заданных на универсальном множестве M , — это нечеткое множество

$$A \oplus B = (A \setminus B) \cup (B \setminus A) = (A \cap \bar{B}) \cup (\bar{A} \cap B)$$

с функцией принадлежности, заданной следующим образом:

$$\mu_{A \oplus B}(x) = \max \{ \min \{ \mu_a(x); 1 - \mu_B(x) \}; \min \{ 1 - \mu_A(x); \mu_B(x) \} \}$$

Пример 3.121. Пусть

$$A = 0.9/a + 1/b + 0.6/d, \quad B = 0.7/a + 1/c + 0.4/d.$$

Тогда

$$A \oplus B = 0.97/a + 1/b + 1/c + 0.76/d.$$

Упражнение 3.21. Пусть задано n нечетких множеств $A_i, i \in 1 : n$ универсального множества M . Обозначим

$$B = A_1 \cup \dots \cup A_n, \quad C = A_1 \cap \dots \cap A_n, \quad i \in 1 : n.$$

Показать, что

$$\mu_B(x) = \max \{ \mu_{A_1}(x), \dots, \mu_{A_n}(x) \}, \quad \mu_C(x) = \min \{ \mu_{A_1}(x), \dots, \mu_{A_n}(x) \}.$$

Определение 3.87. Пусть A_1, \dots, A_n — нечеткие множества универсальных множеств M_1, \dots, M_n соответственно, а $\omega_1, \dots, \omega_n$ — неотрицательные числа такие, что $\sum_{i=1}^n \omega_i = 1$.

Выпуклой комбинацией нечетких множеств A_1, \dots, A_n называется нечеткое подмножество A множества $M = M_1 \times \dots \times M_n$ с n -мерной функцией принадлежности:

$$\mu_A(x_1, \dots, x_n) = \omega_1 \mu_{A_1}(x_1) + \dots + \omega_n \mu_{A_n}(x_n) = \sum_{i=1}^n \omega_i \mu_{A_i}(x_i).$$

Пример 3.122. Пусть для универсальных множеств $M_1 = \{a, b\}$ и $M_2 = \{b, c\}$ определены нечеткие множества A_1 и A_2 :

$$A_1 = 0.5/a + 1/b, \quad A_2 = 0.2/b + 1.0/c, \quad \omega_1 = 0.4, \omega_2 = 0.6.$$

Тогда

$$\begin{aligned} A &= (0.5 \times 0.4 + 0.2 \times 0.6)/(a; b) + (0.5 \times 0.4 + 1 \times 0.6)/(a; c) + \\ &+ (1.0 \times 0.4 + 0.2 \times 0.6)/(b; b) + (1 \times 0.4 + 1.0 \times 0.6)/(b; c) = \\ &= 0.32/(a; b) + 0.8/(a; c) + 0.52/(b; b) + 1.0/(b; c). \end{aligned}$$

Рассмотрим возможность увеличения нечеткости множества. Для этого предварительно определим операцию умножения нечеткого множества на число.

Пусть A — нечеткое множество, определенное на универсальном множестве M .

Определение 3.88. Рассмотрим число $\alpha > 0$ такое, что

$$\alpha \max_{x \in M} \{\mu_A(x)\} \leq 1,$$

тогда функция принадлежности нечеткого множества αA определяется как

$$\mu_{\alpha A}(x) = \alpha \mu_A(x), \quad \forall x \in M.$$

Определение 3.89. Пусть для всех $x \in M$ определены нечеткие множества $K(x)$. Совокупность всех $K(x)$ называется *ядром оператора Φ увеличения нечеткости*. Результатом действия оператора Φ на нечеткое множество A является нечеткое множество вида:

$$\Phi(A, K) = \bigcup_{x \in M} \mu_A(x) K(x).$$

Пример 3.123. Пусть

$$\begin{aligned} M &= \{a, b, c, d\}, \quad A = 0.8/a + 0.6/b + 0.2/c + 0.0/d, \\ K(a) &= 1.0/a + 0.4/b, \quad K(b) = 1.0/b + 0.4/c + 0.4/d, \\ K(c) &= 1.0/c + 0.5/d, \quad K(d) = 1.0/d. \end{aligned}$$

Тогда

$$\begin{aligned} \Phi(A, K) &= \mu_A(a)K(a) \cup \mu_A(b)K(b) \cup \mu_A(c)K(c) \cup \mu_A(d)K(d) = \\ &= 0.8(1.0/a + 0.4/b) \cup 0.6(1.0/b + 0.4/c + 0.4/d) \cup \\ &\quad \cup 0.2(1.0/c + 0.5/d) \cup 0.0(1.0/d) = \\ &= (0.8/a + 0.32/b) \cup (0.6/b + 0.24/c + 0.24/d) \cup \\ &\quad \cup (0.2/c + 0.1/d) \cup (0.0/d) = \\ &= 0.8/a + 0.6/b + 0.24/c + 0.24/d. \end{aligned}$$

3.4.1.4. Мера нечеткости множеств

Формализация понятия меры нечеткости множества имеет важное практическое и теоретическое значение.

Для определения степени нечеткости множества Р. Егер ввел определение меры нечеткости, сводящейся к измерению уровня различия между множеством A и его дополнением \bar{A} . Он исходил из того, что единственным

коренным отличием алгебры нечетких множеств от обычной булевой алгебры является непустота пересечения множеств A и \bar{A} , то есть для нечетких множеств

$$A \cap \bar{A} = B \neq \emptyset.$$

Очевидно, что чем ближе A к \bar{A} , тем больше B и тем сильнее A отличается от четкого множества. На основании этого Р. Егер предложил для описания меры четкости нечеткого множества A следующее выражение.

Определение 3.90. Мера Егера степени нечеткости n -элементного множества A в метрике ρ , обозначаемая $FUZ_\rho(A)$, определяется выражением

$$FUZ_\rho(A) = 1 - \frac{D_\rho(A, \bar{A})}{n^{1/\rho}}, \quad (3.49)$$

где $D_\rho(A, \bar{A})$ — это мера расстояния между множествами A и \bar{A} , содержащими n элементов.

Значение $\rho = 1$ соответствует метрике Хемминга, в которой

$$D_1(A, B) = \sum_{i=1}^n |\mu_A(x_i) - \mu_B(x_i)|, \quad D_1(A, \bar{A}) = \sum_{i=1}^n |2\mu_A(x_i) - 1|, \quad (3.50)$$

а значение $\rho = 2$ соответствует метрике Евклида, в которой

$$D_2(A, B) = \sqrt{\sum_{i=1}^n (\mu_A(x_i) - \mu_B(x_i))^2}, \quad D_2(A, \bar{A}) = \sqrt{\sum_{i=1}^n (2\mu_A(x_i) - 1)^2}. \quad (3.51)$$

Пример 3.124. Пусть

$$A = \{(a, 0.1); (b, 0.5); (c, 0.8); (d, 1.0); (e, 0.8); (f, 0.5); (g, 0.1)\}.$$

Тогда

$$\bar{A} = \{(a, 0.9); (b, 0.5); (c, 0.2); (d, 0.0); (e, 0.2); (f, 0.5); (g, 0.9)\}.$$

В соответствии с определением меры Егера (3.49), используя (3.50) и (3.51), получаем

$$FUZ_1(A) = 1 - \frac{1}{7}(0.8 + 0.0 + 0.6 + 1.0 + 0.6 + 0.0 + 0.8) = 0.457,$$

$$FUZ_2(A) = 1 - \sqrt{\frac{(0.64 + 0.0 + 0.36 + 1.0 + 0.36 + 0.0 + 0.64)}{7}} = 0.347.$$

Другую меру нечеткости предложил Б. Коско. Она основана на понятии кардинального числа множества (определение 3.44). В соответствии с этой мерой:

$$FUZ(A) = \frac{\text{card}(A \cap \bar{A})}{\text{card}(A \cup \bar{A})}. \quad (3.52)$$

Пример 3.125. Для множества из примера 3.124 получаем меру Коско, равную:

$$FUZ(A) = \frac{0.1 + 0.5 + 0.2 + 0.0 + 0.2 + 0.5 + 0.1}{0.9 + 0.5 + 0.8 + 1.0 + 0.8 + 0.5 + 0.9} = \frac{1.6}{5.4} = 0.296.$$

Замечание 3.37

Заметим, что обе меры для четких множеств дают один и тот же нулевой результат, так как в мере Коско $\text{card}(A \cap \bar{A}) = 0$, а для меры Эгера $D_p(A, \bar{A}) = \sqrt[p]{n}$ и вследствие (3.49) дает в результате также $FUZ_p(A) = 0$.

3.4.1.5. Функции принадлежности

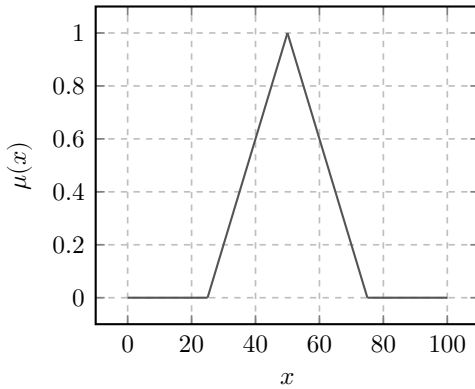
Введенное определение 3.83 для нечеткого множества не накладывает ограничений на выбор функции принадлежности. Однако на практике целесообразно использовать аналитическое представление функции принадлежности $\mu_A(x)$ нечеткого множества A .

Существует свыше десятка типовых форм кривых для задания функций принадлежности. Выделяют следующие типовые функции принадлежности.

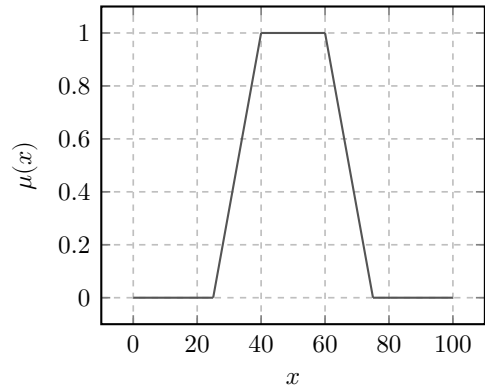
• **Треугольная функция** принадлежности определяется тройкой чисел (a, b, c) , $a < b < c$, и ее значение в точке x вычисляется согласно выражению

$$\mu_A(x) = \begin{cases} 1 - \frac{b-x}{b-a}, & a \leq x \leq b, \\ 1 - \frac{x-b}{c-b}, & b \leq x \leq c, \\ 0, & \text{в остальных случаях.} \end{cases}$$

При $(b-a) = (c-b)$ имеем случай симметричной треугольной функции принадлежности, которая может быть однозначно задана двумя параметрами из тройки (a, b, c) . На рисунке 3.25а приведен график треугольной функции принадлежности при $a = 25$, $b = 50$, $c = 75$.



(a) Треугольная функция



(b) Трапециoidalная функция

Рис. 3.25.

• **Трапециoidalная функция** принадлежности. Аналогично предыдущему случаю для задания трапециoidalной функции принадлежности необходима четверка чисел (a, b, c, d) , $a < b < c < d$:

$$\mu_A(x) = \begin{cases} 1 - \frac{b-x}{b-a}, & a \leq x \leq b, \\ 1, & b \leq x \leq c, \\ 1 - \frac{x-c}{d-c}, & c \leq x \leq d, \\ 0, & \text{в остальных случаях.} \end{cases}$$

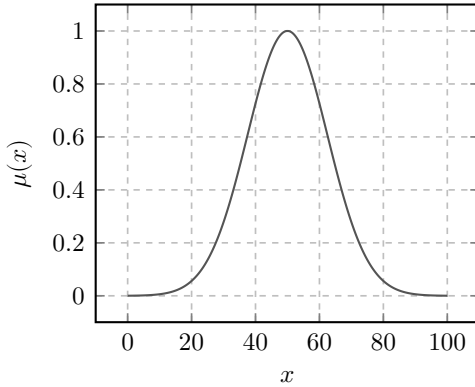
На рисунке 3.25b приведен график трапециoidalной функции принадлежности при $a = 25$, $b = 40$, $c = 60$, $d = 75$.

Треугольные и трапециoidalные функции принадлежности, как правило, используются для задания неопределенностей типа: «приблизительно равно», «среднее значение», «расположен в интервале», «подобен объекту», «похож на предмет» и т. п.

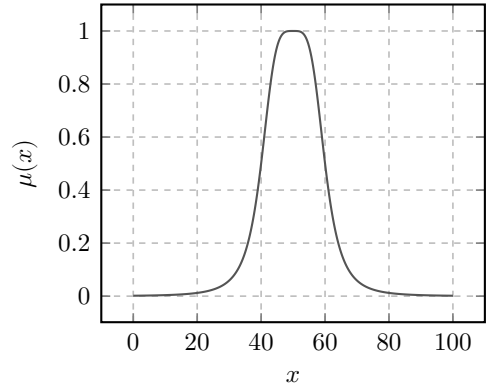
• **Гауссова функция** принадлежности. Функция принадлежности гауссова типа описывается выражением

$$\mu_A(x) = e^{-\left(\frac{x-c}{\sigma}\right)^2}$$

и зависит от двух параметров. Параметр c обозначает центр нечеткого множества, а параметр σ отвечает за крутизну функции. На рисунке 3.26a приведен график гауссовой функции принадлежности при $c = 50$, $\sigma = 12.5$.



(a) Гауссова функция



(b) Колоколообразная функция

Рис. 3.26.

К гауссовым функциям принадлежности часто относят также так называемую *колоколообразную (bell-shaped) функцию*, которая в общем случае задается аналитически следующим выражением:

$$\mu_A(x) = \frac{1}{1 + \left| \frac{x - c}{a} \right|^{2b}},$$

где a, b, c — числовые параметры, принимающие произвольные вещественные значения и упорядоченные отношением параметров: $a < b < c, b > 0$. Параметры геометрически интерпретируются следующим образом: a — коэффициент концентрации функции принадлежности, b — коэффициент крутизны функции принадлежности, c — координата максимума функции принадлежности.

На рисунке 3.26b приведен график колоколообразной функции принадлежности при $a = 10, b = 2, c = 50$.

• **Сплайн-функция** принадлежности. Сплайн-функция (иногда ее называют *Z-образной функцией*) принадлежности в общем случае аналитически задается следующим выражением:

$$\mu_A(x) = \begin{cases} 1, & x < a, \\ \frac{1}{2} + \frac{1}{2} \cos \left(\frac{x - a}{b - a} \pi \right), & a \leq x \leq b, \\ 0, & x > b, \end{cases}$$

где a, b — некоторые числовые параметры, принимающие произвольные вещественные значения и упорядоченные отношением: $a < b$. На рисунке 3.27а приведен график сплайн-функции принадлежности при $a = 25, b = 75$.

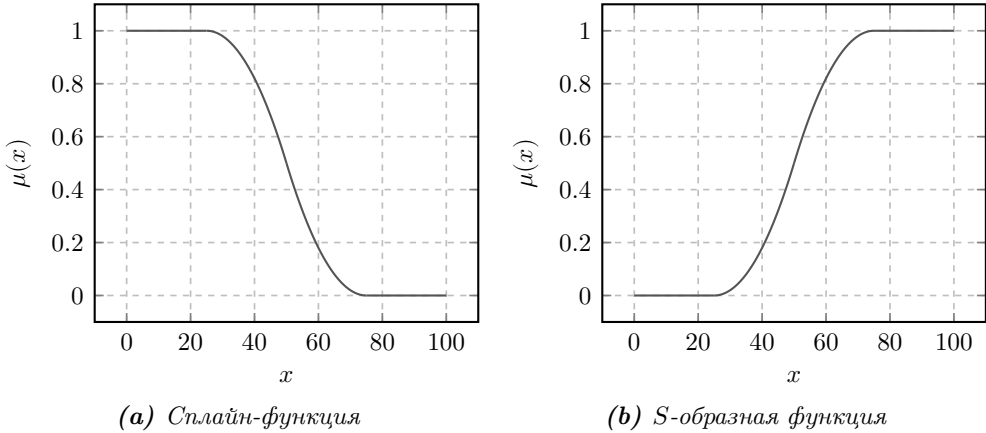


Рис. 3.27.

Сплайн-функции принадлежности используются для задания неопределенностей типа: «малое количество», «небольшое значение», «незначительная величина», «низкий уровень» и т.п.

- **S-образная функция** принадлежности. S-образная функция принадлежности в общем виде может быть задана аналитически следующим выражением:

$$\mu_A(x) = \begin{cases} 0, & a \leq x, \\ 1 - 2 \left(\frac{x - a}{b - a} \right)^2, & a < x \leq \frac{a + b}{2}, \\ 2 \left(\frac{b - x}{b - a} \right)^2, & \frac{a + b}{2} < x < b, \\ 1, & x \geq b, \quad a < b. \end{cases}$$

S-образные функции принадлежности применяются для задания неопределенностей типа: «большое количество», «большое значение», «значительная величина», «высокий уровень» и т. п. На рисунке 3.27б приведен график S-образной функции принадлежности при $a = 25, b = 75$.

К типу S-образных и одновременно Z-образных функций принадлежности может быть отнесена так называемая *сигмоидальная* функция (сигмоид), которая в общем случае задается аналитически следующим выраже-

нием:

$$\mu_A(x) = \frac{1}{1 + e^{-a(x-b)}}.$$

При этом в случае $a > 0$ может быть получена S -образная функция принадлежности, а в случае $a < 0$ — соответственно Z -образная функция принадлежности.

Существует множество других функций принадлежности нечетких множеств, заданных как композиции вышеупомянутых базовых функций (двойная гауссова, двойная сигмоидальная и т. п.) либо как комбинации по участкам возрастания и убывания (сигмоидально-гауссова, сплайн-треугольная и т. п.).

3.4.1.6. Методы построения функций принадлежности

Наиболее распространенными методами построения функций принадлежности являются прямые (*метод прямого оценивания*) и косвенные (*обратный метод оценивания*) методы. При использовании прямого метода построения каждому $x \in X$ экспертом задается значение функции принадлежности $\mu_A(x)$.

• **Прямые методы** характеризуются тем, что эксперт непосредственно задает правила определения значений функции принадлежности $\mu_A(x)$, характеризующей элемент x . Эти значения согласуются с его предпочтениями на множестве элементов M следующим образом:

- 1) для любых $x_1, x_2 \in M$, $\mu_A(x_1) < \mu_A(x_2)$ тогда и только тогда, когда x_2 предпочтительнее x_1 , т. е. в большей степени характеризуется свойством A ;
- 2) для любых $x_1, x_2 \in M$, $\mu_A(x_1) = \mu_A(x_2)$ тогда и только тогда, когда x_1 и x_2 безразличны относительно свойства A .

Прямые методы построения удобны при решении задач, для которых свойства физических величин могут быть измерены, например скорость, время, расстояние, давление и т. д. Задание абсолютно точных значений функций принадлежности не требуется, достаточно определить тип функции принадлежности и характерные значения множества. При необходимости более точного определения функции принадлежности можно воспользоваться последующим анализом результатов для коррекции нечеткой модели.

• **Косвенные методы** построения функций принадлежности используются при решении задач, для которых свойства физических величин не

могут быть измерены. Эти методы основаны на экспертных оценках. В качестве простейшего рассмотрим следующий метод, называемый *частотным*.

Пусть имеется n экспертов. Допустим, что на вопрос о том, принадлежит ли элемент x нечеткому множеству A , $n_1 \leq n$ экспертов отвечают положительно. В этом случае полагаем

$$\mu_A(x) = \frac{n_1}{n}.$$

Наибольшее распространение среди косвенных методов получил метод *парных сравнений* или *парных соотношений*.

Метод основан на обработке матрицы оценок, отражающих мнение эксперта об относительной принадлежности элементов множеству или степени выраженности у них некоторого оцениваемого свойства. Потребуем, чтобы для всех элементов множества A для определяемой функции μ_A выполнялось равенство

$$\sum_{i=1}^n \mu_A(x_i) = 1. \quad (3.53)$$

Оценку преимущества элемента x_i над элементом x_j с точки зрения свойства A обозначим через a_{ij} . Для обеспечения согласованности примем $a_{ij} = 1/a_{ji}$. Оценки a_{ij} составляют матрицу парных сравнений $S = \|a_{ij}\|$. По построению матрица S является диагональной ($a_{ii} = 1$) и обратно симметричной.

Оценки a_{ij} , как правило, определяются по девятибалльной *шкале Сати* (таблица 3.49).

Таблица 3.49.

a_{ij}	Степень преимущества
1	Отсутствует преимущество элемента x_i над элементом x_j
3	Имеется слабое преимущество x_i над x_j
5	Имеется существенное преимущество x_i над x_j
7	Имеется явное преимущество x_i над x_j
9	Имеется абсолютное преимущество x_i над x_j

Значения 2, 4, 6, 8 — промежуточные сравнительные оценки.

Далее найдем $W = (w_1, \dots, w_n)$ — нормированный собственный вектор матрицы S , соответствующий максимальному собственному числу λ_{max} :

$$SW = \lambda_{max}W, \quad w_1 + \dots + w_n = 1. \quad (3.54)$$

Положим $\mu_A(x_i) = w_i, i \in 1 : n$. В силу (3.54) условие (3.53) на μ_A будет выполнено.

Большое число численных примеров для этого метода можно найти в [52].

3.4.2. Нечеткие бинарные отношения

3.4.2.1. Основные понятия

Определение 3.91. Переноса определение нечетких множеств на отношения, определим бинарное нечеткое отношение как нечеткое подмножество $R \subseteq M \times M$. Таким образом, под нечетким отношением R будем понимать функцию принадлежности $\mu_R(x, y)$ такую, что

$$\mu_R : M \times M \rightarrow [0, 1].$$

Значение функции принадлежности понимается как степень выполнения отношения xRy .

Существуют различные способы, которыми в общем случае формально могут быть заданы те или иные нечеткие бинарные отношения. Наибольшее распространение из них получили следующие.

- **В форме списка** с явным перечислением всех пар нечеткого бинарного отношения и соответствующих им значений функции принадлежности:

$$\{(\langle x_1, y_1 \rangle, \mu(x_1, y_1)), \dots, (\langle x_k, y_k \rangle, \mu(x_k, y_k))\}, \quad \mu(x_i, y_i) > 0, \forall i \in 1 : k.$$

При этом для сокращения подобной записи пары с нулевыми значениями функции принадлежности не указываются в данном списке.

- **Аналитически** в форме некоторого математического выражения для соответствующей функции принадлежности этого нечеткого отношения. Этот способ может быть использован для задания произвольных нечетких отношений как с конечным, так и с бесконечным числом элементов. Рассмотрим пример.

Пример 3.126. Пусть X — множество всех вещественных чисел. Отношение $x \gg y$ (x много больше y) можно задать функцией принадлежности:

$$\mu_R(x, y) = \begin{cases} 0, & x \leq y, \\ \frac{1}{1 + \frac{1}{(x - y)^2}}, & x > y. \end{cases}$$

• **В форме матрицы нечеткого отношения.** Этот способ основан на представлении нечеткого бинарного отношения с конечным числом элементов в форме матрицы M_R , строки которой соответствуют первым элементам пары, а столбцы — вторым рассматриваемого нечеткого отношения. При этом элементами матрицы являются соответствующие значения функции принадлежности данного отношения.

Пример 3.127. Рассмотрим нечеткое бинарное отношение R , заданное на множестве $\{a, b, c, d\}$. Матрица отношения M_R имеет вид (таблица 3.50).

Таблица 3.50.

	a	b	c	d
a	0.1	0.9	0.7	0.8
b	0.3	0.5	0.0	0.2
c	0.0	0.3	0.4	0.1
d	0.0	0.0	0.9	0.2

• **В форме ориентированного нечеткого графа.**

Определение 3.92. *Ориентированный нечеткий граф* есть тройка объектов $G = (V, E, \mu_G)$, где $V = \{v_1, \dots, v_n\}$ — множество вершин нечеткого графа, $E = \{e_1, \dots, e_m\}$ — множество дуг нечеткого графа, μ_G — функция принадлежности дуг данному нечеткому графу, т. е. $\mu_G : E \rightarrow [0, 1]$.

Таким образом, ориентированный нечеткий граф представляет собой нагруженный, или взвешенный ориентированный, граф (см. определение 2.57).

Пример 3.128. Граф нечеткого бинарного отношения из предыдущего примера 3.127 представлен на рисунке 3.28.

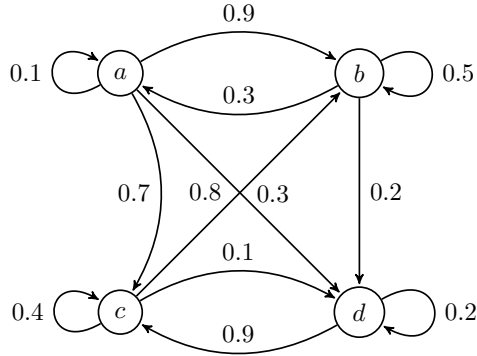


Рис. 3.28.

Пример 3.129. Пусть $X = \{1, 2, 3\}$. Нечеткое отношение *приблизительно равняться*, заданное на множестве X , определим с помощью списка:

$$\begin{aligned} \mu_R(1, 1) &= \mu_R(2, 2) = \mu_R(3, 3) = 1, \\ \mu_R(1, 2) &= \mu_R(2, 1) = \mu_R(2, 3) = \mu_R(3, 2) = 0.8, \\ \mu_R(1, 3) &= \mu_R(3, 1) = 0.3. \end{aligned}$$

Функция принадлежности в аналитическом виде может быть задана следующим образом:

$$\mu_R(u, v) = \begin{cases} 1.0, & u = v, \\ 0.8, & |u - v| = 1, \\ 0.3 & |u - v| = 2. \end{cases}$$

Матрица отношения имеет вид (таблица 3.51).

Таблица 3.51.

	1	2	3
1	1.0	0.8	0.3
2	0.8	1.0	0.8
3	0.3	0.8	1.0

Граф данного нечеткого бинарного отношения представлен на рисунке 3.29.

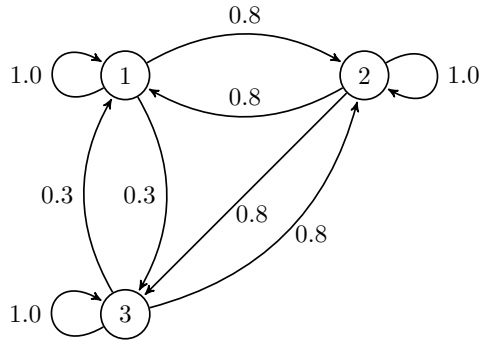


Рис. 3.29.

3.4.2.2. Свойства нечетких бинарных отношений

Различные типы нечетких бинарных отношений определяются с помощью свойств, аналогичных свойствам обычных отношений, причем для нечетких отношений можно указать различные способы обобщения этих свойств. В качестве основных свойств рассмотрим свойства, имеющие такую же алгебраическую запись, что и для обычных бинарных отношений.

Перейдем к изучению основных свойств нечетких отношений. Рассмотрим бинарное нечеткое отношение R , заданное на декартовом произведении $X \times X$.

- **Рефлексивность.** Отношение R называется *рефлексивным*, если выполняется равенство

$$\mu_R(x, x) = 1, \quad \forall x \in X. \quad (3.55)$$

- **Слабая рефлексивность.** Отношение R называется *слабо рефлексивным*, если выполняется неравенство

$$\mu_R(x, y) \leq \mu_R(x, x), \quad \forall x, y \in X. \quad (3.56)$$

- **Сильная рефлексивность.** Отношение R называется *сильно рефлексивным*, если выполняется неравенство

$$\mu_R(x, y) < 1, \quad \forall x, y \in X. \quad (3.57)$$

• **Антирефлексивность**. Отношение R называется *антирефлексивным*, если выполняется равенство

$$\mu_R(x, x) = 0, \forall x \in X. \quad (3.58)$$

• **Слабая антирефлексивность**. Отношение R называется *слабо антирефлексивным*, если выполняется неравенство

$$\mu_R(x, x) \leq \mu_R(x, y), \forall x, y \in X. \quad (3.59)$$

• **Сильная антирефлексивность**. Отношение R называется *сильно антирефлексивным*, если выполняется неравенство

$$\mu_R(x, y) > 0, \forall x, y \in X. \quad (3.60)$$

• **Симметричность**. Отношение R называется *симметричным*, если выполняется равенство

$$\mu_R(x, y) = \mu_R(y, x), \forall x, y \in X. \quad (3.61)$$

• **Асимметричность**. Отношение R называется *асимметричным*, если выполняется равенство

$$\min \{ \mu_R(x, y), \mu_R(y, x) \} = 0, \forall x, y \in X. \quad (3.62)$$

• **Антисимметричность**. Отношение R называется *антисимметричным*, если выполняется равенство

$$\min \{ \mu_R(x, y), \mu_R(y, x) \} = 0, \forall x, y \in X, x \neq y. \quad (3.63)$$

• **Сильная полнота**. Отношение R называется *сильно полным*, если выполняется равенство

$$\max \{ \mu_R(x, y), \mu_R(y, x) \} = 1, \forall x, y \in X. \quad (3.64)$$

• **Слабая полнота**. Отношение R называется *слабо полным*¹, если выполняется неравенство

$$\max \{ \mu_R(x, y), \mu_R(y, x) \} > 0, \forall x, y \in X, x \neq y. \quad (3.65)$$

¹ Линейным или связным.

• **Транзитивность.** Отношение R называется *транзитивным*, если выполняется неравенство

$$\mu_R(x, z) \geq \max_{y \in X} \{ \min \{ \mu_R(x, y), \mu_R(y, z) \} \}, \quad \forall x, y, z \in X. \quad (3.66)$$

• **Котранзитивность.** Отношение R называется *котранзитивным*, если выполняется неравенство

$$\mu_R(x, z) \leq \min_{y \in X} \{ \max \{ \mu_R(x, y), \mu_R(y, z) \} \}, \quad \forall x, y, z \in X. \quad (3.67)$$

Теорема 3.41. Для матрицы M_R бинарного нечеткого отношения R с конечным универсумом X справедливы следующие свойства.

1. Все элементы главной диагонали матрицы рефлексивного бинарного нечеткого отношения равны 1.
2. Все элементы главной диагонали матрицы антирефлексивного бинарного нечеткого отношения равны 0.
3. Матрица симметричного бинарного нечеткого отношения симметрична относительно главной диагонали.
4. Все элементы главной диагонали матрицы асимметричного бинарного нечеткого отношения равны 0.

Доказательство. Перечисленные свойства очевидным образом следуют из данных выше определений (3.55)–(3.67). ■

3.4.3. Нечеткая логика высказываний

3.4.3.1. Нечеткие высказывания

Определение 3.93. *Нечетким высказыванием* называется высказывание \tilde{F} , степень истинности которого $\mu(\tilde{F})$ можно оценить числом из интервала $[0, 1]$, $\mu(\tilde{F}) \in [0, 1]$. Если $\mu(\tilde{F}) = 0.5$, то высказывание называется *индифферентным*.

Нечеткой высказывательной переменной \tilde{X} называется нечеткое высказывание \tilde{X} , степень истинности которого может меняться в интервале $[0, 1]$.

Аналогично обычным (четким) высказываниям будем отождествлять нечеткое высказывание с его степенью истинности¹. Нечеткие высказывания и степень их истинности будем обозначать большими буквами с тильдой: $\tilde{F}, \tilde{G}, \tilde{X}, \tilde{Y}$ и т. д.

¹ Для четких высказываний это были два значения *true* и *false*.

Введем на множестве нечетких высказываний логические операции, аналогичные операциям логики высказываний (соглашения о приоритете и записи операций такие же, как и для обычных высказываний).

- **Отрицание** нечеткого высказывания:

$$\neg \tilde{F} = 1 - \tilde{F}. \quad (3.68)$$

Помимо приведенного выше основного определения нечеткого логического отрицания, введенного Заде (3.68), могут использоваться следующие альтернативные формулы:

$$\begin{aligned} \neg \tilde{F} &= \frac{1 - \tilde{F}}{1 + \lambda \tilde{F}}, \quad \lambda > -1, \quad \text{нечеткое } \lambda - \text{дополнение по Сугено,} \\ \neg \tilde{F} &= \sqrt[p]{1 - \tilde{F}}, \quad p > 0, \quad \text{нечеткое } p - \text{дополнение по Ягеру.} \end{aligned}$$

- **Конъюнкция** нечетких высказываний:

$$\tilde{F}\tilde{G} = \min \{ \tilde{F}, \tilde{G} \}. \quad (3.69)$$

Помимо приведенного выше основного определения логической конъюнкции, введенного Заде (3.69), могут использоваться следующие альтернативные формулы:

$$\begin{aligned} \tilde{F}\tilde{G} &= \max \{ \tilde{F} + \tilde{G} - 1, 0 \}, \quad \text{в базисе Лукашевича — Гилеса;} \\ \tilde{F}\tilde{G} &= \begin{cases} \tilde{G} & \text{при } \tilde{F} = 1, \\ \tilde{F} & \text{при } \tilde{G} = 1, \\ 0, & \text{в остальных случаях.} \end{cases} \quad \text{в базисе Вебера;} \end{aligned}$$

- **Дизъюнкция** нечетких высказываний:

$$\tilde{F} \vee \tilde{G} = \max \{ \tilde{F}, \tilde{G} \}. \quad (3.70)$$

Помимо приведенного выше основного определения логической дизъюнкции, введенного Заде (3.70), могут использоваться следующие альтер-

нативные формулы:

$$\begin{aligned} \tilde{F} \vee \tilde{G} &= \tilde{F} + \tilde{G} - \tilde{F}\tilde{G}, && \text{в базисе Бандлера — Кохоута;} \\ \tilde{F} \vee \tilde{G} &= \min \{ \tilde{F} + \tilde{G}, 1 \}, && \text{в базисе Лукашевича — Гилеса;} \\ \tilde{F} \vee \tilde{G} &= \begin{cases} \tilde{G} & \text{при } \tilde{F} = 0, \\ \tilde{F} & \text{при } \tilde{G} = 0, \\ 1, & \text{в остальных случаях.} \end{cases} && \text{в базисе Вебера;} \end{aligned}$$

- **Импликация** нечетких высказываний:

$$\tilde{F} \rightarrow \tilde{G} = \neg\tilde{F} \vee \tilde{G} = \max \{ 1 - \tilde{F}, \tilde{G} \}. \quad (3.71)$$

Помимо приведенного выше основного определения нечеткой импликации, введенного Заде (3.71), могут использоваться следующие альтернативные определения нечеткой импликации, предложенные различными исследователями в области теории нечетких множеств:

$$\begin{aligned} \tilde{F} \rightarrow \tilde{G} &= \max \{ 1 - \tilde{F}, \tilde{G} \}, && \text{Гедель;} \\ \tilde{F} \rightarrow \tilde{G} &= \min \{ \tilde{F}, \tilde{G} \}, && \text{Мамдани;} \\ \tilde{F} \rightarrow \tilde{G} &= \min \{ 1, 1 - \tilde{F} + \tilde{G} \}, && \text{Лукашевич;} \\ \tilde{F} \rightarrow \tilde{G} &= \min \{ \tilde{F} + \tilde{G}, 1 \}, && \text{Лукашевич — Гилес;} \\ \tilde{F} \rightarrow \tilde{G} &= \tilde{F}\tilde{G}, && \text{Бандлер — Кохоут;} \\ \tilde{F} \rightarrow \tilde{G} &= \max \{ \tilde{F}\tilde{G}, 1 - \tilde{F} \}, && \text{Вади.} \end{aligned}$$

Замечание 3.38

Общее число введенных определений нечеткой импликации не ограничивается приведенными выше. Большое количество работ по изучению различных вариантов нечеткой импликации обусловлено тем, что понятие нечеткой импликации является ключевым при нечетких выводах и принятии решений в нечетких условиях. Наибольшее применение при решении прикладных задач нечеткого управления находит нечеткая импликация Заде.

- **Эквивалентность** нечетких высказываний:

$$\tilde{F} \leftrightarrow \tilde{G} = \tilde{F}\tilde{G} \vee \neg\tilde{F}\neg\tilde{G} = \max \{ \min \{ 1 - \tilde{F}, 1 - \tilde{G} \}, \min \{ \tilde{F}, \tilde{G} \} \}. \quad (3.72)$$

Формулу (3.72) можно записать в другом виде:

$$\tilde{F} \leftrightarrow \tilde{G} = (\tilde{F} \rightarrow \tilde{G}) (\tilde{G} \rightarrow \tilde{F}) = \min \left\{ \max \{1 - \tilde{F}, \tilde{G}\}, \max \{\tilde{F}, 1 - \tilde{G}\} \right\}. \quad (3.73)$$

Для нечетких логических операций выполняются все законы логики высказываний, кроме законов противоречия и исключенного третьего, т. е.

$$\neg \tilde{F} \wedge \tilde{F} \neq 0, \quad \neg \tilde{F} \vee \tilde{F} \neq 1.$$

Пример 3.130. Найдем степень истинности высказывания \tilde{H} :

$$\tilde{H} = (\tilde{F} \vee \tilde{G}) \leftrightarrow (\tilde{F} \rightarrow \tilde{F}\tilde{G})$$

при $\tilde{F} = 0.75$, $\tilde{G} = 0.2$.

Обозначим $\tilde{F}_1 = (\tilde{F} \vee \tilde{G})$, а $\tilde{F}_2 = (\tilde{F} \rightarrow \tilde{F}\tilde{G})$. Согласно (3.68)–(3.71) получаем:

$$\begin{aligned} \tilde{F}_1 &= \max \{ \tilde{F}, \tilde{G} \} = 0.75, \\ \tilde{F}_2 &= \max \{ 1 - \tilde{F}, \tilde{F}\tilde{G} \} = \max \{ 0.25, \min \{ \tilde{F}, \tilde{G} \} \} = \\ &= \max \{ 0.25, \min \{ 0.75, 0.2 \} \} = 0.25. \end{aligned}$$

Отсюда, используя (3.72), имеем

$$\begin{aligned} \tilde{H} &= \tilde{F}_1 \leftrightarrow \tilde{F}_2 = \max \left\{ \min \{ 1 - \tilde{F}_1, 1 - \tilde{F}_2 \}, \min \{ \tilde{F}_1, \tilde{F}_2 \} \right\} = \\ &= \max \{ \min \{ 0.25, 0.75 \}, \min \{ 0.75, 0.25 \} \} = 0.25. \end{aligned}$$

3.4.3.2. Нечеткие формулы логики высказываний

Аналогично определению для формул логики высказываний введем индуктивное определение нечеткой логической формулы.

Определение 3.94. *Нечеткой логической формулой* называется:

- 1) любая нечеткая высказывательная переменная;
- 2) если \tilde{F} и \tilde{G} — нечеткие логические формулы, то

$$\neg \tilde{F}, \quad \tilde{F}\tilde{G}, \quad \tilde{F} \vee \tilde{G}, \quad \tilde{F} \rightarrow \tilde{G}, \quad \tilde{F} \leftrightarrow \tilde{G}$$

тоже нечеткие логические формулы.

Определение 3.95. Пусть $\tilde{F}(\tilde{X}_1, \dots, \tilde{X}_n)$ и $\tilde{G}(\tilde{X}_1, \dots, \tilde{X}_n)$ — две нечеткие логические формулы. *Степенью равносильности формул \tilde{F} и \tilde{G}*

называется величина

$$\mu(\tilde{F}, \tilde{G}) = \bigwedge_{(\alpha_1, \dots, \alpha_n)} \{ \tilde{F}(\alpha_1, \dots, \alpha_n) \leftrightarrow \tilde{G}(\alpha_1, \dots, \alpha_n) \}, \quad (3.74)$$

где конъюнкция берется по всем наборам степеней истинности $(\alpha_1, \dots, \alpha_n)$ нечетких переменных $(\tilde{X}_1, \dots, \tilde{X}_n)$.

Очевидно, что множество всех наборов степеней истинности имеет мощность континуум¹, в отличие от двузначной логики высказываний, где число всех наборов переменных конечно и равно 2^n .

Если $\mu(\tilde{F}, \tilde{G}) = 0.5$, то нечеткие формулы \tilde{F} и \tilde{G} называются *индифферентными*.

Если $\mu(\tilde{F}, \tilde{G}) > 0.5$, то нечеткие формулы \tilde{F} и \tilde{G} называются *нечетко равносильными*.

Если $\mu(\tilde{F}, \tilde{G}) < 0.5$, то нечеткие формулы \tilde{F} и \tilde{G} называются *нечетко неравносильными*.

Степенью неравносильности формул \tilde{F} и \tilde{G} называется величина

$$\bar{\mu}(\tilde{F}, \tilde{G}) = 1 - \mu(\tilde{F}, \tilde{G}).$$

Пример 3.131. Определим степень равносильности нечетких формул $\tilde{F} = \tilde{X} \rightarrow \tilde{Y}$ и $\tilde{G} = \neg(\tilde{X}\tilde{Y})$ при условии, что нечеткие переменные \tilde{X} и \tilde{Y} принимают значения степеней истинности из множества $\{0.1, 0.2\}$.

Рассмотрим все возможные наборы значений \tilde{X} и \tilde{Y} :

$$(0.1, 0.1), (0.1, 0.2), (0.2, 0.1), (0.2, 0.2). \quad (3.75)$$

С учетом (3.68)–(3.72) имеем

$$\begin{cases} \tilde{F} = \tilde{X} \rightarrow \tilde{Y} = \max\{1 - \tilde{X}, \tilde{Y}\}, \\ \tilde{G} = \neg(\tilde{X}\tilde{Y}) = 1 - (\tilde{X}\tilde{Y}) = 1 - \min\{\tilde{X}, \tilde{Y}\}. \end{cases}$$

Вычислим значения формул \tilde{F} и \tilde{G} на каждом из наборов (3.75):

$$\tilde{F}(0.1, 0.1) = \max\{1 - 0.1, 0.1\} = 0.9,$$

$$\tilde{F}(0.1, 0.2) = \max\{1 - 0.1, 0.2\} = 0.9,$$

$$\tilde{F}(0.2, 0.1) = \max\{1 - 0.2, 0.1\} = 0.8,$$

$$\tilde{F}(0.2, 0.2) = \max\{1 - 0.2, 0.2\} = 0.8$$

$$\tilde{G}(0.1, 0.1) = 1 - \min\{0.1, 0.1\} = 0.9,$$

¹ Мощность множества всех вещественных чисел.

$$\begin{aligned}\tilde{G}(0.1, 0.2) &= 1 - \min \{0.1, 0.2\} = 0.9, \\ \tilde{G}(0.2, 0.1) &= 1 - \min \{0.2, 0.1\} = 0.9, \\ \tilde{G}(0.2, 0.2) &= 1 - \min \{0.2, 0.2\} = 0.8.\end{aligned}$$

Вычислим теперь степень равносильности формул \tilde{F} и \tilde{G} . Согласно (3.73) и (3.74) имеем:

$$\begin{aligned}\tilde{F}(0.1, 0.1) &\leftrightarrow \tilde{G}(0.1, 0.1) = \min \{ \max \{1 - 0.9, 0.9\}, \max \{0.9, 1 - 0.9\} \} = 0.9, \\ \tilde{F}(0.1, 0.2) &\leftrightarrow \tilde{G}(0.1, 0.2) = \min \{ \max \{1 - 0.9, 0.9\}, \max \{0.9, 1 - 0.9\} \} = 0.9, \\ \tilde{F}(0.2, 0.1) &\leftrightarrow \tilde{G}(0.2, 0.1) = \min \{ \max \{1 - 0.8, 0.9\}, \max \{0.8, 1 - 0.9\} \} = 0.8, \\ \tilde{F}(0.2, 0.2) &\leftrightarrow \tilde{G}(0.2, 0.2) = \min \{ \max \{1 - 0.8, 0.8\}, \max \{0.8, 1 - 0.8\} \} = 0.8.\end{aligned}$$

Окончательно согласно (3.74) получим

$$\mu(\tilde{F}, \tilde{G}) = \min(0.9, 0.9, 0.8, 0.8) = 0.8,$$

таким образом, формулы \tilde{F} и \tilde{G} нечетко равносильны на заданных наборах степеней истинности.

Замечание 3.39

На других наборах степеней истинности нечетких переменных \tilde{X} и \tilde{Y} формулы \tilde{F} и \tilde{G} из примера 3.131 могут быть нечетко неравносильны.

Задачи и вопросы

1. Что такое нечеткое множество, заданное на универсуме M ?
2. Пусть имеются два нечетких множества A и B , заданные функциями принадлежности $\mu_A(x), \mu_B(x)$. Определите $A \cup B$.
3. Пусть имеются два нечетких множества A и B , заданные функциями принадлежности $\mu_A(x), \mu_B(x)$. Определите $A \cap B$.
4. Приведите пример функции принадлежности для нечеткого множества.
5. Что такое мера нечеткости для нечеткого множества.
6. Приведите пример нечеткого бинарного отношения, заданного графом или таблицей.

7. Дано универсальное множество

$$M = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$$

и нечеткие множества $A \subseteq M$ и $B \subseteq M$:

$$A = \{(x_1, 0.6); (x_2, 0.3); (x_3, 0.7); (x_4, 0.4); (x_5, 0.4); (x_6, 0.5); (x_7, 1.0)\},$$

$$B = \{(x_1, 0.4); (x_2, 0.6); (x_3, 0.5); (x_4, 0.4); (x_5, 0.5); (x_6, 0.5); (x_7, 0.7)\}.$$

Найти $A \cup B$, $A \cap B$, меры нечеткости A и B по Егеру.

3.5. Теория алгоритмов

Историческая справка

Понятие алгоритма, являющееся одним из основных понятий математики, возникло задолго до появления вычислительных машин. Арабский математик IX в. Аль-Хорезми (из города Хорезма на реке Аму-Дарья)^a в своей книге подробно описал индийскую арифметику. В ней он впервые выдвинул идею о том, что решение любой поставленной математической задачи может быть оформлено в виде последовательности выполняемых инструкций, т. е. может быть алгоритмизировано.

Триста лет спустя эту книгу перевели на латинский язык, и она стала первым учебником «индийской» (то есть нашей современной) арифметики для всей Европы. Переводчик, имя которого не сохранилось в истории, дал ей название *Algoritmi de numero Indorum*.

Кроме того, Аль-Хорезми написал книгу об общих правилах решения арифметических задач при помощи уравнений. Она называлась «Китаб ал-Джебр». Эта книга дала имя науке алгебре.

^a Полное имя Аль-Хорезми следующее: Аль-Хорезми Абу Абдалла Мухаммед бен Муса аль-Маджуси.

Формальные определения алгоритма появились в первой половине XX в. Можно выделить три основные алгоритмические модели.

1 Алгоритм представляется как некоторое детерминированное устройство, способное выполнять в каждый отдельный момент некоторые примитивные операции (раздел 3.5.1).

2 Понятие алгоритма связывают с вычислениями и числовыми функциями и рассматривают алгоритм с точки зрения того, что можно вычислить с его помощью (раздел 3.5.5).

3 Алгоритм рассматривается как преобразование слов в некотором алфавите, при этом элементарными операциями являются подстановки, т. е. замены части слова другим словом (раздел 3.5.6).

3.5.1. Машина Тьюринга и функции, вычислимые по Тьюрингу

Историческая справка

Алан Матисон Тьюринг (1912–1954) — английский математик, получил докторскую степень в Принстонском университете (США), где его научным руководителем был А. Черч. После получения докторской степени Тьюринг отклонил предложение Дж. фон Неймана остаться в США и вернулся в Кембридж.

В 1936 г. в трудах Лондонского математического общества появилась знаменитая работа Тьюринга «О вычислимых числах, с приложением к проблеме разрешимости», в которой он описал универсальную вычислительную машину, предна-

значенную для решения любых математических или логических задач. Устройство, впоследствии названное машиной Тьюринга, обладало основными свойствами современного компьютера. Кстати, именно Тьюринг впервые употребил термин «компьютер» применительно к машине (раньше так называли людей, проводивших расчеты на арифмометре).

Идея машины Тьюринга легла в основу современной теории алгоритмов.

С началом Второй мировой войны Тьюринг перешел на работу в правительственную Школу кодов и шифров (Government Code and Cypher School). Используя более ранние польские наработки, совместно с У. Уэлчманом раскрыл шифры германских ВВС, создав дешифровочную машину «Бомба», а затем взломал гораздо более сложный шифр, использовавшийся в шифровальных машинах «Энигма», которыми были оснащены германские подводные лодки. В этот период Тьюринг занимался также разработкой шифров для переписки У. Черчилля и Ф. Рузвельта.

Определение 3.96. *Машина Тьюринга* состоит из следующих элементов.

1. *Ленты*, разбитой на ячейки и бесконечной в обе стороны. В каждой ячейке ленты может быть записан один из символов конечного алфавита $A = \{\alpha_0, \alpha_1, \dots, \alpha_t\}$, называемого *внешним алфавитом*. Условимся считать, что символ α_0 является пустым символом. Иногда для обозначения пустого символа алфавита используют символы Λ или ϵ .

2. *Управляющего устройства*, которое может находиться в одном из *внутренних состояний* $Q = \{q_0, q_1, \dots, q_n\}$. Число элементов в Q характеризует объем внутренней памяти машины. В множестве Q выделены два специальных состояния: q_0 и q_1 , называемых *заключительным* и *начальным* состояниями соответственно. Машина всегда начинает работу в состоянии q_1 и всегда останавливается, попав в состояние q_0 (часто это состояние называют стоп-сигналом).

3. *Считывающей/пишущей головки*, которая может перемещаться вдоль ленты и в каждый момент времени обзрывает (считывает) одну из ее ячеек. Машина Тьюринга функционирует в дискретные моменты времени $t = 0, 1, 2, \dots$ и в зависимости от внутреннего состояния машины и считываемого символа на ленте:

- 1) записывает в эту ячейку символ внешнего алфавита;
- 2) сдвигает считывающую головку на один шаг влево (вправо) или оставляет ее на месте;
- 3) переходит в новое внутреннее состояние.

Схематично машина Тьюринга представлена на рисунке 3.30.

Таким образом, работа машины Тьюринга определяется системой *команд* вида

$$q_i \alpha_j \rightarrow q_k \alpha_l M_s, \quad (3.76)$$

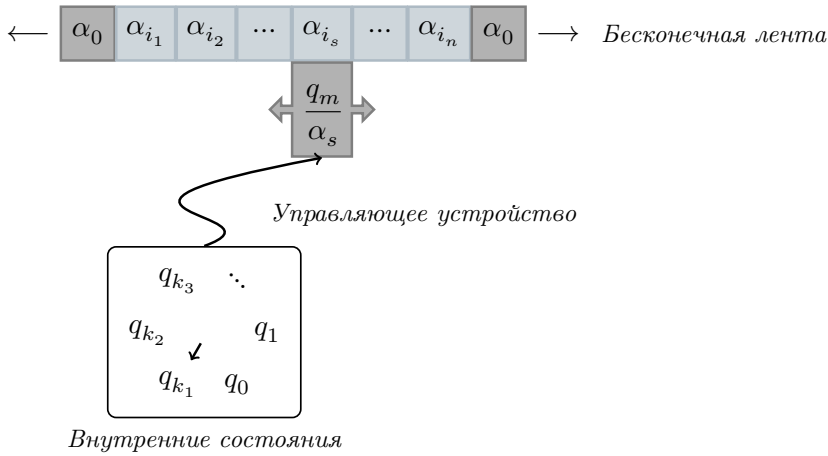


Рис. 3.30.

где q_i — текущее внутреннее состояние машины; a_j — считываемый символ; q_k — новое внутреннее состояние; a_l — записываемый символ; M_s — направление движения головки, обозначаемое одним из символов: $M_1 = L$ (влево), $M_2 = R$ (вправо), $M_3 = C$ (на месте).

Определение 3.97. Считаем, что для каждой пары

$$\{q_i, \alpha_j\}, \quad q_i \in Q, \quad \alpha_j \in A, \quad i \in 1 : n, j \in 0 : m$$

существует ровно одна команда вида (3.76). Множество этих команд называют *программой* машины. Таким образом, в программе всего $n(m + 1)$ команд.

Замечание 3.40

Из определения 3.96 следует, что у машины Тьюринга заключительное состояние q_0 может стоять только в правой части (3.76).

Определение 3.98. Совокупность внутреннего состояния, состояния ленты (т. е. размещения символов внешнего алфавита по ячейкам) и положения головки на ленте называют *конфигурацией*.

Работа машины заключается в изменении конфигураций согласно программе.

Состояние ленты для конфигурации с заключительным внутренним состоянием q_0 называют *результатом работы* машины Тьюринга.

Часто программу для машины Тьюринга записывают в виде таблицы команд, называемой *тьюринговой функциональной схемой*.

Пример 3.132. Рассмотрим машину Тьюринга с внешним алфавитом $\mathcal{A} = \{\alpha_0, \alpha_1, \alpha_2\}$ и множеством состояний $\mathcal{Q} = \{q_0, q_1, q_2, q_3, q_4\}$. Ее функциональная схема представлена в таблице 3.52. Проанализируем работу данной машины Тьюринга.

Таблица 3.52.

	α_0	α_1	α_2		α_0	α_1	α_2
q_1	$q_3\alpha_2L$	$q_2\alpha_1R$	$q_1\alpha_2L$	q_3	$q_0\alpha_0R$	$q_4\alpha_1R$	$q_1\alpha_2C$
q_2	$q_2\alpha_0C$	$q_1\alpha_2C$	$q_2\alpha_1C$	q_4	$q_3\alpha_1C$	$q_4\alpha_0R$	$q_4\alpha_2R$

Пусть начальная конфигурация имеет вид $\mathcal{K}_1 = \alpha_0 \alpha_2 \frac{\alpha_2}{q_1} \alpha_0$, т. е. управляющая головка обзревает символ α_2 , а машина находится в состоянии q_1 . Тогда

$$\alpha_0 \alpha_2 \frac{\alpha_2}{q_1} \alpha_0 \rightarrow \alpha_0 \frac{\alpha_2}{q_1} \alpha_2 \alpha_0 \rightarrow \frac{\alpha_0}{q_1} \alpha_2 \alpha_2 \alpha_0 \rightarrow \frac{\alpha_0}{q_3} \alpha_2 \alpha_2 \alpha_2 \alpha_0 \rightarrow \alpha_0 \frac{\alpha_2}{q_0} \alpha_2 \alpha_2 \alpha_0.$$

Так как при последней конфигурации машина находится в состоянии q_0 , то слово $\alpha_2 \alpha_2 \alpha_2$ является результатом.

Рассмотрим другую начальную конфигурацию: $\mathcal{K}_2 = \alpha_0 \alpha_1 \alpha_1 \alpha_2 \frac{\alpha_2}{q_1} \alpha_0$. Тогда

$$\begin{aligned} \alpha_0 \alpha_1 \alpha_1 \alpha_2 \frac{\alpha_2}{q_1} \alpha_0 &\rightarrow \alpha_0 \alpha_1 \alpha_1 \frac{\alpha_2}{q_1} \alpha_2 \alpha_0 \rightarrow \alpha_0 \alpha_1 \frac{\alpha_1}{q_1} \alpha_2 \alpha_2 \alpha_0 \rightarrow \\ &\rightarrow \alpha_0 \alpha_1 \alpha_1 \frac{\alpha_2}{q_2} \alpha_2 \alpha_0 \rightarrow \alpha_0 \alpha_1 \alpha_1 \frac{\alpha_1}{q_2} \alpha_2 \alpha_0 \rightarrow \alpha_0 \alpha_1 \alpha_1 \frac{\alpha_2}{q_1} \alpha_2 \alpha_0. \end{aligned}$$

Видно, что последняя конфигурация повторяет вторую, следовательно, машина никогда не придет в заключительное состояние q_0 .

Определение 3.99. Если машина Тьюринга, начиная работу в конфигурации \mathcal{K} , за конечное число шагов приходит в состояние q_0 (в конфигурации \mathcal{K}_0), то ее называют *применимой* к исходной конфигурации \mathcal{K} , в противном случае — *неприменимой*.

Конфигурацию \mathcal{K} называют *начальной*, а конфигурацию \mathcal{K}_0 — *заключительной*.

Пример 3.133. В примере 3.132 машина применима к конфигурации \mathcal{K}_1 и неприменима к конфигурации \mathcal{K}_2 .

Определение 3.100. *Активной зоной* конфигурации называют минимальную связную часть ленты, содержащую обозреваемую ячейку, а также все ячейки, в которых записаны непустые символы.

 **Замечание 3.41**

Будем считать, что машина Тьюринга всегда начинает и заканчивает работу в начале активной зоны. Такие начальную и конечную конфигурации называют стандартными.


Определение 3.101. Две машины Тьюринга эквивалентны если они применимы к одним и тем же начальным конфигурациям и результаты применения обеих машин совпадают.

Теорема 3.42. Для любой машины Тьюринга \mathcal{T} можно построить эквивалентную ей машину \mathcal{T}_S , у которой все заключительные конфигурации находятся в стандартной форме.

Доказательство. Машину \mathcal{T}_S построим, добавив к машине \mathcal{T} два новых состояния q' и q'' и команды:

$$q_0\alpha_i \rightarrow q'\alpha_iL, \quad i \in 0 : m, \quad q'\alpha_i \rightarrow q'\alpha_iL, \quad i \in 1 : m, \quad q'\alpha_0 \rightarrow q''\alpha_0R.$$

При этом состояние q'' будем считать заключительным для машины \mathcal{T}_S . Очевидно, что построенная машина \mathcal{T}_S эквивалентна машине \mathcal{T} . ■

 **Замечание 3.42**

Условимся далее по умолчанию рассматривать машины Тьюринга с внешним алфавитом $\mathcal{A} = \{\alpha_0, 1\}$, в который при необходимости будем включать символ-разделитель $*$.

Произвольное число $x \in N_0 = N \cup \{0\}$ будем представлять на ленте в виде $x + 1$ единицы

$$1^{x+1} = \overbrace{1 \ 1 \ \dots \ 1}^{x+1}, \quad (3.77)$$

чтобы запись нуля была непустой. Представление числа в виде (3.77) называют *унарным*.

Определение 3.102. Говорят, что машина \mathcal{T} правильно вычисляет функцию $f(x_1, \dots, x_n) : N_0^n \rightarrow N_0$, если начальную конфигурацию

$$\mathcal{K}_{start} = 1^{x_1+1} * \dots * 1^{x_n+1}$$

она переводит в заключительную конфигурацию

$$\mathcal{K}_{finish} = 1^{f(x_1, \dots, x_n)+1}$$

при условии, что значение функции $f(x_1, \dots, x_n)$ определено, и машина \mathcal{T} неприменима к начальной конфигурации \mathcal{K}_{start} , если значение функции не определено.

Рассмотрим несколько примеров на построение машин Тьюринга (таблица 3.53а–с).

Таблица 3.53.

	α_0	1
q_1	$q_2 1L$	$q_1 1R$
q_2	$q_0 \alpha_0 R$	$q_2 1L$

(a)

	α_0	1
q_1		$q_2 \alpha_0 R$
q_2	$q_0 1C$	$q_2 \alpha_0 R$

(b)

	α_0	1	*
q_1		$q_2 \alpha_0 R$	
q_2	$q_3 \alpha_0 L$	$q_2 1R$	$q_2 1R$
q_3		$q_4 \alpha_0 L$	
q_4	$q_0 \alpha_0 R$	$q_4 1L$	

(c)

Пример 3.134.

1 Пусть \mathcal{T}_a — машина Тьюринга с программой в таблице 3.53а. Машина \mathcal{T}_a , начиная работать в конфигурации $\mathcal{K}_0 = q_1 1^{x+1}$, останавливается в конфигурации $\mathcal{K}_1 = q_0 1^{x+2}$, таким образом правильно вычисляет функцию $f(x) = x + 1$, $x \in N_0$.

2 Пусть машина \mathcal{T}_b имеет программу в таблице 3.53б. Обозначим начальную конфигурацию как $\mathcal{K}_0 = q_1 1^{x+1}$. Тогда заключительная конфигурация выглядит так: $\mathcal{K}_1 = q_0 1$, т. е. машина \mathcal{T}_b правильно вычисляет функцию $f(x) = 0$, $x \in N_0$.

3 Рассмотрим машину \mathcal{T}_c , заданную таблицей 3.53с. Тогда при начальной конфигурации $\mathcal{K}_0 = q_1 1^{x+1} * 1^{y+1}$ заключительная конфигурация имеет вид $\mathcal{K}_1 = q_0 1^{x+y+1}$. Следовательно, машина \mathcal{T}_c правильно вычисляет функцию $f(x) = x + y$, $x, y \in N_0$.

Машину Тьюринга можно представить с помощью ориентированного графа. Каждому состоянию $q_i \in Q$ ставится в соответствие узел графа, а каждой команде — помеченная дуга.

Пример 3.135. Машина Тьюринга \mathcal{T}_a из примера 3.134 с программой в таблице 3.53а будет представлена в виде ориентированного графа следующим образом (рисунок 3.31).

Если на очередном такте работы машины Тьюринга символ на ленте не изменяется, то в правой части команды его можно не писать, например α_0 на ребре $q_2 \rightarrow q_0$.

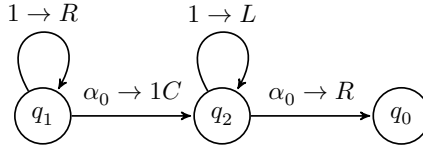


Рис. 3.31.

Замечание 3.43

Для простоты изложения конфигурацию в унарном алфавите $q_i 1^{\alpha+1}$ будем записывать как $q_i \alpha$.

В определении 3.96 для машины Тьюринга лента предполагалась бесконечной в обе стороны. Ограничим ленту с одной стороны и покажем, что машина Тьюринга с правой или левой полулентой эквивалентна машине Тьюринга с бесконечной лентой.

Теорема 3.43. Функция, правильно вычислимая на машине Тьюринга с обычной лентой, правильно вычислима и на машине Тьюринга с правой (левой) полулентой, т. е. для любой машины Тьюринга \mathcal{T} существует эквивалентная ей машина с правой (левой) полулентой \mathcal{T}_R (\mathcal{T}_L).

Доказательство. Пусть лента ограничена слева неподвижным маркером. Машина \mathcal{T}_R работает как обычная машина Тьюринга \mathcal{T} до тех пор, пока активная зона текущей конфигурации не выходит на неподвижный маркер. Тогда \mathcal{T}_R производит сдвиг активной зоны вправо на одну ячейку и снова работает как машина \mathcal{T} .

Такие же рассуждения можно провести для машины с левой полулентой. ■

3.5.2. Примеры и задачи на построение машин Тьюринга

Пример 3.136. Построим машину Тьюринга (как и в примере 3.134), правильно вычисляющую функцию $f(x) = x + 1$, но не для унарного алфавита, а в алфавите $\mathcal{A} = \{\alpha_0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Идея построения машины заключается в следующем. Нужно «передвинуть» управляющую головку на последнюю цифру числа. Если это цифра от 0 до 8, то заменить ее цифрой на 1 больше, вернуться в начало и остановиться. Если же это цифра 9, тогда заменить ее на 0 и сдвинуть управляющую головку к предыдущей цифре, после чего таким же способом увеличить на 1 эту предпоследнюю цифру.

Пусть на ленте только девятки (например, $9 \dots 9$). Тогда управляющая головка будет сдвигаться влево, заменяя девятки на нули, и в конце концов окажется на пустой ячейке. В эту пустую ячейку надо записать 1 и остановиться.

$$\begin{aligned} q_1 \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} &\rightarrow q_1 \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} R, \\ q_1 \alpha_0 &\rightarrow q_2 \alpha_0 L, \\ q_2 \{0, 1, 2, 3, 4, 5, 6, 7, 8\} &\rightarrow q_3 \{1, 2, 3, 4, 5, 6, 7, 8, 9\} L, \\ q_2 9 &\rightarrow q_2 0 L, \\ q_3 \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} &\rightarrow q_3 \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} L, \\ q_3 \alpha_0 &\rightarrow q_0 \alpha_0 R, \\ q_2 \alpha_0 &\rightarrow q_0 1 E. \end{aligned}$$

Пример 3.137. Построим машину Тьюринга, которую назовем «переводчиком». Алфавит этой машины $\mathcal{A} = \{\alpha_0, s_1, s_2, t_1, t_2, *\}$. Машина должна осуществлять *посимвольный перевод* цепочки s_{i_1}, \dots, s_{i_k} в цепочку t_{i_1}, \dots, t_{i_k} , сохраняя при этом *оригинальный текст*. То есть машина Тьюринга, начиная работать в конфигурации $\mathcal{K}_0 = q_1 s_{i_1}, \dots, s_{i_k} *$, должна закончить работу в конфигурации $\mathcal{K}_1 = q_1 s_{i_1}, \dots, s_{i_k} * t_{i_1}, \dots, t_{i_k}$ (рисунок 3.32).

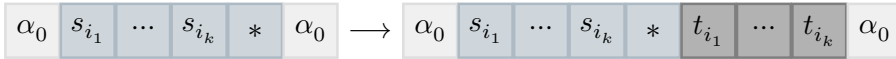


Рис. 3.32.

Как должна работать такая машина? Она должна запомнить обозреваемый символ s_i , то место, где он считан, найти первую пустую ячейку и записать туда символ t_i . Будем запоминать символ s_1 состоянием q_{s_1} , символ s_2 — состоянием q_{s_2} , а текущую копируемую ячейку — символом пустой ячейки α_0 . Имеем

$$q_1 s_i \rightarrow q_{s_i} \alpha_0 R, \quad i = 1, 2.$$

Далее, машина в состоянии q_{s_1} или q_{s_2} должна дойти, двигаясь вправо, до первой пустой ячейки и записать в нее символы t_1 или t_2 (перейдя соответственно в состояния q_{t_1} и q_{t_2}). Это реализуется следующими командами:

$$\begin{aligned} q_{s_i} \{s_1, s_2, t_1, t_2, *\} &\rightarrow q_{s_i} \{s_1, s_2, t_1, t_2, *\} R, \\ q_{s_i} \alpha_0 &\rightarrow q_{t_i} t_i L, \quad i = 1, 2. \end{aligned}$$

Теперь нам нужно вернуться в текущую копируемую ячейку и символ α_0 заменить на s_1 или s_2 . На этом один *цикл* работы машины закончится.

Добавим следующие команды:

$$\begin{aligned} q_{t_i} \{s_1, s_2, t_1, t_2, *\} &\rightarrow q_{t_i} \{s_1, s_2, t_1, t_2, *\} L, \\ q_{t_i} \alpha_0 &\rightarrow q_1 s_i R, \quad i = 1, 2. \end{aligned}$$

Возникает вопрос: когда машина должна закончить свою работу — тогда, когда после завершения очередного цикла встретит символ-разделитель *. Добавляем последнюю команду:

$$q_1 * \rightarrow q_0 * C.$$

Замечание 3.44

Как и в задаче 3.136, легко добавить команды, чтобы машина завершила работу в стандартной конечной конфигурации (в самой левой непустой ячейке ленты).

Пример 3.138. Построим машину Тьюринга, которая обращает непустое входное слово алфавите $\{a, b, \alpha_0\}$, где α_0 — пустой символ. Более точно, машина должна начальную конфигурацию $\mathcal{K}_S = q_1 W$ переводить в конфигурацию $\mathcal{K}_F = q_0 W^r$, где W — непустое слово в алфавите $\{a, b\}$, а W^r — слово W , записанное в обратном порядке. Назовем такую машину «конвертером».

Основная идея работы машины состоит в том, что обращение слова W надо начинать с конца этого слова. Машина будет считывать очередную букву $x \in \{a, b\}$, ставить на ее месте временный маркер x_1 , двигаться вправо и в первую пустую ячейку записывать x .

В состоянии q_1 машина находит первый с конца слова W еще «не обращенный» символ $x \in \{a, b\}$, заменяет его временным маркером $x_1 \in \{a_1, b_1\}$ и переходит в состояние q_a или q_b в зависимости от считанного символа.

$$q_1 \{a, b\} \rightarrow q_1 \{a, b\} R, \quad q_1 \alpha_0 \rightarrow q_2 * L, \quad q_2 a \rightarrow q_a a_1 R, \quad q_2 b \rightarrow q_b b_1 R.$$

В состоянии q_a или q_b она находит первую пустую ячейку, записывает в нее соответственно символ a или b и переходит в состояние q_3 .

$$\begin{aligned} q_a \{a, b, a_1, b_1, *\} &\rightarrow q_a \{a, b, a_1, b_1, *\} R, \\ q_b \{a, b, a_1, b_1, *\} &\rightarrow q_b \{a, b, a_1, b_1, *\} R, \\ q_a \alpha_0 &\rightarrow q_3 a L, \quad q_b \alpha_0 \rightarrow q_3 b L. \end{aligned}$$

В этом состоянии машина идет влево по входной ленте и при переходе через символ * приходит в состояние q_2 , и цикл повторяется.

$$q_3 \{a, b\} \rightarrow q_3 \{a, b\} L, \quad q_3 * \rightarrow q_2 * L.$$

Если в состоянии q_2 она читает пустой символ α_0 , то это означает, что обращение слова W завершено. Тогда машина переходит в состояние q_4 , удаляет временные маркеры $\{a_1, b_1\}$ и останавливается в состоянии q_0 .

$$q_2\alpha_0 \rightarrow q_4\alpha_0R, q_4\{a_1, b_1\} \rightarrow q_4\alpha_0R, q_4^* \rightarrow q_0\alpha_0R.$$

Диаграмма данной машины Тьюринга приведена на рисунке 3.33.

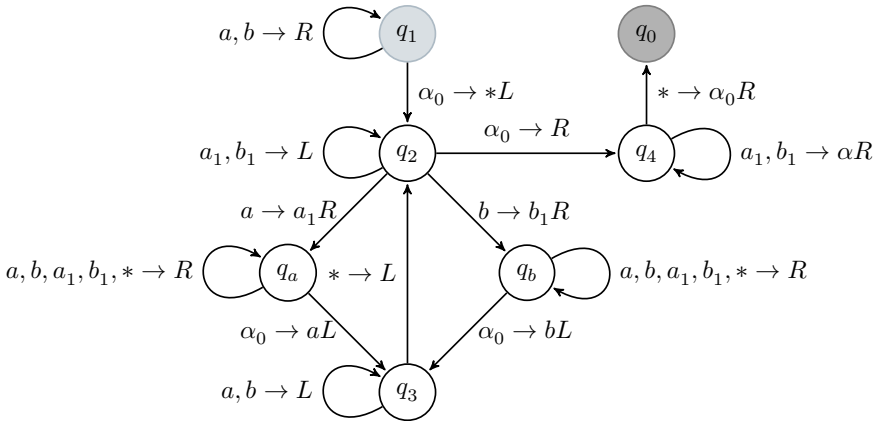


Рис. 3.33.

Задача 3.29. Постройте машины Тьюринга, правильно вычисляющие следующие функции:

1) $f(x) = x + k, x, k \in N_0;$

2) $f(x) = \text{sign}(x) = \begin{cases} 0 & \text{при } x = 0, \\ 1 & \text{при } x > 0, \end{cases} \quad x \in N_0;$

3) $f(x) = \overline{\text{sign}(x)} = \begin{cases} 1 & \text{при } x = 0, \\ 0 & \text{при } x > 0, \end{cases} \quad x \in N_0;$

4) Функцию выбора аргумента $f(x_1, x_2, x_3) = x_2, \quad x_1, x_2, x_3 \in N_0;$

5) $f(x, y) = xy, x, y \in N_0.$

Задача 3.30. Используя машину Тьюринга из примера 3.137, постройте машину, правильно вычисляющую функцию $f(x) = 2x$.

3.5.3. Операции над машинами Тьюринга

Прямое построение машин Тьюринга для решения даже простых задач может оказаться затруднительным. Однако можно облегчить работу, если использовать сочетания программ нескольких машин в результирующую программу.

Напомним, что согласно определению 2.20 словом, или цепочкой, в алфавите \mathcal{A} называют последовательность символов из этого алфавита. Множество слов алфавита \mathcal{A} обозначают \mathcal{A}^* .

• **Композиция машин Тьюринга.** Пусть даны две машины Тьюринга \mathcal{T}_1 и \mathcal{T}_2 , вычисляющие соответственно функции $f_1(\alpha)$ и $f_2(\alpha)$, заданные на множестве \mathcal{A}^* . Тогда существует машина Тьюринга \mathcal{T} , которая вычисляет функцию суперпозиции $f(\alpha) = f_2(f_1(\alpha))$. При этом для любого слова α функция $f(\alpha)$ определена в том и только в том случае, когда $f_1(\alpha)$ определена и $f_2(f_1(\alpha))$ также определена.

Состояния машин \mathcal{T}_1 и \mathcal{T}_2 обозначим дополнительно верхними индексами с номерами машин. Не умаляя общности, считаем, что множества внутренних состояний машин не пересекаются. Программа машины \mathcal{T} строится следующим образом. Полагаем:

$$q_1 = q_1^{\mathcal{T}_1}, \quad q_0 = q_0^{\mathcal{T}_2}, \quad q_0^{\mathcal{T}_1} = q_1^{\mathcal{T}_2}. \quad (3.78)$$

Модифицированные с учетом (3.78) команды для обеих машин \mathcal{T}_1 и \mathcal{T}_2 объединяем в одну программу.

Рассмотрим начальную конфигурацию $\mathcal{K}_1 = q_1\alpha$. В силу (3.78) машина \mathcal{T} начинает работать как машина \mathcal{T}_1 , и если последняя применима к \mathcal{K}_1 , то на некотором шаге будет получена конфигурация

$$\mathcal{K}_2 = q_0^{\mathcal{T}_1} f_1(\alpha) = q_1^{\mathcal{T}_2} f_1(\alpha),$$

являющаяся стартовой для \mathcal{T}_2 . Теперь машина \mathcal{T} действует как машина \mathcal{T}_2 .

Если \mathcal{T}_2 применима к конфигурации \mathcal{K}_2 , то за конечное число шагов будет получена конфигурация $\mathcal{K}_3 = q_0^{\mathcal{T}_2} f_2(f_1(\alpha))$, которая согласно (3.78) является заключительной для машины \mathcal{T} .

Если машина \mathcal{T}_1 неприменима к конфигурации \mathcal{K}_1 или машина \mathcal{T}_2 неприменима к конфигурации \mathcal{K}_2 , то машина \mathcal{T} неприменима к конфигурации \mathcal{K}_1 .

Рассмотренную машину Тьюринга \mathcal{T} называют *композицией* или *суперпозицией* машин \mathcal{T}_1 и \mathcal{T}_2 и обозначают как $\mathcal{T} = \mathcal{T}_2 \circ \mathcal{T}_1$ или $\mathcal{T} = \mathcal{T}_2(\mathcal{T}_1)$.

Схематически работу машины \mathcal{T} можно изобразить следующим образом:

$$q_1\alpha = q_1^{\mathcal{T}_1}\alpha \xrightarrow{\mathcal{T}_1} q_0^{\mathcal{T}_1}f_1(\alpha) = q_1^{\mathcal{T}_2}f_1(\alpha) \xrightarrow{\mathcal{T}_2} q_0^{\mathcal{T}_2}f_2(f_1(\alpha)) = q_0f_2(f_1(\alpha)).$$

• **Разветвление машин Тьюринга.** Пусть машины Тьюринга \mathcal{T}_1 и \mathcal{T}_2 вычисляют функции $f_1(\alpha)$ и $f_2(\alpha)$ соответственно. Тогда существует машина Тьюринга \mathcal{T} , которая начальную конфигурацию $\mathcal{K}_1 = q_1\lambda * \alpha$, где $\lambda \in \{0, 1\}$, переводит в заключительную $\mathcal{K}_0 = q_0f_1(\alpha)$, если $\lambda = 0$, и в $\mathcal{K}_0 = q_0f_2(\alpha)$, если $\lambda = 1$. Машину \mathcal{T} называют *разветвлением* машин и обозначают $\mathcal{T}_1 \vee \mathcal{T}_2$.

Как и ранее, состояния машин \mathcal{T}_1 и \mathcal{T}_2 будем обозначать верхними индексами, с номерами машин. Полагаем также, что множества внутренних состояний машин не пересекаются. Программа машины \mathcal{T} строится следующим образом. Объединим программы машин \mathcal{T}_1 и \mathcal{T}_2 , добавим новые начальное и заключительное состояния q_1, q_0 и следующие команды:

$$\begin{aligned} q_10 &\rightarrow q_1^{\mathcal{T}_1}\alpha_0R, & q_1^{\mathcal{T}_1}* &\rightarrow q_1^{\mathcal{T}_1}\alpha_0R, & q_11 &\rightarrow q_1^{\mathcal{T}_2}\alpha_0R, & q_1^{\mathcal{T}_2}* &\rightarrow q_1^{\mathcal{T}_2}\alpha_0R, \\ q_0 &= q_0^{\mathcal{T}_1} = q_0^{\mathcal{T}_2}. \end{aligned} \quad (3.79)$$

Если $\mathcal{K}_1 = q_1\lambda * \alpha$ — начальная конфигурация, то \mathcal{T} согласно (3.79) через два шага перейдет в конфигурацию $\mathcal{K}_2 = q_1^{\mathcal{T}_1}\alpha$, если $\lambda = 0$, и в конфигурацию $\mathcal{K}_2 = q_1^{\mathcal{T}_2}\alpha$ при $\lambda = 1$, а затем будет работать как \mathcal{T}_1 или \mathcal{T}_2 соответственно. Схематично работу машины можно представить так:

$$q_1\lambda * \alpha \xrightarrow{\mathcal{T}_1 \vee \mathcal{T}_2} \begin{cases} q_10 * \alpha \rightarrow q_1^{\mathcal{T}_1} * \alpha \rightarrow q_1^{\mathcal{T}_1}\alpha \rightarrow q_0^{\mathcal{T}_1}f_1(\alpha) = q_0f_1(\alpha) & \text{при } \lambda = 0, \\ q_11 * \alpha \rightarrow q_1^{\mathcal{T}_2} * \alpha \rightarrow q_1^{\mathcal{T}_2}\alpha \rightarrow q_0^{\mathcal{T}_2}f_2(\alpha) = q_0f_2(\alpha) & \text{при } \lambda = 1. \end{cases}$$

• **Машина Тьюринга, реализующая цикл.** Представим работу некоторого цикла следующим образом. Пусть на множестве \mathcal{A}^* заданы функции $f_1(\alpha), f_2(\alpha)$ и предикат $P(\alpha)$. Для произвольного слова α проверяется условие $P(\alpha) = 1$. Если оно выполнено, то цикл завершает работу и результат его работы есть $f_1(\alpha)$. Если же $P(\alpha) = 0$, то полагаем $\alpha = f_2(\alpha)$ и вся описанная процедура повторяется.

Построим машину \mathcal{T} , реализующую данную процедуру. Обозначим машины Тьюринга для вычисления функций f_1, f_2 и предиката P соответственно $\mathcal{T}_1, \mathcal{T}_2$ и \mathcal{T}_P . Пусть \mathcal{T}_0 — машина, которая добавляет к текущей конфигурации справа разделитель и слово α . Машина \mathcal{T} строится в соответствии со схемой (рисунок 3.34).

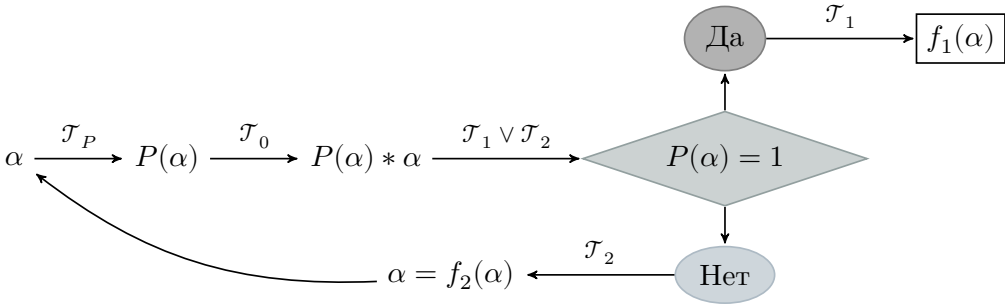


Рис. 3.34.

Положим:

$$q_0 = q_0^{\mathcal{T}_1}, \quad q_1 = q_1^{\mathcal{T}_P}, \quad q_1^{\mathcal{T}_0} = q_0^{\mathcal{T}_P}, \quad q_1^{\mathcal{T}_1 \vee \mathcal{T}_2} = q_0^{\mathcal{T}_0}.$$

Тогда если $\mathcal{T}_1 \vee \mathcal{T}_2$ работает как \mathcal{T}_1 , то полученное ею значение является выходом машины \mathcal{T} , если же $\mathcal{T}_1 \vee \mathcal{T}_2$ работает как \mathcal{T}_2 , то полученное ею значение снова подается на вход машины \mathcal{T} .

Упражнение 3.22. Используя цикл, постройте машину Тьюринга, реализующую перевод произвольного числа $n \geq 1$, заданного в унарной записи, т. е. в виде 1^{n+1} , в его двоичную запись, начинающуюся с 1.

Замечание 3.45

Таким образом, машины Тьюринга позволяют осуществлять последовательное выполнение программ (композиция), использовать условные переходы (разветвление), реализовывать циклы, т. е. содержат основные операторы программирования. Это дает основания предположить, что для любого алгоритма существует реализующая его машина Тьюринга. Данное предположение известно как **тезис Тьюринга**. Строго доказать его нельзя, поскольку для формулировки используется интуитивное понятие алгоритма. Однако принятие тезиса Тьюринга позволяет ставить знак равенства между утверждением о несуществовании машины Тьюринга для решения некоторой задачи и утверждением об отсутствии алгоритма ее решения вообще.

3.5.4. Проблема останова машины Тьюринга

Зададимся вопросом: существует ли алгоритм (машина Тьюринга), позволяющий по описанию произвольного алгоритма и его исходных данных (алгоритм и данные заданы символами на ленте машины Тьюринга) определить, останавливается ли этот алгоритм на этих данных или работает бесконечно. Данная задача получила название *проблемы останова*.

Введем необходимые определения.

Определение 3.103. Нумерация Геделя — это функция, сопоставляющая каждому объекту некоторого формального языка ее номер. При этом требуется, чтобы для любого натурального числа можно было определить, является ли оно номером или нет, и если является, то построить соответствующий ему объект языка.

Построение нумерации Геделя для объектов теории называется *арифметизацией теории*, оно позволяет переводить высказывания, аксиомы, теоремы, теории в натуральные числа.

Например, для логики предикатов с ее помощью можно явно пронумеровать объекты языка: переменные, предметные константы, функциональные символы, предикатные символы и формулы, построенные из них.

Историческая справка

Курт Фридрих Гедель (1906–1978) — австрийский логик и математик. Считается одним из наиболее выдающихся мыслителей XX в. В 1931 г. предложил идею кодирования произвольных объектов натуральными числами.

Предположим, что существуют общие внешний и внутренний алфавиты \mathcal{A}_0 и \mathcal{Q}_0 , которые включают символы для всех машин Тьюринга. Тогда для произвольной команды машины Тьюринга в виде (3.76) символы $q_i, \alpha_j, q_k, \alpha_l$ есть символы из \mathcal{A}_0 и \mathcal{Q}_0 . Укажем способ нумерации всех машин Тьюринга на основе геделевой нумерации.

Рассмотрим последовательность $\{p_i\}, i = 1, 2, \dots$, всех простых чисел в порядке возрастания, т. е. последовательность 2, 3, 5, 7, 11, 13, ...

Определение 3.104. Пусть машина Тьюринга \mathcal{T} определена набором команд вида (3.76):

$$q_{i_1} \alpha_{j_1} \rightarrow q_{k_1} \alpha_{l_1} M_{s_1}, q_{i_2} \alpha_{j_2} \rightarrow q_{k_2} \alpha_{l_2} M_{s_2}, \dots, q_{i_m} \alpha_{j_m} \rightarrow q_{k_m} \alpha_{l_m} M_{s_m}. \quad (3.80)$$

Ее номером будем называть число

$$n(\mathcal{T}) = \prod_{t=1}^m p_{5t-4}^{i_t} p_{5t-3}^{j_t} p_{5t-2}^{k_t} p_{5t-1}^{l_t} p_{5t}^{s_t}, \text{ где } i_t, k_t \in \mathcal{Q}_0, j_t, l_t \in \mathcal{A}_0, s_t \in \{1, 2, 3\}.$$

Замечание 3.46

Очевидно, что если $n(\mathcal{T})$ — номер машины Тьюринга, то по нему можно однозначно восстановить набор команд (3.80). Таким образом, есть взаимно однозначное соответствие между номером и программой машины Тьюринга.

Определение 3.105. Машину \mathcal{T} , применимую к слову $n(\mathcal{T})$, называют *самоприменимой*, в противном случае — *несамоприменимой*.

Пример 3.139. Очевидно, что машины \mathcal{T}_1 и \mathcal{T}_2 из примера 3.134 являются самоприменимыми.

Примером несоприменимой машины является машина, в правых частях команд которой не встречается заключительное состояние q_0 . Такая машина не применима ни к одной конфигурации.

Существует ли алгоритм, который по любой машине Тьюринга устанавливал бы, самоприменима она или нет? Эту задачу называют *проблемой самоприменимости*.

Согласно тезису Тьюринга (см. замечание 3.45) такой алгоритм следует искать в виде машины Тьюринга, т. е. требуется построить машину Тьюринга \mathcal{T}_A , которая была бы применима к номерам всех машин Тьюринга и в зависимости от самоприменимости машины с данным номером имела бы различные заключительные конфигурации. Пусть, например, в случае самоприменимой машины заключительная конфигурация имеет вид q_01 , а в случае несоприменимой машины — q_00 .

Теорема 3.44. Машины Тьюринга \mathcal{T}_A , решающей проблему самоприменимости, не существует.

Доказательство. Предположим противное. Используя \mathcal{T}_A , построим машину \mathcal{T}_S со следующими свойствами:

- (a) \mathcal{T}_S применима ко всем номерам несоприменимых машин;
- (b) \mathcal{T}_S неприменима ко всем номерам самоприменимых машин.

Машина \mathcal{T}_S получается из \mathcal{T}_A следующим образом: заключительное состояние $q_0^{\mathcal{T}_A}$ машины \mathcal{T}_A считается незаключительным состоянием машины \mathcal{T}_S , ее заключительное состояние обозначим $q_0^{\mathcal{T}_S}$. К программе машины \mathcal{T}_A добавляются две команды:

$$q_0^{\mathcal{T}_A}1 \rightarrow q_0^{\mathcal{T}_A}1E, \quad q_0^{\mathcal{T}_A}0 \rightarrow q_0^{\mathcal{T}_S}0E. \quad (3.81)$$

Очевидно, что в силу 3.81 машина \mathcal{T}_S удовлетворяет условиям (a) и (b).

Сама машина \mathcal{T}_S либо самоприменима, либо несоприменима. Если она самоприменима, то тем самым применима к номеру некоторой самоприменимой машины, но тогда нарушается требование (b). Если же машина \mathcal{T}_S несоприменима, тогда не выполняется требование (a).

Полученное противоречие доказывает теорему. ■

Используя теорему 3.44, докажем неразрешимость *проблемы останова*.

Будем считать, что описание произвольного алгоритма (машины Тьюринга) задается ее номером. Сформулируем задачу: существует ли машина Тьюринга \mathcal{T}_S , которая была бы применима ко всем словам вида $n(\mathcal{T}) * \alpha$, где $n(\mathcal{T})$ — номер произвольной машины, α — произвольное слово, и в случае, если машина \mathcal{T} применима к слову α , приходила бы к заключительной конфигурации q_01 , а в случае неприменимости — к конфигурации q_00 .

Теорема 3.45. Машины Тьюринга \mathcal{T}_S , решающей проблему останова, не существует.

Доказательство. Предположим, что машина \mathcal{T}_S существует. Рассмотрим композицию машин $\mathcal{T}_D = \mathcal{T}_S \circ \mathcal{T}_{copy}$, где \mathcal{T}_{copy} — машина, которая выполняет копирование аргумента на ленте из задачи 3.137. Запишем на ленте номер произвольной машины Тьюринга $n(\mathcal{T})$ и начнем работу машины \mathcal{T}_D с конфигурации $q_1^{J^D} n(\mathcal{T})$.

После того как машина \mathcal{T}_{copy} завершит работу, на ленте будет слово $n(\mathcal{T}) * n(\mathcal{T})$. По предположению машина \mathcal{T}_S применима ко всем таким словам, следовательно, машина \mathcal{T}_D остановится в конфигурации $q_0^{J^D} 1$, если машина \mathcal{T} применима к слову $n(\mathcal{T})$, и в конфигурации $q_0^{J^D} 0$, если неприменима. Таким образом, построенная машина \mathcal{T}_D решает проблему самоприменимости, что противоречит теореме 3.44. ■

3.5.5. Частично рекурсивные функции

Рассмотрим второй подход (исторически он был первым), предложенный А. Черчем для формализации понятия алгоритма, — класс частично рекурсивных функций. Функции данного класса строятся из базисных функций с помощью вычислимых операторов. Как и в предыдущем подразделе, будем рассматривать функции $f : N_0^n \rightarrow N_0$.

Историческая справка

Алонзо Черч (1903–1995) — американский математик и логик, внесший значительный вклад в основы информатики. Автор теории лямбда-исчислений, которая легла в основу функциональных языков программирования, в частности семейства **Lisp**.

Множество базисных функций таково:

- 1) *нуль-функция*: $0(x) = 0$;
- 2) *одноместная функция следования*: $s(x) = x + 1$;
- 3) *проектирующая функция*: $I_m^n(x_1, \dots, x_n) = x_m, 1 \leq m \leq n$.

Вычислимыми операторами являются операторы *суперпозиции, примитивной рекурсии и минимизации*.

- **Оператор суперпозиции.** Будем говорить, что функция

$$h(x_1, \dots, x_m) = g(f_1(x_1, \dots, x_m), \dots, f_n(x_1, \dots, x_m))$$

получается с помощью оператора суперпозиции из функций g, f_1, \dots, f_n , символически это записывается как $h = S(g, f_1, \dots, f_n)$.

Пример 3.140. Функция $f(x_1, x_2) = x_1 + 1$ может быть получена как $S(I_1^2(x_1, x_2))$.

- **Оператор примитивной рекурсии.** Функция $f(x_1, \dots, x_n, y)$ получается из функций $g(x_1, \dots, x_n)$ и $h(x_1, \dots, x_n, y, z)$ с помощью оператора примитивной рекурсии (символически это записывается как $f = R(g, h)$), если она может быть определена следующей схемой:

$$\begin{cases} f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n); \\ f(x_1, \dots, x_n, y + 1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)). \end{cases}$$

Пример 3.141. Функция $f(x_1, x_2) = x_1 + x_2$ задается схемой примитивной рекурсии

$$\begin{cases} x_1 + 0 = I_1^1(x_1); \\ x_1 + (x_2 + 1) = s(I_3^3(x_1, x_2, x_1 + x_2)). \end{cases}$$

Таким образом, $f = R(I_1^1, s(I_3^3))$.

- **Оператор минимизации.** Функция $f(x_1, \dots, x_n)$ получается из функции $g(x_1, \dots, x_n, y)$ с помощью оператора минимизации (символически записывается как $f = M_y g$)

$$f(x_1, \dots, x_n) = \mu y [g(x_1, \dots, x_n, y) = 0],$$

если значение $f(x_1, \dots, x_n)$ определено и равно y тогда и только тогда, когда значения $g(x_1, \dots, x_n, 0), \dots, g(x_1, \dots, x_n, y - 1)$ определены и отличны от 0, а $g(x_1, \dots, x_n, y) = 0$.

Оператор минимизации иногда называют μ -оператором.

Дадим теперь основное определение.

Определение 3.106. Функцию $f(x_1, \dots, x_n)$ называют *примитивно-рекурсивной*, если она может быть получена из базисных функций с помощью конечного числа применений операторов суперпозиции и примитивной рекурсии.

Функцию $f(x_1, \dots, x_n)$ называют *частично рекурсивной*, если она может быть получена из базисных функций с помощью конечного числа применений операторов суперпозиции, примитивной рекурсии и минимизации.

Функцию $f(x_1, \dots, x_n)$ называют *общерекурсивной*, если она частично рекурсивна и всюду определена.

Очевидно, что операторы S и R , применяемые ко всюду определенным функциям, дают всюду определенные функции. Отсюда следует, что всякая примитивно-рекурсивная функция общерекурсивна.

Однако μ -оператор может давать частичные функции даже при применении ко всюду определенным функциям.

Замечание 3.47

По аналогии с тезисом Тьюринга приведем **тезис Черча**: класс алгоритмически вычислимых функций совпадает с классом всех частично рекурсивных функций [66].

Принятие данного тезиса позволяет считать доказательство того, что некоторая функция не является частично рекурсивной, доказательством отсутствия алгоритма вычисления ее значений.

Пример 3.142. Установим примитивную рекурсивность некоторых числовых функций.

1. Функция $f(x) = m = s(s(\dots s(0(x)) \dots))$.

2. Функция $f(x_1, x_2) = x_1 x_2$.

В примере 3.141 было показано, что $h(x_1, x_2, x_3) = x_1 + x_3$ — частично рекурсивная функция. Тогда $f(x_1, x_2)$ задается схемой примитивной рекурсии:

$$\begin{cases} x_1 \cdot 0 = 0(x_1); \\ x_1(x_2 + 1) = h(x_1, x_2, x_1 x_2) = x_1 + x_1 x_2. \end{cases}$$

3. Функция $f(x_1, x_2) = x_1^{x_2}$.

Из пункта 2 следует, что $h(x_1, x_2, x_3) = x_1 x_3$ — частично рекурсивная функция. Тогда $f(x_1, x_2)$ задается схемой примитивной рекурсии:

$$\begin{cases} x_1^0 = s(0(x)); \\ x_1^{x_2+1} = h(x_1, x_2, x_1^{x_2}). \end{cases}$$

Упражнение 3.23. Установите частичную рекурсивность функции $f(x) = \text{sign}(x)$ из примера 3.29.

3.5.6. Нормальные алгорифмы Маркова

Рассмотрим еще одну алгоритмическую модель, предложенную А. А. Марковым [36] и называемую *нормальные алгорифмы*¹.

Данный подход представляет интуитивное понятие алгоритма как переработку слов в некотором конечном алфавите по определенным правилам. В качестве примитивных операций используются подстановки одного слова вместо другого. Множество таких подстановок определяет схему алгоритма.

Историческая справка

Андрей Андреевич Марков (1903–1979) — советский математик, основоположник советской школы конструктивной математики. Понятие **нормального алгоритма Маркова (НАМ)** было предложено им в 1947 г.

На основе НАМ в 1966 г. Валентином Федоровичем Турчиным был создан язык программирования **Рефал** [58] — один из старейших функциональных языков программирования, ориентированный на символьные вычисления.

Дадим необходимые определения. Напомним, что согласно определению 2.20 словом (цепочкой) в алфавите \mathcal{A} называют последовательность символов из этого алфавита. Пустое слово обозначают символом \wedge или ε . Множество слов алфавита \mathcal{A} обозначают \mathcal{A}^* .

Определение 3.107. Рассмотрим алфавит $\mathcal{A} = \{a_1, \dots, a_n\}$. Если слова $P, Q \in \mathcal{A}^*$, то выражения

$$P \rightarrow Q, P \rightarrow \cdot Q$$

называют *формулами подстановки* в алфавите \mathcal{A} . При этом формулу $P \rightarrow Q$ называют *простой*, а формулу $P \rightarrow \cdot Q$ — *заключительной*. В общем виде будем писать $P \rightarrow (\cdot) Q$.

Конечную *упорядоченную* последовательность таких формул называют *нормальным алгоритмом Маркова (НАМ)*, иногда говорят *схема нормального алгоритма Маркова*, обозначается как \mathcal{S}_N . Таким образом, схема \mathcal{S}_N имеет вид:

$$\mathcal{S}_N = \begin{cases} P_1 \rightarrow (\cdot) Q_1, \\ P_2 \rightarrow (\cdot) Q_2, \\ \dots\dots\dots \\ P_m \rightarrow (\cdot) Q_m. \end{cases} \quad (3.82)$$

Предполагается, что знаки \rightarrow и \cdot не входят в алфавит \mathcal{A} .

¹ При транскрипции греческих слов латинскими буквами звук [ф] передается сочетанием «th», так что русское прочтение слова *algorithm* как *алгорифм* имеет под собой некоторое основание.

Определение 3.108. Слово Q называют *подсловом* слова R , если справедливо представление

$$R = V_1 Q V_2, \quad Q, R, V_1, V_2 \in \mathcal{A}^*.$$

Определение 3.109. Рассмотрим формулу подстановки

$$P_i \rightarrow (\cdot) Q_i \in \mathcal{S}_N.$$

Пусть слово $R \in \mathcal{A}^*$ содержит подслово P_i (таких подслов в слове R может быть несколько).

Подстановкой называют замену первого по порядку подслова P_i исходного слова R на слово Q_i . В этом случае будем говорить, что формула подстановки *действует на слово R* .

Говорят, что схема \mathcal{S}_N *действует на слово R* , если существует формула подстановки, действующая на это слово.

Замечание 3.48

В формулах подстановок $P \rightarrow (\cdot) Q$ слова P и (или) Q могут быть пустыми. Если слово Q пустое, то результатом действия такой подстановки на слово $R = V_1 Q V_2$ будет слово $V_1 V_2$. Если пустым является слово P , то считается, что результатом подстановки является слово QR , т. е. правая часть формулы подстановки приписывается слева к слову R .

Действие схемы \mathcal{S}_N применительно к слову R может быть описано следующим образом.

Если в схеме \mathcal{S}_N имеются формулы подстановок, действующие на слово R , то первая из них применяется к R , в результате чего оно переходит в другое слово R_1 . К нему вновь применяется схема подстановок и т. д. Процесс прекращается в двух случаях:

- 1) схема \mathcal{S}_N не действует на очередное слово R_n ;
- 2) при получении R_n была применена заключительная подстановка.

Если процесс не заканчивается за конечное число шагов, то говорят, что схема \mathcal{S}_N *неприменима к слову R* , обозначая это следующим образом: $\mathcal{S}_N : R - - - v$.

Если же для слова R процесс заканчивается некоторым результатом $\tilde{R} = R_n$, то говорят, что схема \mathcal{S}_N *применима к слову R* и обозначают этот факт так: $\mathcal{S}_N : R \vDash \tilde{R}$.

Определение 3.110. Словарную функцию $f : \mathcal{A}^* \rightarrow \mathcal{A}^*$ называют *вычислимой по Маркову*, если существует схема \mathcal{S}_N такая, что

$$f(P) = Q \iff \mathcal{S}_N : P \vDash Q, \quad \forall P, Q \in \mathcal{A}^*.$$

 **Замечание 3.49**

Аналогично тезису Тьюринга (замечание 3.45) и тезису Черча (замечание 3.47) А. А. Марков предложил свой **принцип нормализации**, который утверждает, что для любого алгоритма в произвольном конечном алфавите можно построить эквивалентный ему нормальный алгоритм в том же алфавите.

Принятие данного принципа позволяет принимать доказательство несуществования нормального алгоритма как утверждение о несуществовании алгоритма вообще.

Пример 3.143. Рассмотрим схему $\mathcal{S}_N : \Lambda \rightarrow \cdot \Lambda$ в произвольном алфавите \mathcal{A} . Данная схема вычисляет тождественную функцию $f(P) = P$ для любого слова P .

Если же взять схему $\mathcal{S}_N : \Lambda \rightarrow \Lambda$, то она вычисляет нигде не определенную функцию.

Пример 3.144. Пусть алфавит $\mathcal{A} = \{*, 1\}$. Рассмотрим схему \mathcal{S}_N :

$$\mathcal{S}_N = \begin{cases} *1 \rightarrow \Lambda, \\ \Lambda \rightarrow \cdot \Lambda. \end{cases} \quad (3.83)$$

Найдем результат работы схемы (3.83) для слова $R = 111 * 1111 * 11$. Имеем

$$\overbrace{111 * 1111 * 11}^{2+3+1} = 111 \underset{\Lambda}{*1} 111 * 11 \rightarrow 111111 \underset{\Lambda}{*1} 1 \rightarrow 1111111 \rightarrow \cdot \overbrace{1111111}^6,$$

т. е. схема (3.83) суммирует числа в унарной системе счисления.

Заметим, что вторая команда в схеме (3.83) является избыточной.

Пример 3.145. Рассмотрим алфавит $\mathcal{A} = \{\alpha_1, \alpha_2\}$ и схему \mathcal{S}_N :

$$\mathcal{S}_N = \begin{cases} \alpha_1 \rightarrow \cdot \Lambda, \\ \alpha_2 \rightarrow \alpha_2. \end{cases} \quad (3.84)$$

Схема (3.84) слово P переводит в слово Q , полученное из P путем вычеркивания первого вхождения буквы α_1 . Схема неприменима к словам, не содержащим вхождений буквы α_1 .

Пример 3.146. Рассмотрим алфавит $\mathcal{A} = \{\alpha, \beta, \gamma\}$. Построим НАМ, заменяющий подряд стоящие одинаковые буквы одной буквой. Схема \mathcal{S}_N имеет вид:

$$\mathcal{S}_N = \begin{cases} \alpha\alpha \rightarrow \alpha, \\ \beta\beta \rightarrow \beta, \\ \gamma\gamma \rightarrow \gamma. \end{cases} \quad (3.85)$$

Пример 3.147. Построим схему НАМ для обращения слов P в алфавите $\mathcal{A} = \{\alpha_1, \dots, \alpha_n\}$, т. е. слова из тех же букв, но записанных в обратном порядке (обозначаем как P^{-1}).

Добавим к алфавиту \mathcal{A} два служебных символа $\mathcal{B} = \mathcal{A} \cup \{\beta, \gamma\}$ и рассмотрим следующую схему \mathcal{S}_N :

$$\mathcal{S}_N = \begin{cases} 1) \beta\beta \rightarrow \gamma \\ 2) \gamma\alpha_i \rightarrow \alpha_i\gamma, \forall \alpha_i \in \mathcal{A} \\ 3) \gamma\beta \rightarrow \gamma \\ 4) \gamma \rightarrow \cdot \wedge \\ 5) \beta\alpha_i\alpha_j \rightarrow \alpha_j\beta\alpha_i, \forall \alpha_i, \alpha_j \in \mathcal{A} \\ 6) \wedge \rightarrow \beta \end{cases} \quad (3.86)$$

Пусть $P = \alpha_{j_0} \dots \alpha_{j_k} \in \mathcal{A}^*$. Над стрелкой \rightarrow будем обозначать номер примененной формулы схемы (3.86). Тогда

$$\begin{aligned} P &\xrightarrow{6} \beta P \xrightarrow{5} \alpha_{j_1} \beta \alpha_{j_0} \alpha_{j_2} \dots \alpha_{j_k} \xrightarrow{5} \alpha_{j_1} \alpha_{j_2} \beta \alpha_{j_0} \alpha_{j_3} \dots \alpha_{j_k} \xrightarrow{5} \dots \xrightarrow{5} \\ &\xrightarrow{5} \alpha_{j_1} \alpha_{j_2} \dots \alpha_{j_k} \beta \alpha_{j_0} \xrightarrow{6} \beta \alpha_{j_1} \alpha_{j_2} \dots \alpha_{j_k} \beta \alpha_{j_0} \xrightarrow{5} \dots \xrightarrow{5} \\ &\xrightarrow{5} \alpha_{j_2} \alpha_{j_3} \dots \alpha_{j_k} \beta \alpha_{j_1} \beta \alpha_{j_0}. \end{aligned}$$

Повторяя этот процесс, получим

$$\begin{aligned} \beta \alpha_{j_k} \beta \alpha_{j_{k-1}} \dots \beta \alpha_{j_1} \beta \alpha_{j_0} &\xrightarrow{6} \beta \beta \alpha_{j_k} \beta \alpha_{j_{k-1}} \dots \beta \alpha_{j_1} \beta \alpha_{j_0} \xrightarrow{4} \\ &\xrightarrow{4} \gamma \alpha_{j_k} \beta \alpha_{j_{k-1}} \dots \beta \alpha_{j_1} \beta \alpha_{j_0} \xrightarrow{2} \alpha_{j_k} \gamma \beta \alpha_{j_{k-1}} \dots \beta \alpha_{j_1} \beta \alpha_{j_0} \xrightarrow{3} \\ &\xrightarrow{3} \alpha_{j_k} \gamma \alpha_{j_{k-1}} \dots \beta \alpha_{j_1} \beta \alpha_{j_0} \xrightarrow{2,3,4} \alpha_{j_k} \alpha_{j_{k-1}} \dots \alpha_{j_1} \alpha_{j_0} = P^{-1}. \end{aligned}$$

Пример 3.148. Построим схему НАМ инвертирования булевых наборов, т. е. перевод строки $\{\alpha_1, \dots, \alpha_n\}$, где $\alpha_i \in \{0, 1\}$, $1 \leq i \leq n$, в двойственную строку $\{\bar{\alpha}_1, \dots, \bar{\alpha}_n\}$.

Добавим к алфавиту служебный символ $*$ и рассмотрим схему \mathcal{S}_N :

$$\mathcal{S}_N = \begin{cases} *0 \rightarrow 1* \\ *1 \rightarrow 0* & \text{индикатор позиции } * \text{ движется вдоль строки, меняя } \alpha_i \text{ на } \overline{\alpha_i} \\ * \rightarrow \cdot \wedge & \text{удаляем символ } * \text{ и заканчиваем работу алгоритма} \\ \wedge \rightarrow * & \text{ставим символ } * \text{ индикатора позиции в начало слова} \end{cases}$$

Применяя описанную схему НАМ для слова $P = \{1101\}$, получаем:

$$\{1101\} \rightarrow \{*1101\} \rightarrow \{0*101\} \rightarrow \dots \rightarrow \{0010*\} \rightarrow \{0010\}.$$

Пример 3.149. Построим алгоритм сортировки числовых строк в порядке возрастания компонент, для простоты рассмотрим алфавит

$$\mathcal{A} = \{0, 1, 2\}.$$

Рассмотрим схему \mathcal{S}_N :

$$\mathcal{S}_N = \begin{cases} 20 \rightarrow 02 \\ 10 \rightarrow 01 \\ 21 \rightarrow 12 \end{cases} \quad (3.87)$$

Применим схему (3.87) к слову $P = \{2110\}$, имеем:

$$\{2110\} \rightarrow \{2101\} \rightarrow \{2011\} \rightarrow \{0211\} \rightarrow \{0121\} \rightarrow \{0112\}.$$

3.5.7. Примеры и задачи на нормальные алгорифмы Маркова

Пример 3.150. Рассмотрим алфавит скобочных выражений $\mathcal{A} = \{(\,)\}$. Построить схему НАМ, определяющую *правильные* скобочные выражения: строки правильных скобочных выражений (и только они) должны преобразовываться в пустое слово.

Решение представляет собой схему из одной команды:

$$() \rightarrow \Lambda.$$

Пример 3.151. Пусть $\mathcal{A} = \{a, b\}$. Требуется приписать символ a к концу слова P .

Чтобы приписать символ a справа, надо сначала пометить конец слова. Для этого воспользуемся дополнительным знаком $*$, который сначала поместим в конец слова P , а затем заменим его на символ q .

Но как поместить * в конец слова? Сделаем это следующим образом: сначала * припишем слева к слову P , а затем *перегоним* ее через все буквы слова:

$$bbab \rightarrow *bbab \rightarrow b * bab \rightarrow bb * ab \rightarrow bba * b \rightarrow bbab*$$

Заметим, что *перегон* символа * через какой-то символ α — это замена пары $*\alpha$ на пару $\alpha*$, которая реализуется заменой $*\alpha \rightarrow \alpha*$. Таким образом получаем следующую схему \mathcal{S}_N :

$$\mathcal{S}_N = \begin{cases} *a \rightarrow a*, \\ *b \rightarrow b*, \\ * \rightarrow \cdot a, \\ \Lambda \rightarrow *. \end{cases}$$

Пример 3.152. Рассмотрим алфавит $\mathcal{A} = \{0, 1, 2, 3\}$. Пусть P — непустое слово. Считая его записью неотрицательного целого числа в четверичной системе счисления, требуется получить запись этого же числа, но в двоичной системе. Например: $0123 \rightarrow 00011011$.

Такая замена, казалось бы, реализуется следующей схемой \mathcal{S}_N :

$$\mathcal{S}_N = \begin{cases} 0 \rightarrow 00, \\ 1 \rightarrow 01, \\ 2 \rightarrow 10, \\ 3 \rightarrow 11. \end{cases} \quad (3.88)$$

Но схема (3.88) работает неправильно, в чем можно убедиться на входном слове 0123. Действительно,

$$0123 \rightarrow 00123 \rightarrow 000123 \rightarrow \dots$$

Ошибка здесь в том, что после замены четверичной цифры на пару двоичных цифр уже никак нельзя отличить двоичные цифры от четверичных, поэтому наш алгоритм начинает заменять и двоичные цифры. Значит, надо как-то отделить ту часть числа, в которой уже была произведена замена, от той части, где замены еще не было.

Для этого, как и в предыдущем примере, воспользуемся дополнительным знаком *. Будем помечать слева символом * ту четверичную цифру, которая сейчас должна быть заменена на пару соответствующих двоичных цифр, а после того как такая замена будет выполнена, дополнительный

символ нужно поместить перед следующей четверичной цифрой:

$$0123 \rightarrow 00 * 0123 \rightarrow 00 * 123 \rightarrow 0001 * 23 \rightarrow 000110 * 3 \rightarrow 00011011 * .$$

Как видно, слева от * всегда находится та часть числа, которая уже переведена в двоичный вид, а справа — часть, которую еще предстоит заменить. Поэтому никакой путаницы между четверичными и двоичными цифрами уже не будет.

Итак, * сначала нужно разместить слева от первой цифры четверичного числа, а затем он должен *перепрыгивать* через очередную четверичную цифру, оставляя слева от себя соответствующие ей двоичные цифры. В конце, когда справа от * уже не окажется никакой цифры, этот символ надо удалить и остановиться.

Получаем следующую схему \mathcal{S}_N :

$$\mathcal{S}_N = \left\{ \begin{array}{l} *0 \rightarrow 00*, \\ *1 \rightarrow 01*, \\ *2 \rightarrow 10*, \\ *3 \rightarrow 11*, \\ * \rightarrow \cdot A, \\ A \rightarrow *. \end{array} \right. \quad (3.89)$$

Проверим работу схемы (3.89) на входном слове 0123:

$$0123 \rightarrow *0123 \rightarrow 00 * 123 \rightarrow 0001 * 23 \rightarrow 000110 * 3 \rightarrow 00011011* \rightarrow 00011011.$$

Задача 3.31. Пусть алфавит $\mathcal{A} = \{\alpha, \beta, \gamma\}$. Построить НАМ:

- 1) заменяющий все пары $\gamma\beta$ на α ;
- 2) заменяющий любое входное слово на слово α ;
- 3) который за первым символом непустого слова вставляет символ γ ;
- 4) который удаляет из слова второй символ, если такой есть;
- 5) который удаляет из непустого слова его последний символ;
- 6) который оставляет в слове только первое вхождение символа α , если такое есть;
- 7) который применим ко всем словам в алфавите, кроме двух слов — $\alpha\beta\gamma$ и $\alpha\alpha\alpha\gamma$;

8) который из всех слов в алфавите применим только к двум словам — пустому слову и слову $abcсba$.

3.5.8. Вычислительная сложность алгоритма

Определение 3.111. Назовем *размером входа* общее количество двоичных разрядов (n), необходимых для представления входов с помощью выбранного способа кодирования.

Наихудшим временем работы алгоритма будем считать верхнюю границу времени работы для произвольного входа, выраженную в виде функции размера входа $T(n)$.

Часто невозможно определить точное время работы алгоритма. В такой ситуации приходится приближенно оценивать время работы и обычно удается получить лишь асимптотические оценки при безграничном увеличении размера задачи.

Далее будем рассматривать функции, определенные на множестве натуральных чисел и принимающие натуральные значения, начиная с некоторых значений аргумента.

Определение 3.112. Введем следующие обозначения (O — символика):

1) $f(n) = O(g(n))$ (*верхняя оценка*), если существует константа $\alpha > 0$ и $n_0 \in \mathbb{N}$ такие, что

$$0 \leq f(n) \leq \alpha g(n), \quad n \geq n_0;$$

2) $f(n) = \Omega(g(n))$ (*нижняя оценка*), если существует константа $\alpha > 0$ и $n_0 \in \mathbb{N}$ такие, что

$$0 \leq \alpha g(n) \leq f(n), \quad n \geq n_0;$$

3) $f(n) = \Theta(g(n))$ (*плотная оценка*), если существуют положительные константы α, β и $n_0 \in \mathbb{N}$ такие, что

$$\alpha g(n) \leq f(n) \leq \beta g(n), \quad n \geq n_0.$$

Замечание 3.50

Таким образом, $f(n) = O(g(n))$ означает, что $f(n)$ растет асимптотически не быстрее, чем $g(n)$, с точностью до мультипликативной константы, тогда как $f(n) = \Omega(g(n))$ означает, что $f(n)$ растет асимптотически по крайней мере так же быстро, как $g(n)$, с точностью до мультипликативной константы.

Определение 3.113. Будем говорить, что $T(n)$ алгоритма имеет *порядок роста* или *теоретическую сложность* $f(n)$, если $T(n) = O(f(n))$.

В порядке возрастания $T(n)$ выделяют следующие асимптотические классы сложности алгоритмов.

- **Константный** $T(n) = O(1)$. Любой алгоритм, всегда требующий независимо от размера данных одного и того же времени, имеет константную сложность. Если не считать эффективность в наилучшем случае, в этот класс попадает очень небольшое количество алгоритмов.

- **Логарифмический** $T(n) = O(\log(n))$. Эффективность алгоритма логарифмическая. Алгоритм начинает работать намного медленнее с увеличением размера входных данных (n). Такое время работы характерно для алгоритмов, в которых большая задача делится на маленькие, каждая из которых решается по отдельности.

- **Линейный** $T(n) = O(n)$. Эффективность алгоритма меняется линейно в зависимости от размера входных данных (обычно такое наблюдается, когда каждый элемент входных данных требуется обработать линейное число раз). К этому классу относят алгоритмы, выполняющие сканирование списка, состоящего из n элементов (например, алгоритм поиска методом последовательного перебора).

- **Линейно-логарифмический** $T(n) = O(n \log(n))$. Такая эффективность характерна для алгоритмов, которые делят большую проблему на маленькие, а затем, решив их, соединяют их решения. К этому классу относится большое количество алгоритмов декомпозиции, таких как алгоритмы сортировки, быстрого преобразования Фурье.

- **Квадратичный** $T(n) = O(n^2)$. Обычно подобная зависимость характеризует эффективность алгоритмов, содержащих два встроженных цикла (например, ряд операций, выполняемых над матрицами размера $n \times n$).

- **Кубический** $T(n) = O(n^3)$. Подобная зависимость характеризует эффективность алгоритмов, содержащих три встроженных цикла.

- **Экспоненциальный** $T(n) = O(2^n)$. Данная зависимость типична для алгоритмов, выполняющих обработку всех подмножеств некоторого n -элементного множества. Часто термин «экспоненциальный» используется в широком смысле и означает очень высокий порядок роста.

- **Факториальный** $T(n) = O(2!)$. Характеризует алгоритмы, выполняющие обработку всех перестановок некоторого n -элементного множества.

Замечание 3.51

Иногда квадратичный и кубический классы объединяют в полиномиальный класс:

$$T(n) = O(n^k), \quad k \geq 2.$$

Пример 3.153. Алгоритм 1.11 вычисления a^n имеет логарифмическую сложность $T(n) = O(\log_2(n))$.

Вычисление чисел Фибоначчи по рекуррентному соотношению (1.46) имеет линейную сложность алгоритма $T(n) = O(n)$.

Пример 3.154. Представим, что у нас есть алгоритм сложности 1.7^n для некоторой задачи, который за «разумное» время позволяет решать примеры этой задачи размера не более n_0 ¹.

- The «hardware» approach: возьмем в 10 раз более быстрый компьютер. Теперь мы можем решать примеры размера $n_0 + 4$. Действительно, $1.7^4 \cong 10$.

- The «brainware» approach: придумаем алгоритм сложности 1.3^n . Это позволит нам решать примеры размера $2n_0$. Действительно, $1.3^2 \cong 1.7$.

Таким образом, уменьшение основания экспоненты времени работы алгоритма увеличивает размер решаемых за данное время примеров **в константное число раз**, в то время как использование более быстрого компьютера способно увеличить размер лишь **на константу**.

3.5.9. Сложностные классы задач

Установление точных оценок сложности алгоритма, о которых речь шла в предыдущем разделе удается далеко не всегда. В работах Кобмена [8] и Эдмундса [9], опубликованных независимо друг от друга, получил развитие другой подход — сложностные классы задач.

- Класс P (Polynomial) — задачи с полиномиальной сложностью.

Определение 3.114. Задача называется *полиномиальной*, если существует константа k и алгоритм A , решающий задачу со сложностью $T_A(n) = O(n^k)$ (для большинства таких задач $k < 6$, это «практически разрешимые задачи»).

- Класс NP (Non-deterministic Polynomial) — *полиномиально проверяемые задачи*.

¹ Пример взят с сайта <http://www.csedays.ru/theory2010/program/kulikov/algorithm>.

Определение 3.115. Задача называется *полиномиальной проверяемой*, если ее решение может быть проверено с помощью алгоритма полиномиальной сложности.

- Класс *NPC* (*NP-complete*) — *NP-полные задачи*.

Определение 3.116. Задача называется *NP-полной*, если:

- 1) она принадлежит к классу *NP*;
- 2) к ней полиномиально сводятся все задачи из класса *NP*.

Полиномиальная сводимость одной задачи к другой означает существование алгоритма полиномиальной сложности, способного выполнить трансляцию описания исходной задачи на одном языке в эквивалентное ему описание на другом языке.

Пример 3.155. Значение класса *NP-полных проблем* состоит еще и в том, что ему принадлежат очень многие известные и важные в прикладном отношении задачи. Перечислим некоторые из них.

- *Гамильтонов цикл.* Существует ли в графе гамильтонов цикл?
- *Задача коммивояжера.* Задан нагруженный граф и некоторое натуральное число γ (иногда называемое бюджетом). Требуется ответить на вопрос: найдется ли в графе маршрут, проходящий через все вершины графа такой, что его длина не превышает γ ?
- *Задача пропозициональной выполнимости* (*Boolean satisfiability problem, SAT*). Определить, выполнима ли данная формула в конъюнктивной нормальной форме.

После введения в теорию алгоритмов понятий сложностных классов была поставлена основная проблема теории сложности: $P = NP$?

Словесная формулировка проблемы имеет вид: можно ли все задачи, решение которых проверяется с полиномиальной сложностью, решить за полиномиальное время? Очевидно, что любая задача, принадлежащая классу P , принадлежит и классу NP , так как она может быть полиномиально проверена — задача проверки решения может состоять просто в повторном решении задачи.

На сегодня отсутствуют теоретические доказательства, как совпадения этих классов ($P = NP$), так и их несовпадения. Однако тот факт, что ни для одной *NP-полной* задачи не найдено полиномиального алгоритма, косвенно подтверждает гипотезу строгого включения $P \subset NP$, $P \neq NP$. Математический институт Клэя включил эту проблему в список проблем тысячелетия.

Задачи и вопросы

1. Что означает, что машина Тьюринга *неприменима* к данной конфигурации?

2. Что означает, что машина Тьюринга *применима* к данной конфигурации?

3. Постройте машину Тьюринга не применимую ни к одной конфигурации.

4. Что означает, что машина Тьюринга правильно вычисляет функцию $f(x_1, \dots, x_n)$?

5. Что такое *гедделева нумерация* для машины Тьюринга?

6. Построить машину Тьюринга в алфавите $\mathcal{A} = \{a, b, c\}$, которая:

а) заменяет на a каждый второй символ конфигурации;

б) определяет, входит ли в конфигурацию символ a . Ответ: конфигурация из одного символа a (да, входит) или пустая конфигурация (нет).

7. Построить машину Тьюринга в алфавите $\mathcal{A} = \{0, 1\}$, которая:

а) считая конфигурацию записью двоичного числа, удаляет из нее незначащие нули, если такие есть;

б) осуществляет транспозицию:

$$q_1 1^{x+1} * 1^{y+1} \rightarrow q_0 1^{y+1} * 1^{x+1};$$

в) осуществляет циклический сдвиг:

$$q_1 1^{x_1+1} * \dots * 1^{x_n+1} \rightarrow q_0 1^{x_2+1} * \dots * 1^{x_n+1} * 1^{x_1+1}.$$

8. Построить машину Тьюринга, правильно вычисляющую:

а) проектирующую функцию $I_m^n(x_1, \dots, x_n) = x_m$, $1 \leq m \leq n$;

б) $f(x) = \text{sign}(x)$;

в) $f(x_1, x_2) = x_1 x_2$.

9. Докажите примитивную рекурсивность следующих функций:

а) $f(x, y) = \max(x, y)$;

б) $f(x, y) = \min(x, y)$;

в) $f(x) = x!$

10. Докажите, что если функция $f(x_1, \dots, x_n)$ частично рекурсивна, то функция

$$g(x_1, x_2, x_3, \dots, x_n) = f(x_2, x_1, x_3, \dots, x_n)$$

также частично рекурсивна.

11. Когда нормальный алгоритм Маркова заканчивает работу?

12. Постройте нормальный алгоритм Маркова:

а) для сложения двух чисел, записанных в унарной системе счисления;

б) для возведения числа в унарной записи в квадрат;

в) для вычисления функции $F(P) = P$ для любого слова P ;

г) неприменимый ни к одному входному слову.

13. Что такое P и NP классы сложности?

3.6. Формальные языки и грамматики

Впервые понятие грамматики было формализовано лингвистами при изучении естественных языков. Предполагалось, что это может помочь при их автоматической трансляции. На интуитивном уровне язык можно определить как некоторое множество предложений допустимой структуры. Правила, определяющие допустимость той или иной конструкции языка, составляют синтаксис языка. Эти рассуждения можно отнести и к искусственным языкам программирования. Именно применительно к ним были достигнуты наилучшие результаты в этом направлении.

Если бы все языки состояли из небольшого конечного числа предложений, то проблемы синтаксического анализа не существовало. Можно было бы просто перечислить все предложения данного языка. Но так как большинство языков содержат неограниченное (или достаточно большое) число правильно построенных предложений, то возникает проблема разработки формальных средств описания и анализа языка (формальных грамматик).

Для формального описания языка необходимо задать набор символов (алфавит) языка и правила, по которым из символов строятся их последовательности (предложения), принадлежащие данному языку.

Существуют два вида описания языков:

- 1) *порождающая грамматика*, которая дает алгоритм, порождающий все правильные предложения языка;
- 2) *распознающая грамматика*, которая предполагает наличие алгоритма, определяющего принадлежность любого предложения данному языку.

Основные идеи формальных грамматик были заложены в работах Н. Хомского.

Историческая справка

Ноам Хомский — американский лингвист и общественный деятель, родился в Филадельфии 7 декабря 1928 г. Его отец эмигрировал из России незадолго до начала Первой мировой войны. С 1945 г. Хомский изучал в Пенсильванском университете лингвистику, математику и философию. Там же в 1955 г. получил докторскую степень, однако большая часть исследований, положенных в основу его первой монографии [62], была выполнена в Гарвардском университете в 1951–1955 гг.

Хомский — создатель системы грамматического описания, известной как генеративная (порождающая) грамматика; соответствующее направление лингвистики часто называют генеративизмом. Уже несколько десятилетий он является самым знаменитым лингвистом мира; ему и его теории посвящено множество статей и монографий, а развитие лингвистики в последней трети XX в. иногда называют хомскианской революцией.

Работы Хомского оказали влияние не только на лингвистику, но и инициировали развитие новых идей в философии, психологии, музыке. Например, Леонард Бернштейн использовал теорию Хомского для анализа музыкального ряда. Хомский также входит в первую десятку самых цитируемых в научном мире авторов. За резкость и бескомпромиссность суждений его часто называют Че Геварой в лингвистике.

3.6.1. Основные определения порождающих грамматик

Напомним, что согласно определению 2.20 цепочкой в алфавите \mathcal{A} называют последовательность символов из этого алфавита. Сохраняя ранее введенные обозначения, пустое слово будем обозначать символом ε , а множество слов алфавита \mathcal{A} как \mathcal{A}^* .

Введем формальные определения.

Определение 3.117. *Длиной* цепочки называют число составляющих ее символов.

Например, если $\alpha = abcdabcd$, то длина α равна 8. Длину цепочки α будем обозначать $|\alpha|$. Длина пустой цепочки ε по определению равна 0.

Через $\mathcal{A}^+ \subset \mathcal{A}^*$ будем обозначать множество, содержащее все *непустые* цепочки конечной длины в алфавите \mathcal{A} .

Определение 3.118. *Порождающей грамматикой* G называют четверку объектов (V_T, V_N, P, S) , где:

- 1) V_T — алфавит (словарь) терминальных символов (терминалов или примитивов);
- 2) V_N — алфавит (словарь) нетерминальных символов (нетерминалов), предполагается, что $V_T \cap V_N = \emptyset$;
- 3) множество P — конечное подмножество декартова произведения

$$(V_T \cup V_N)^+ \times (V_T \cup V_N)^*.$$

Элементы (α, β) множества P называют правилами вывода (продукционными правилами или продукциями) и записывают в виде $\alpha \rightarrow \beta$;

- 4) S — начальный символ (цель или аксиома) грамматики, $S \in V_N$.

Для того чтобы различать терминальные и нетерминальные символы, принято обозначать терминальные символы строчными, а нетерминальные символы прописными буквами латинского алфавита.

Для записи n правил с одинаковыми левыми частями

$$\alpha \rightarrow \beta_1, \dots, \alpha \rightarrow \beta_n$$

часто используют сокращенную запись

$$\alpha \rightarrow \beta_1 | \dots | \beta_n.$$

Иногда при описании грамматики словари терминальных и нетерминальных символов не указывают. В таком случае обычно предполагается, что грамматика содержит только те терминальные и нетерминальные символы, которые встречаются в правилах вывода.

Определение 3.119. Цепочка $\beta \in (V_T \cup V_N)^*$ непосредственно выводима из цепочки $\alpha \in (V_T \cup V_N)^+$ в грамматике $G = (V_T, V_N, P, S)$ (обозначают $\alpha \rightarrow \beta$), если

$$\alpha = \xi_1 \gamma \xi_2, \quad \beta = \xi_1 \delta \xi_2, \quad \xi_1, \xi_2, \delta \in (V_T \cup V_N)^*, \quad \gamma \in (V_T \cup V_N)^+$$

и правило вывода $\gamma \rightarrow \delta$ содержится в P .

Пример 3.156. Рассмотрим грамматику $G_1 = (\{a, b\}, \{A, S\}, P_1, S)$, где множество P_1 состоит из правил

$$S \rightarrow aAb, \quad aA \rightarrow aaAb, \quad A \rightarrow \epsilon.$$

Цепочка $aaAbb$ непосредственно выводима из цепочки aAb .

Определение 3.120. Цепочка $\beta \in (V_T \cup V_N)^*$ выводима из цепочки $\alpha \in (V_T \cup V_N)^+$ в грамматике $G = (V_T, V_N, P, S)$ (обозначают $\alpha \Rightarrow \beta$), если существует последовательность цепочек $\gamma_0, \dots, \gamma_n$ такая, что

$$\alpha = \gamma_0 \rightarrow \dots \rightarrow \gamma_n = \beta.$$

Последовательность $\gamma_0, \dots, \gamma_n$ называют *выводом* длины n .

Пример 3.157. В грамматике G_1 из примера 3.156 $S \Rightarrow aaaAbbb$, так как существует вывод

$$S \rightarrow aAb \rightarrow aaAbb \rightarrow aaaAbbb.$$

При этом длина вывода равна 3.

Замечание 3.52

Бинарное отношение \Rightarrow является транзитивным замыканием отношения \rightarrow , определенного на множестве $(V_T \cup V_N)^*$.

Определение 3.121. *Языком*, порождаемым грамматикой G , называют множество $L(G) = \{\alpha \in V_T^* \mid S \Rightarrow \alpha\}$, т. е. $L(G)$ — это все цепочки в алфавите V_T , которые выводимы из начального символа S с помощью правил вывода P .

Упражнение 3.24. Покажите, что для грамматики G_1 из примера 3.156 $L(G_1) = \{a^n b^n \mid n > 0\}$.

Определение 3.122. Грамматики G_1 и G_2 называют *эквивалентными*, если $L(G_1) = L(G_2)$.

Пример 3.158. Рассмотрим грамматику $G_2 = (\{a, b\}, \{S\}, P_2, S)$, где множество P_2 состоит из правил $S \rightarrow aSb \mid ab$. Эта грамматика эквивалентна грамматике G_1 из примера 3.156, так как обе порождают один и тот же язык $L(G_1) = L(G_2) = \{a^n b^n \mid n > 0\}$.

3.6.2. Классификация грамматик

Накладывая различные ограничения на правила вывода, можно выделить классы грамматик. Рассмотрим классификацию, предложенную Н. Хомским [62].

- **Тип G_0 (свободные, или неограниченные грамматики).**

Граматику G называют грамматикой типа 0, если на ее правила вывода не накладывается никаких ограничений.

- **Тип G_1 (контекстно-зависимые, или НС-грамматики).**

Граматику $G = (V_T, V_N, P, S)$ называют грамматикой типа 1, если каждое правило из P имеет вид

$$\xi_1 A \xi_2 \rightarrow \xi_1 \gamma \xi_2, \quad A \in V_N, \gamma \in (V_T \cup V_N)^+, \xi_1, \xi_2 \in (V_T \cup V_N)^*.$$

Цепочку ξ_1 называют левым контекстом, цепочку ξ_2 называют правым контекстом.

Пример 3.159. Грамматика $G_1 = (\{a, b, c\}, \{S, B, C, W, Z\}, P, S)$, где множество P состоит из правил:

$$\begin{aligned} S &\rightarrow aSBC \mid aBC, & CB &\rightarrow CZ, & CZ &\rightarrow WZ, & WZ &\rightarrow WC, & WC &\rightarrow BC, \\ aB &\rightarrow ab, & bB &\rightarrow bb, & bC &\rightarrow bc, & cC &\rightarrow cc, \end{aligned}$$

является контекстно-зависимой грамматикой. Грамматика G_1 порождает язык $L(G_1) = \{a^n b^n c^n \mid n \geq 1\}$.

• **Тип G_2 (контекстно-свободные, или КС-грамматики).**

Грамматику $G = (V_T, V_N, P, S)$ называют грамматикой типа 2, если каждое правило из P имеет вид

$$A \rightarrow \beta, \quad A \in V_N, \quad \beta \in (V_T \cup V_N)^+.$$

Пример 3.160. Грамматика $G_2 = (\{a, b, c\}, \{S, A\}, P, S)$, где множество правил P имеет вид

$$S \rightarrow aAb \mid acb, \quad A \rightarrow cSc,$$

является контекстно-свободной грамматикой. Грамматика G_2 порождает язык $L(G_2) = \{(ac)^n(cb)^n \mid n > 0\}$.

Пример 3.161. Рассмотрим контекстно-свободную грамматику G_D , описывающую язык Дика (последовательность правильных расстановок скобок)

$$G_D = (\{a_1, b_1, \dots, a_n, b_n\}, \{S\}, P, S),$$

где множество правил P имеет вид:

$$S \rightarrow \epsilon, \quad S \rightarrow a_1 S b_1 S, \quad \dots, \quad S \rightarrow a_n S b_n S$$

Если интерпретировать символ a_i как открывающую скобку i -го типа, а b_i как закрывающую скобку i -го типа, то язык Дика описывает правильную расстановку скобок n типов. А именно:

- 1) для любой начальной последовательности число открывающих скобок i -го типа не меньше числа закрывающих скобок этого же типа;
- 2) для всей последовательности число вхождений открывающей скобки i -го типа равно числу вхождений закрывающей скобки i -го типа.

• **Тип G_3 (автоматные или регулярные грамматики).** Грамматики типа 3 делятся на две группы:

а) левосторонние (леворекурсивные), правила вывода для которых имеют вид: $A \rightarrow a|Ba$, где $A, B \in V_N$, $a \in V_T$;

б) правосторонние (праворекурсивные), правила вывода для которых имеют вид: $A \rightarrow a|aB$, где $A, B \in V_N$, $a \in V_T$.

Пример 3.162. Грамматика $G_3 = (\{a, b\}, \{S, A, B\}, P, S)$, где множество P содержит правила:

$$S \rightarrow aA \mid bB, \quad A \rightarrow aA \mid a, \quad B \rightarrow b,$$

является регулярной (праволинейной) грамматикой. Грамматика G_3 порождает язык $L(G_3) = \{a^n, b^2 \mid n > 1\}$.

Пример 3.163. Грамматика $G_4 = (\{a, b, c\}, \{S, A, B, C\}, P, S)$, где множество P содержит правила:

$$(1)S \rightarrow aA, (2)A \rightarrow aA \mid bC, (3)C \rightarrow cC \mid c, \quad (3.90)$$

является регулярной (праволинейной) грамматикой. Грамматика G_4 порождает язык $L(G_4) = \{a^n b c^m \mid n, m > 0\}$.

Рассмотрим вывод слова $aaabcc \in L(G_4)$ в грамматике G_4 . При применении правила сверху будем ставить его номер согласно (3.90).

$$S \xrightarrow{1} aA \xrightarrow{2} aaA \xrightarrow{2} aaaA \xrightarrow{2} aaabC \xrightarrow{3} aaab c C \xrightarrow{3} aaabcc.$$

Классы грамматик типа G_0, G_1, G_2 и G_3 образуют *иерархию Хомского*.

Определение 3.123. Язык называют *контекстным* (контекстно-свободным, регулярным), если он порождается некоторой контекстной (контекстно-свободной, регулярной) грамматикой. Контекстно-свободные языки называют также *алгебраическими* языками.

Замечание 3.53

Двигаясь от грамматики G_0 к грамматике G_3 , можно заметить, что, в то время как ограничения на правила вывода усиливаются, описательные возможности языков уменьшаются.

Справедливо следующее очевидное утверждение.

Теорема 3.46. $L(G_3) \subset L(G_2) \subset L(G_1) \subset L(G_0)$.

Упражнение 3.25. Постройте грамматику, порождающую язык булевых формул от трех переменных x, y, z .

3.6.3. Контекстно-свободные грамматики

Наиболее важный класс грамматик в иерархии Хомского представляют собой контекстно-свободные грамматики. Во-первых, на них основаны почти все используемые языки программирования, во-вторых, для КС-грамматик разработаны эффективные грамматические анализаторы.

Определение 3.124. Часто для контекстно-свободных грамматик правила вывода $A \rightarrow \beta_1 \mid \dots \mid \beta_n$ записывают в виде $A ::= \beta_1 \mid \dots \mid \beta_n$. Такое описание правил называют записью в *форме Бэкуса — Наура*.

Историческая справка

В 1960 году группой программистов из Цюриха был создан алгоритмический язык, впоследствии получивший название Algol-60. В обсуждении нового языка принял активное участие американский математик Джон Бэкус.

Однако здесь возникла проблема — английский язык, на котором изъяснялся Бэкус, был мало понятен швейцарским программистам. В связи с этим для описания конструкций языка были применены специальные диаграммы, которые Бэкус разработал совместно с датским астрономом и программистом Питером Науром.

С тех пор форма Бэкуса — Наура (Backus — Naur Form — BNF) стала своеобразным «эсперанто» мирового программирования.

Пример 3.164. Рассмотрим бесскобочные алгебраические выражения, т. е. множество формул вида: $a+b/c*d-a*b$. Попробуем это неформально заданное множество формул превратить в язык с контекстно-свободной грамматикой.

Положим $V_T = \{a, \dots, z\}$. Далее определим множество нетерминалов

$$V_N = \{ \langle \text{бесскобочное выражение} \rangle, \langle \text{буква} \rangle, \langle \text{знак операции} \rangle \}$$

(нетерминальному символу будем давать смысловое название, которое будет заключаться в угловые скобки). В качестве начального символа грамматики возьмем

$$S = \langle \text{бесскобочное выражение} \rangle .$$

Множество P состоит из трех правил:

- 1) $\langle \text{бесскобочное выражение} \rangle ::=$
 $\langle \text{бесскобочное выражение} \rangle \langle \text{знак операции} \rangle \langle \text{бесскобочное выражение} \rangle \mid \langle \text{буква} \rangle;$
- 2) $\langle \text{буква} \rangle ::= a \mid \dots \mid z;$
- 3) $\langle \text{знак операции} \rangle ::= + \mid - \mid * \mid /.$ (3.91)

Грамматик, описывающих данный язык, может быть много. Например, первое правило в (3.91) можно заменить на:

$$\begin{aligned} \langle \text{бесскобочное выражение} \rangle ::= & \\ \langle \text{бесскобочное выражение} \rangle \langle \text{знак операции} \rangle \langle \text{буква} \rangle \mid & \langle \text{буква} \rangle \end{aligned}$$

или

$$\begin{aligned} \langle \text{бесскобочное выражение} \rangle ::= & \\ \langle \text{буква} \rangle \langle \text{знак операции} \rangle \langle \text{бесскобочное выражение} \rangle \mid & \langle \text{буква} \rangle . \end{aligned}$$

Можно расширить множество нетерминальных символов пустым символом ε и получить такую грамматику:

$$\begin{aligned} \langle \text{бесскобочное выражение} \rangle &::= \langle \text{буква} \rangle \langle \text{окончание} \rangle, \\ \langle \text{окончание} \rangle &::= \langle \text{знак операции} \rangle \langle \text{бесскобочное выражение} \rangle \mid \varepsilon. \end{aligned} \quad (3.92)$$

Задача 3.32. Постройте несколько вариантов грамматик, описывающих полную скобочную запись арифметических выражений. Под полными скобочными записями понимаются записи вида $((a + b) * c) - ((a + d) / e)$, в которых каждая бинарная операция вместе с ее аргументами заключается в скобки.

Понятие грамматики как средства описания языка неотделимо от понятия грамматического (синтаксического) разбора.

Определение 3.125. Последовательность цепочек нетерминальных и терминальных символов β_1, \dots, β_n называют *грамматическим разбором* для цепочки терминальных символов α , если $\beta_1 = S, \beta_n = \alpha$, а β_i отличается от β_{i+1} тем, что один из нетерминальных символов цепочки β_i заменен по одному из правил грамматики.

В дальнейшем грамматический разбор будем записывать в столбик.

Пример 3.165. Проведем грамматический разбор выражения $a + b * d$ языка бесскобочных арифметических выражений, заданного грамматикой (3.91), из предыдущего примера 3.164:

$$\begin{aligned} &\langle \text{бесскобочное выражение} \rangle \\ &\langle \text{бесскобочное выражение} \rangle \langle \text{знак операции} \rangle \langle \text{бесскобочное выражение} \rangle \\ &\langle \text{буква} \rangle \langle \text{знак операции} \rangle \langle \text{бесскобочное выражение} \rangle \\ &a \langle \text{знак операции} \rangle \langle \text{бесскобочное выражение} \rangle \\ &a + \langle \text{бесскобочное выражение} \rangle \\ &a + \langle \text{бесскобочное выражение} \rangle \langle \text{знак операции} \rangle \langle \text{бесскобочное выражение} \rangle \\ &a + \langle \text{буква} \rangle \langle \text{знак операции} \rangle \langle \text{бесскобочное выражение} \rangle \\ &a + b \langle \text{знак операции} \rangle \langle \text{бесскобочное выражение} \rangle \\ &a + b * \langle \text{бесскобочное выражение} \rangle \\ &a + b * \langle \text{буква} \rangle \\ &a + b * d \end{aligned}$$

Замечание 3.54

Нетрудно видеть, что, как правило, существует несколько грамматических разборов одной и той же формулы.

3.6.4. Синтаксический анализатор

Представляет интерес построение программы — *синтаксического анализатора*, которая бы автоматически выясняла принадлежность цепочки терминальных символов (слова) данному языку. Однако, вследствие неоднозначности выбора расшифровки нетерминального символа, в общем случае грамматический разбор требует перебора вариантов.

Рассмотрим один важный частный случай, когда выбор цепочки при расшифровке нетерминального символа определяется однозначно. Назовем это свойство грамматики *однозначностью ветвления по первому символу*.

Формально это свойство можно описать следующим образом.

Определение 3.126. Грамматика имеет свойство однозначности ветвления по первому символу, если:

1) каждое правило грамматики $\alpha ::= \beta_1 \mid \dots \mid \beta_n$ должно удовлетворять условию $L(\beta_i) \cap L(\beta_j) = \emptyset$, где $L(\beta_i)$ обозначает множество терминальных символов, с которых может начинаться цепочка β_i (после применения к ней правил вывода);

2) каждое правило грамматики

$$\alpha ::= \dots \mid \beta\gamma \mid \dots, \text{ где } \beta ::= \xi_1 \mid \dots \mid \xi_n \mid \varepsilon$$

должно удовлетворять условию: $L(\gamma) \cap L(\xi_i) = \emptyset$.

Замечание 3.55

Второй случай запрещает скрытую неоднозначность, когда за счет использования пустого символа ее проверка «сдвигается» на другое правило.

Пример 3.166. В примере 3.164 для грамматики (3.91) неоднозначность имеет место, так как

$$L(\langle \text{бескбочное выражение} \rangle) = \{a, \dots, z\} = L(\langle \text{буква} \rangle)$$

и, очевидно, их пересечение не пусто.

Для грамматики (3.92) того же языка неоднозначности уже не будет (так как $\alpha = \langle \text{бескбочное выражение} \rangle$, $\beta = \langle \text{окончание} \rangle$, γ отсутствует).

Для грамматик, удовлетворяющих свойству однозначности ветвления по первому символу, существует алгоритм, который, получая на вход правила грамматики, на выходе порождает алгоритм синтаксического анализа слов языка, задаваемого этой грамматикой. Дадим неформальное описание этого алгоритма, демонстрируя изложение на примере.

Пример 3.167. Рассмотрим технологию построения синтаксического анализатора для языка арифметических выражений в полной скобочной записи (для краткости будем называть их формулами).

Грамматика, определяющая этот язык, устроена очень просто:

$$\begin{aligned} \langle \text{формула} \rangle &::= \langle \text{переменная} \rangle | (\langle \text{формула} \rangle \langle \text{знак операции} \rangle \langle \text{формула} \rangle) \\ \langle \text{переменная} \rangle &::= a | \dots | z, \quad \langle \text{знак операции} \rangle ::= + | - | * | /. \end{aligned} \quad (3.93)$$

Нетрудно видеть, что грамматика (3.93) удовлетворяет условию ветвления по первому символу.

Для построения синтаксического анализа сделаем преобразование грамматики из формы Бэкуса — Наура в форму графа. Такое представление грамматики называют *синтаксической диаграммой*.

Синтаксическая диаграмма строится по следующим правилам (аналогично соединению контактов в разделе «Контактные схемы»).

① *Правило 1.* Каждый терминальный символ t изображается кружочком (вершиной графа), помеченным самим символом t (рисунок 3.35а).

② *Правило 2.* Каждый нетерминальный символ N изображается прямоугольником (вершиной графа), помеченным тем же символом N (рисунок 3.35b).

③ *Правило 3.* Последовательность терминальных и нетерминальных символов изображается соединением соответствующих вершин дугами в порядке их следования. Например, изображение последовательности $N_1 N_2$ синтаксической диаграммой представлено на рисунке 3.35с.

④ *Правило 4.* Значку $|$ сопоставляется ветвление. Например, $N_1 | N_2$ на синтаксической диаграмме изображено на рисунке 3.35d.

⑤ *Правило 5.* Пустой цепочке символов ставится в соответствие дуга графа.

Синтаксическая диаграмма для языка полных арифметических скобочных записей (3.93), построенная на основе этих правил, показана на рисунке 3.36.

Далее на основе следующих правил по этой диаграмме можно написать программу синтаксического анализа. Заметим, что для нашей грамматики

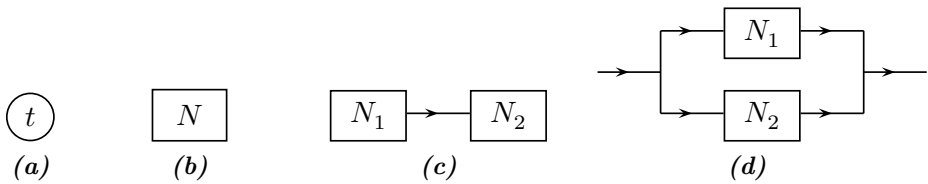


Рис. 3.35.

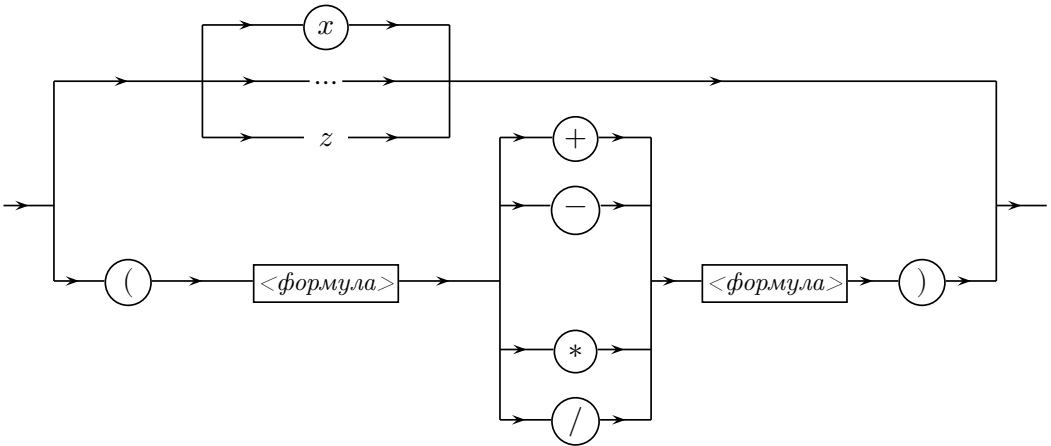


Рис. 3.36.

условие однозначности ветвления выполнено, так как единственное ветвление осуществляется на основе проверки, является ли очередной символ выражения открывающей скобкой или латинской буквой.

1 *Правило 1.* Терминальному символу t синтаксической диаграммы в программе синтаксического анализа сопоставляется элементарный оператор распознавания:

```

if  $ch = t$  then
    READ( $ch$ )
else
    return ERROR()
end if

```

Здесь переменная ch обозначает текущий символ записи выражения, процедура READ(ch) считывает следующий символ записи, процедура ERROR() обрабатывает ошибку (в простейшем варианте просто прерывает дальнейший анализ выражения).

2 *Правило 2.* Нетерминальному символу N синтаксической диаграммы сопоставляется процедура RECOGNITION(N) распознавания этого

символа (при выполнении программы анализа вызывается соответствующая процедура).

3 *Правило 3.* Последовательности терминальных или нетерминальных символов соответствует последовательный вызов соответствующих символам операторов (правило 1) и процедур распознавания (правило 2).

4 *Правило 4.* Ветвлению соответствует оператор условного перехода. Так, диаграмме с ветвлением $N_1 \mid N_2$ на рисунке 3.35d соответствует такая процедура распознавания:

```

if  $ch \in L(N_1)$  then
    RECOGNITION( $N_1$ )
else
    if  $ch \in L(N_2)$  then
        RECOGNITION( $N_1$ )
    else
        return ERROR()
    end if
end if

```

Здесь $L(N_1)$ обозначает множество начальных символов, с которых могут начинаться правила вывода для нетерминального символа N_1 (см. определение 3.126).

Заметим, что множества начальных символов N_1 и N_2 не имеют общих элементов в силу однозначности ветвления. Если это условие не выполняется, то строить синтаксический анализатор по предложенным правилам нельзя.

Упражнение 3.26. Используя перечисленные правила, написать программу синтаксического анализа языка полных скобочных записей арифметических выражений (см. рисунок 3.36).

3.6.5. Преобразования контекстно-свободных грамматик

Определение 3.127. Нетерминальный символ A называется *непродуктивным* (*непроизводящим*), если он не порождает ни одной терминальной цепочки, т. е. не существует вывода $A \Rightarrow \alpha \mid \alpha \in V_T^*$.

Представляет интерес задача удаления из КС-грамматики всех непродуктивных нетерминальных символов. Рассмотрим алгоритм построения КС-грамматики, эквивалентной исходной, все нетерминальные символы которой продуктивны.

Алгоритм 3.18. Удаление непродуктивных нетерминалов**Входные данные:**Грамматика $G = (V_T, V_N, P, S)$ **Выход:** \tilde{V}_N — словарь продуктивных нетерминальных символов

// Инициализация

 $W_0 \leftarrow \emptyset$ $W_1 \leftarrow \{A \mid A \rightarrow \alpha \in P, \alpha \in V_T^*\}$ $k \leftarrow 1$

// Основной цикл

while $W_k \neq W_{k-1}$ **do** $W_{k+1} \leftarrow W_k \cup \{B \mid B \rightarrow \beta \in P, \beta \in (V_T \cup W_k)^*\}$ $k \leftarrow k + 1$ **end while** $\tilde{V}_N \leftarrow W_k$

Пример 3.168. Пусть задана грамматика G со следующими правилами вывода:

$$S \rightarrow SA \mid BSb \mid bAb, \quad A \rightarrow aSa \mid bb, \quad B \rightarrow bBb \mid BaA.$$

Применим алгоритм 3.18 для построения множества продуктивных нетерминалов. Имеем:

- 1) $W_1 = \{A\}$, т. к. в множестве P есть правило $A \rightarrow bb$;
- 2) $W_2 = \{A, S\}$, т. к. имеется правило $S \rightarrow bAb$ и $A \in W_1$;
- 3) $W_3 = W_2$.

Все символы множества $V_N \setminus W_2$ являются непродуктивными, т. е. не используются в выводе никакой терминальной цепочки, и их можно удалить из грамматики вместе со всеми правилами, в которые они входят. Грамматика G_1 , эквивалентная исходной грамматике, будет включать следующее множество правил:

$$S \rightarrow SA \mid bAb, \quad A \rightarrow aSa \mid bb.$$

В грамматике G_1 все нетерминалы продуктивны.

Существует еще один тип нетерминальных символов, которые можно удалять из грамматики вместе с правилами, в которые они входят. Например, в грамматике G , заданной множеством правил

$$S \rightarrow aSb \mid ba, \quad A \rightarrow aAa \mid bba,$$

нетерминал A не участвует ни в каком выводе, так как из аксиомы нельзя вывести цепочку, содержащую этот нетерминал. Поэтому из заданной грамматики можно удалить нетерминал A .

Рассмотрим, как для произвольной КС-грамматики можно построить эквивалентную КС-грамматику, аксиома которой зависит от всех нетерминальных символов.

Алгоритм 3.19. Удаление «неиспользуемых» нетерминалов

Входные данные:

Грамматика $G = (V_T, V_N, P, S)$

Выход:

\tilde{V}_N — словарь «используемых» нетерминальных символов

// Инициализация

$W_0 \leftarrow \emptyset$

$W_1 \leftarrow \{S\}$

$k \leftarrow 1$

// Основной цикл

while $W_k \neq W_{k-1}$ **do**

$W_{k+1} \leftarrow W_k \cup \{B \mid A \rightarrow \alpha B \beta \in P, A \in W_k\}$

$k \leftarrow k + 1$

end while

$\tilde{V}_N \leftarrow W_k$

 **Замечание 3.56**

Для последнего множества W_n нетерминал $B \in W_n$ тогда и только тогда, когда аксиома S зависит от B . Все нетерминалы, не содержащиеся в W_n , можно удалить из грамматики вместе с правилами, в которые они входят.

Пример 3.169. Пусть дана КС-грамматика G :

$$S \rightarrow abS \mid ASa \mid ab, \quad A \rightarrow abAa \mid ab, \quad B \rightarrow bAab \mid bB.$$

Применим алгоритм 3.19 для построения грамматики, аксиома которой зависит от всех нетерминалов. Имеем:

1) $W_1 = \{S\}$;

2) $W_2 = \{S, A\}$, так как имеется правило $S \rightarrow ASa$ и $S \in W_1$;

3) $W_3 = W_2$.

Получаем эквивалентную грамматику G_1 , аксиома которой зависит от всех нетерминальных символов:

$$S \rightarrow abS \mid ASa \mid ab, \quad A \rightarrow abAa \mid ab.$$

3.6.6. Примеры на построение грамматик

Пример 3.170. Построить грамматику G_1 , порождающую язык

$$L(G_1) = \{a^n b c^m \mid n, m > 0\}.$$

Структура любой цепочки языка имеет вид $\alpha b \beta$, где часть α состоит только из символов a , а часть β — только из символов c . Поэтому обе части можно порождать независимо друг от друга. Получаем следующую грамматику:

$$S \rightarrow AbC; \quad A \rightarrow aA \mid a; \quad C \rightarrow cC \mid c.$$

Построенная грамматика G_1 является контекстно-свободной грамматикой.

Пример 3.171. Построить грамматику G_2 , порождающую язык

$$L(G_2) = \{\alpha \alpha^R \mid \alpha \in \{a, b\}^*\},$$

где α^R обозначает слово α , записанное в обратном порядке. Имеем

$$S \rightarrow aSa \mid bSb \mid \epsilon$$

Построенная грамматика G_2 является контекстно-свободной грамматикой.

Пример 3.172. Построить грамматику G_3 , порождающую язык

$$L(G_3) = \{\alpha \mid \alpha \in \{a, b\}^*\},$$

где слово α начинается и заканчивается одним и тем же символом. Имеем:

$$S \rightarrow aAa \mid bAb; \quad A \rightarrow aA \mid bA \mid \epsilon.$$

Построенная грамматика G_3 является контекстно-свободной грамматикой.

Пример 3.173. Построить грамматику G_4 , порождающую язык

$$L(G_4) = \{a_1 a_2 \dots a_n a_n a_{n-1} \dots a_1 \mid a_i \in \{0, 1\}, 1 \leq i \leq n\}.$$

Имеем:

$$S \rightarrow 0S0 \mid 1S1 \mid \epsilon.$$

Построенная грамматика G_4 является контекстно-свободной грамматикой.

Пример 3.174. Построить грамматику G_5 , порождающую язык

$$L(G_5) = \{a^{2n+1}, | n \geq 0\}.$$

Имеем:

$$S \rightarrow aB \mid a; \quad B \rightarrow aS.$$

Построенная грамматика G_5 является автоматной грамматикой.

Пример 3.175. Построить грамматику G_6 , порождающую язык

$$L(G_6) = \{\lambda \mid \lambda \in \{a, b\}^+\},$$

где цепочка λ не содержит двух рядом стоящих символов a . Имеем:

$$S \rightarrow a \mid aB \mid b \mid bB \mid bA; \quad A \rightarrow a \mid aB; \quad B \rightarrow b \mid bB \mid bA$$

Построенная грамматика G_6 является автоматной грамматикой.

3.6.7. Префиксная и постфиксная записи формул

Привычная запись арифметического выражения в виде инфиксной записи, когда знак операции ставится между операндами, обладает очевидным недостатком: невозможно записать любое арифметическое выражение без использования скобок. В то же время есть другие записи формулы строкой символов, при которых последовательность действий в формуле определяется однозначно без использования дополнительных символов (скобок).

Историческая справка

Способ записи арифметических формул, в котором не используются скобки, предложил в 1920 г. уже упоминавшийся в разделе «Введение в нечеткие множества» польский математик Ян Лукасевич (1878–1956). В честь него постфиксную и префиксную формы записи называют прямой и обратной польской записью. На практике наиболее часто используется постфиксная форма. Поэтому под польской обычно понимают именно постфиксную форму записи.

Определим *постфиксную запись* арифметического выражения.

$$\langle \text{постфикс} \rangle ::= \langle \text{постфикс} \rangle \langle \text{постфикс} \rangle \langle \text{знак операции} \rangle \mid \langle \text{буква} \rangle,$$

где $\langle \text{постфикс} \rangle$ означает постфиксную запись арифметического выражения, а $\langle \text{знак операции} \rangle$ и $\langle \text{буква} \rangle$ определяются, как и раньше, в грамматике (3.91):

$$\langle \text{буква} \rangle ::= a \mid \dots \mid z, \quad \langle \text{знак операции} \rangle ::= + \mid - \mid * \mid /.$$

Пример 3.176. Написать постфиксную запись арифметического выражения $(a + b) * c - d$.

Найдем в нашем выражении последнюю операцию (минус) и перепишем выражение в виде, соответствующем определению постфиксной записи:

$$\langle (a + b) * c \rangle \langle d \rangle -,$$

где выражение, взятое в угловые скобки ($\langle f \rangle$), следует читать как *постфиксная запись формулы f*.

Далее аналогично раскроем выражения в угловых скобках

$$\langle a + b \rangle \langle c \rangle * \langle d \rangle -$$

и, наконец,

$$\langle a \rangle \langle b \rangle + \langle c \rangle * \langle d \rangle -$$

или

$$ab + c * d -$$

поскольку постфиксная запись выражения из одной буквы — это сама эта буква.

Аналогично определяется *префиксная запись*.

$$\langle \text{префикс} \rangle ::= \langle \text{знак операции} \rangle \langle \text{префикс} \rangle \langle \text{префикс} \rangle \mid \langle \text{буква} \rangle \quad (3.94)$$

Пример 3.177. Найти обычную запись арифметического выражения по его префиксной записи $/a + b * -c d f$.

Заметим, что грамматика (3.94) удовлетворяет условию однозначности ветвления по первому символу. Поэтому грамматический анализ выражения выполняется однозначно:

$$\begin{aligned} &\langle \text{префикс} \rangle \\ &\langle \text{знак операции} \rangle \langle \text{префикс} \rangle \langle \text{префикс} \rangle \\ &/ \langle \text{префикс} \rangle \langle \text{префикс} \rangle \\ &/a \langle \text{префикс} \rangle \\ &/a \langle \text{знак операции} \rangle \langle \text{префикс} \rangle \langle \text{префикс} \rangle \\ &/a + \langle \text{префикс} \rangle \langle \text{префикс} \rangle \\ &/a + b \langle \text{префикс} \rangle \\ &/a + b \langle \text{знак операции} \rangle \langle \text{префикс} \rangle \langle \text{префикс} \rangle \\ &/a + b * \langle \text{префикс} \rangle \langle \text{префикс} \rangle \\ &/a + b * \langle \text{знак операции} \rangle \langle \text{префикс} \rangle \langle \text{префикс} \rangle \langle \text{префикс} \rangle \\ &/a + b * - \langle \text{префикс} \rangle \langle \text{префикс} \rangle \langle \text{префикс} \rangle \\ &/a + b * -c \langle \text{префикс} \rangle \langle \text{префикс} \rangle \end{aligned}$$

$$\begin{aligned} & /a + b * -c d \langle \text{префикс} \rangle \\ & /a + b * -c d f \end{aligned}$$

Если просмотреть полученный разбор повторно, переставив операнды (для удобства знаки операций и операнды можно пронумеровать) и взяв при необходимости выражения в скобки (это можно делать всегда, либо, если использовать приоритет операции, когда предыдущий знак имеет приоритет больше данного), то получим искомую формулу (см. таблицу 3.54).

Таблица 3.54.

$\langle \text{префикс} \rangle_1$	$\langle \text{инфикс} \rangle_1$
$\langle \text{знак операции} \rangle_1 \langle \text{префикс} \rangle_{2a}$ $\langle \text{префикс} \rangle_{2b}$	$\langle \text{инфикс} \rangle_{2a} \langle \text{знак операции} \rangle_1$ $\langle \text{инфикс} \rangle_{2b}$
$/ \langle \text{префикс} \rangle_{2a} \langle \text{префикс} \rangle_{2b}$	$\langle \text{инфикс} \rangle_{2a} / \langle \text{инфикс} \rangle_{2b}$
$/a \langle \text{префикс} \rangle_{2b}$	$a / \langle \text{инфикс} \rangle_{2b}$
$/a \langle \text{знак операции} \rangle_2 \langle \text{префикс} \rangle_{3a}$ $\langle \text{префикс} \rangle_{3b}$	$a / \langle \text{инфикс} \rangle_{3a} \langle \text{знак операции} \rangle_2$ $\langle \text{инфикс} \rangle_3 b$
$/a + \langle \text{префикс} \rangle_{3a} \langle \text{префикс} \rangle_{3b}$	$a / (\langle \text{инфикс} \rangle_{3a} + \langle \text{инфикс} \rangle_{3b})$
$/a + b \langle \text{префикс} \rangle_{3b}$	$a / (b + \langle \text{инфикс} \rangle_{3b})$
$/a + b \langle \text{знак операции} \rangle_3 \langle \text{префикс} \rangle_{4a}$ $\langle \text{префикс} \rangle_{4b}$	$a / (b + \langle \text{инфикс} \rangle_{4a} \langle \text{знак операции} \rangle_3$ $\langle \text{инфикс} \rangle_{4b})$
$/a + b * \langle \text{префикс} \rangle_{4a} \langle \text{префикс} \rangle_{4b}$	$a / (b + \langle \text{инфикс} \rangle_{4a} * \langle \text{инфикс} \rangle_{4b})$
$/a + b * \langle \text{знак операции} \rangle_4 \langle \text{префикс} \rangle_{5a}$ $\langle \text{префикс} \rangle_{5b} \langle \text{префикс} \rangle_{4b}$	$a / (b + \langle \text{инфикс} \rangle_{5a} \langle \text{знак операции} \rangle_4$ $\langle \text{инфикс} \rangle_{5b} * \langle \text{инфикс} \rangle_{4b})$
$/a + b * - \langle \text{префикс} \rangle_{5a} \langle \text{префикс} \rangle_{5b}$ $\langle \text{префикс} \rangle_{5b}$	$a / (b + (\langle \text{инфикс} \rangle_{4a} - \langle \text{инфикс} \rangle_{5b})$ $* \langle \text{инфикс} \rangle_{4b})$
$/a + b * -c \langle \text{префикс} \rangle_{5b} \langle \text{префикс} \rangle_{4b}$	$a / (b + (c - \langle \text{инфикс} \rangle_{5b}) *$ $\langle \text{инфикс} \rangle_{4b})$
$/a + b * -c d \langle \text{префикс} \rangle_{4b}$	$a / (b + (c - d) * \langle \text{инфикс} \rangle_{4b})$
$/a + b * -c d f$	$a / (b + (c - d) * f)$

Докажем, что постфиксная и префиксная записи однозначно определяют арифметическое выражение.

Доказательство. Сопоставим каждой постфиксной записи упорядоченный набор целых чисел следующим образом: двигаясь слева направо (и начиная с нуля), будем добавлять к предыдущему числу 1, если очередной символ буква, и -1 , если это знак операции.

Пример 3.178. Выражению $ab + cd * f - /$ соответствует такой набор:

a	b	$+$	c	d	$*$	f	$-$	$/$
1	2	1	2	3	2	3	2	1

Лемма 3.22. Набор, соответствующий постфиксной записи арифметического выражения, обязательно заканчивается на 1.

Доказательство. Докажем индукцией по n — длине постфиксной записи. Если запись имеет длину 1, то она состоит из одной буквы и по правилу ей будет сопоставлена 1, что дает базу индукции. Предположим теперь, что утверждение доказано для всех постфиксных записей арифметических выражений, длина которых меньше n , и докажем его справедливость для n . По определению постфиксная запись арифметического выражения для $n > 1$ имеет вид:

$$\langle \text{постфикс} \rangle ::= \langle \text{постфикс} \rangle \langle \text{постфикс} \rangle \langle \text{знак операции} \rangle,$$

где длина постфиксных записей в правой части меньше n , и поэтому к ним применимо индукционное предположение. Таким образом первая постфиксная запись будет оканчиваться на 1. Тогда числа, сопоставляемые второй постфиксной записи, будут увеличены на 1, а значит (по индукционному предположению), вторая запись будет оканчиваться на 2.

$\langle \text{постфикс} \rangle$	$\langle \text{постфикс} \rangle$	$\langle \text{знак операции} \rangle$
1 ... 1	... 2	1

Наконец, по правилу сопоставления знаку операции будет сопоставлено $2 - 1 = 1$, что и доказывает лемму. ■

Лемма 3.23. Все числа набора, соответствующего постфиксной записи арифметического выражения, больше либо равны 1 (т. е. натуральные).

Упражнение 3.27. Докажите лемму 3.23 самостоятельно по аналогии с леммой 3.22.

Рассмотрим последнюю выполняемую операцию арифметического выражения и два операнда относительно нее. Заметим, что по определению

постфиксной записи все символы одного операнда идут подряд. Как выделить из записи первый и второй операнды? Очевидно, знак операции стоит последним. Где граница между первым и простым операндом? Оказывается, она определяется предпоследней единицей записи.

Лемма 3.24. Все числа, соответствующие символам второго операнда, больше либо равны 2.

Доказательство. По лемме 3.22 все числа набора, соответствующего постфиксной записи арифметического выражения, больше либо равны 1. При нумерации символов второго операнда (который сам по себе является постфиксной записью арифметического выражения и при нумерации с нуля дает числа больше либо равные 1) происходит увеличение всех чисел на 1 (так как первый операнд заканчивается на 1), что и доказывает лемму. ■

Для окончания доказательства теоремы 3.47 сформулируем алгоритм разбора, основанный на леммах 3.22—3.24.

Алгоритм 3.20. Разбор постфиксной записи арифметического выражения

- 1: Сопоставляем символам постфиксной записи числа, как указано выше.
- 2: Находим предпоследнюю 1 в полученном наборе, которая укажет конец первого операнда (последняя единица ограничивает справа второй операнд).
- 3: Применяем шаги 1 и 2 к полученным операндам.

Теорема 3.47 доказана. ■

Пример 3.179. Построить синтаксическое дерево арифметического выражения, заданного постфиксной записью $ab + cd * f - /$.

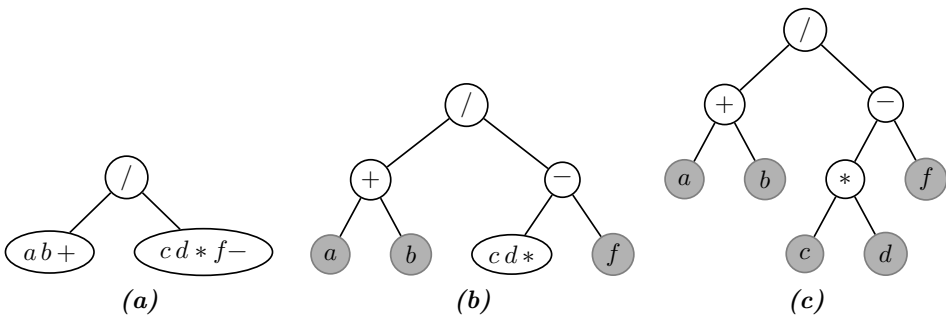


Рис. 3.37.

Разделив операнды, поместим знак в корень дерева, а операнды рассмотрим как соответственно левое и правое поддеревья (рисунок 3.37a).

a	b	$+$	c	d	$*$	f	$-$	$/$
1	2	1	2	3	2	3	2	1

Далее применим ту же процедуру к поддеревьям (рисунок 3.37b).

a	b	$+$	c	d	$*$	f	$-$
1	2	1	1	2	1	2	1

Процедура закончится, когда все поддеревья будут состоять из отдельных символов (рисунок 3.37c).

c	d	$*$	f
1	2	1	1

f
1

3.6.8. Алгоритмы обработки скобочных записей

Рассмотрим сначала алгоритм вычисления значений в постфиксной форме. Отчасти он отвечает на вопрос: зачем нужна такая запись арифметического выражения? Алгоритм крайне прост. Он использует стековую структуру организации памяти для хранения промежуточных вычислений (напомним, что стек определяется операциями *очистить стек*, *поместить в стек*, *извлечь из стека*, причем последовательность извлечений является обратной по сравнению с последовательностью добавлений).

Алгоритм 3.21. Вычисление значений в постфиксной форме

- 1: Очищаем стек.
 - 2: Если очередной символ входной строки — константа, то помещаем ее в стек.
 - 3: Если очередной символ — знак операции, то извлекаем последовательно из стека две верхние константы, используем их в качестве соответственно второго и первого операндов для этой операции, затем помещаем результат обратно в стек.
 - 4: Когда вся входная строка будет разобрана, в стеке должно остаться одно число, которое и будет результатом данного выражения.
-

Пример 3.180. Вычислим значение выражения $ab + cd * f - /$ при $a = 7$, $b = 8$, $c = 4$, $d = 3$, $f = 9$. Тогда исходная строка примет вид $78 + 43 * 9 - /$. Протокол работы алгоритма 3.21 запишем в таблице 3.55.

Таблица 3.55.

Текущий символ	Действие	Состояние стека
7	Помещаем константу 7 в стек	7
8	Помещаем константу 8 в стек	87
+	Извлекаем из стека две верхние константы (8 и 7), совершаем операцию $7 + 8 = 15$, результат помещаем в стек	15
4	Помещаем константу 4 в стек	415

Продолжение таблицы 3.55

Текущий символ	Действие	Состояние стека
3	Помещаем константу 3 в стек	3 4 15
*	Извлекаем из стека две верхние константы (3 и 4), совершаем операцию $4 * 3 = 12$, результат помещаем в стек	12 15
9	Помещаем константу 9 в стек	9 12 15
–	Извлекаем из стека две верхние константы (9 и 12), совершаем операцию $12 - 9 = 3$, результат помещаем в стек	3 15
/	Извлекаем из стека две верхние константы (3 и 15), совершаем операцию $15/3 = 5$, результат помещаем в стек	5

Теперь строка разобрана и в стеке находится одна константа 5, которая является результатом исходного выражения.

Теорема 3.48. Приведенный алгоритм 3.21 вычисляет значение постфиксной записи арифметического выражения (если оно определено).

Доказательство. Докажем индукцией по n — длине постфиксной записи. Если $n = 1$, то правильность алгоритма очевидна (значение переменной заносится в стек, и оно является значением выражения).

Пусть утверждение верно для всех постфиксных записей арифметических выражений с числом символов меньше n . Докажем, что тогда оно будет верно и для записи с n символами.

Воспользуемся определением и рассмотрим постфиксную запись как последовательность трех частей. Протокол вычислений запишем в таблице 3.56.

Таблица 3.56.

$\langle \text{постфикс} \rangle$	$\langle \text{постфикс} \rangle$	$\langle \text{знак операции} \rangle$
Значение первого операнда	Значение второго операнда	Результат выполнения данной операции над первым и вторым операндом
	Значение первого операнда	

Здесь использован тот факт, что правильность вычисления как первого, так и второго операнда обеспечивается индукционным предположением. При вычислении второго операнда значение первого останется *на дне* стека, поэтому перед вычислением операции в стеке будут два числа: значение первого операнда внизу, и второго операнда — наверху. В соответствии с алгоритмом вычислений над ними будет проведена операция, которая, очевидно, и даст значение выражения. ■

Сформулируем теперь алгоритм перевода скобочной инфиксной записи арифметического выражения в постфиксную форму. Этот алгоритм также будет использовать стек, но только для символьных переменных. Кроме того, для каждой операции будет определен приоритет (таблица 3.57). (Поскольку запись не является полной скобочной, это необходимо для определения точного порядка действий.)

Таблица 3.57.

Операция	Приоритет	Операция	Приоритет
(0	—	1
)	0	*	2
+	1	/	2

Алгоритм 3.22. Перевод инфиксной формы в постфиксную

- 1: Очищаем стек.
 - 2: Если текущий символ — константа или переменная, то помещаем его в выходную строку.
 - 3: Если текущий символ — знак операции или скобка, то определяем приоритет операции согласно таблице 3.57. Далее проверяем стек.
 - 4: Если стек пуст или находящиеся в нем символы (находиться в нем могут только знаки операций и открывающая скобка) имеют меньший приоритет, чем приоритет текущего символа, то помещаем текущий символ в стек.
 - 5: Если символ, находящийся на вершине стека, имеет приоритет, больший или равный приоритету текущего символа, то извлекаем символы из стека в выходную строку до тех пор, пока выполняется это условие; затем переходим к шагу 4.
 - 6: Если текущий символ — открывающая скобка, то помещаем ее в стек.
 - 7: Если текущий символ — закрывающая скобка, то извлекаем символы из стека в выходную строку до тех пор, пока не встретим в стеке открывающую скобку, которую уничтожаем. Закрывающая скобка также уничтожается.
 - 8: Если вся входная строка разобрана, а в стеке еще остаются знаки операций, извлекаем их из стека в выходную строку.
-

Пример 3.181. Переведем арифметическое выражение $(a + b) * c - d$ в постфиксную форму по алгоритму 3.22. Протокол его работы запишем в таблице 3.58.

Таблица 3.58.

Текущий символ	Действие	Выходная строка	Состояние стека
(Помещаем (в стек		(

Продолжение таблицы 3.58

Текущий символ	Действие	Выходная строка	Состояние стека
a	Помещаем a в выходную строку	a	(
+	Проверяем стек: на вершине находится символ (, приоритет которого меньше, чем приоритет текущего символа +. Помещаем в стек +	a	+(
b	Помещаем b в выходную строку	ab	+(
)	Извлекаем из стека + и (. Помещаем + в выходную строку. Скобки уничтожаем	$ab+$	
*	Помещаем * в стек	$ab+$	*
c	Помещаем c в выходную строку	$ab+c$	*
–	Проверяем стек: на вершине находится символ *, приоритет которого больше, чем приоритет текущего символа –. Извлекаем из стека в выходную строку *. Помещаем в стек –	$ab+c*$	–
d	Помещаем d в выходную строку	$ab+c*d$	–
	Выходная строка пуста. Извлекаем из стека – и помещаем выходную строку	$ab+c*d-$	

Задачи и вопросы

1. Дайте определение формальной грамматики.
2. Цепочки 01 и 000111 являются правильными словами языка. Постройте грамматику для этого языка.
3. Что такое *непосредственная выводимость* в порождающей грамматике?
4. Что такое *выводимость* в порождающей грамматике?
5. Что такое язык, порожденный грамматикой?
6. Какие ограничения накладываются на правила вывода контекстно-зависимых грамматик?
7. Какие ограничения накладываются на правила вывода контекстно-свободных грамматик?

8. Какие ограничения накладываются на правила вывода автоматных грамматик?

9. Что означает условие *однозначности ветвления по первому символу* для КС-грамматики?

10. Что такое *непродуктивный (непроизводящий)* нетерминальный символ грамматики? Как устранить непродуктивные символы из грамматики?

11. К какому типу (по Хомскому) относится данная грамматика? Какой язык она порождает?

а) $S \rightarrow bSS|a$;

б) $S \rightarrow Ab|a, A \rightarrow Ba, B \rightarrow cS$;

в) $S \rightarrow Ab, A \rightarrow Aa|ba$;

г) $S \rightarrow aSb, S \rightarrow \epsilon$;

д) $S \rightarrow AaA|AaB, A \rightarrow aA|\epsilon, B \rightarrow b$.

12. Построить КС-грамматику, порождающую

а) язык $L = \{a^n b^n c^m \mid n, m \geq 1\}$;

б) слова из нулей и единиц с одинаковым числом тех и других;

в) язык регулярных выражений $0^*1(0+1)^*$.

13. Опишите язык, который определяет КС-грамматика $S ::= S101|01$. Удовлетворяет ли она условию однозначности ветвления?

14. Сколько различных алгебраически эквивалентных постфиксных записей соответствует выражению $a_1 + \dots + a_n$?

15. Постройте грамматику постфиксной записи арифметических выражений, содержащих, кроме бинарных операций (сложение, умножение и т. д.), также и унарные операции (не умаляя общности, достаточно рассмотреть одну — извлечение квадратного корня).

16. Сформулируйте алгоритм:

а) вычисления значений арифметического выражения в постфиксной записи, если оно содержит как бинарные, так и унарные операции;

б) перевода арифметических выражений в скобочной инфиксной записи, содержащих как бинарные, так и унарные операции, в постфиксную форму;

в) перевода инфиксной записи ДНФ, не содержащей операций отрицания, в префиксную форму и докажете его корректность;

- г) перевода инфиксной записи многочлена Жегалкина в постфиксную форму и докажете его корректность;
- д) перевода выражения из скобочной инфиксной записи в префиксную;
- е) вычисления значений арифметического выражения в префиксной записи.

3.7. Конечные автоматы-распознаватели

3.7.1. Введение

Теория автоматов относится к числу ключевых разделов современной математики. Она непосредственно связана с математической логикой, теорией алгоритмов, теорией формальных грамматик. Универсальность и сравнительная простота автоматов обусловили широкое использование автоматных моделей на практике. Теория автоматов, например, успешно используется при построении узлов цифровых вычислительных машин, применяется при разработке парсеров для формальных языков (в том числе языков программирования), а также при построении компиляторов и разработке самих языков программирования.

Конечный автомат — это один из самых простых механизмов, используемых при работе с языками. Основная часть конечного автомата — это функция перехода, определяющая возможные переходы для любого текущего состояния и любого входного символа. Подразумевается, что допускается возможность перехода сразу в несколько различных состояний автомата, т. е. управляющее устройство распознавателя может быть и недетерминированным.

Под автоматом в общем смысле обычно понимают дискретный преобразователь информации, который принимает входные сигналы в дискретные моменты времени, изменяет свое состояние (с учетом прежнего состояния) и, возможно, формирует выходные сигналы.

Историческая справка

Основы теории автоматов были заложены выдающимся математиком венгерского происхождения Джоном фон Нейманом (1903–1957). Он родился в Будапеште в семье состоятельного венгерского банкира. Родители назвали его Яношем, в годы учебы и работы в Швейцарии и Германии он именовал себя Иоганном, а после переезда в США (в 1930 г.) — Джоном. В 1952 г. в работе «Вероятностная логика и синтез надежных организмов из ненадежных элементов» он показал путь создания надежных ЭВМ и других автоматов. Это даже не книга, а 5 лекций, написанных Нейманом для своего друга Ричарда Пирса. На русском языке работа была опубликована в 1956 г. в сборнике «Автоматы» [43].

Джон фон Нейман — один из создателей атомной бомбы (математически доказал осуществимость взрывного способа детонации атомной бомбы), он сформулировал основную концепцию логической организации ЭВМ (принцип совместного хранения команд и данных в памяти компьютера, называемый архитектурой фон Неймана, моделью фон Неймана или принстонской архитектурой), что послужило огромным толчком к развитию электронно-вычислительной техники.

3.7.2. Автоматы Мили и Мура

Дадим теперь формальные определения. Начнем с детерминированных автоматов.

Определение 3.128. Конечный детерминированный автомат с выходом (*автомат Мили*¹) — это система $A = (Q, \Sigma, \Omega, \delta, \lambda, q_0)$, где:

- Q — конечное *непустое* множество состояний;
- $q_0 \in Q$ — начальное состояние;
- Σ — конечное *непустое* множество входных символов (входной алфавит);
- Ω — конечное *непустое* множество выходных символов (выходной алфавит);
- $\delta : Q \times \Sigma \rightarrow Q$ — *всюду определенное* отображение множества $Q \times \Sigma$ в множество Q , определяющее поведение автомата (эту функцию часто называют *функцией переходов*);
- $\lambda : Q \times \Sigma \rightarrow \Omega$ — *всюду определенное* отображение множества $Q \times \Sigma$ в множество Ω , определяющее выходную последовательность автомата (эту функцию часто называют *функцией выходов*).

Автомат начинает работу в состоянии q_0 , считывая по одному символу входной строки. Считанный символ переводит автомат в новое состояние из Q в соответствии с функцией переходов δ и возвращает соответствующий выходной символ согласно функции выходов λ .

Существуют два основных способа описания конечного автомата.

1. *Диаграмма состояний* (или иногда граф переходов) — графическое представление множества состояний и функции переходов. Представляет собой нагруженный ориентированный граф, вершины которого — состояния автомата, дуги — переходы из одного состояния в другое, а нагрузка — входные/выходные символы, при которых осуществляется данный переход. Если переход из состояния q_i в q_j может быть осуществлен при появлении одного из нескольких символов, то над дугой должны быть написаны все они.

2. *Таблица переходов* (табличное представление) функции δ . Обычно в такой таблице каждой строке соответствует одно состояние, а столбцу — один допустимый входной символ. В ячейке на пересечении строки и столбца записывается состояние, в которое должен перейти автомат, если в ситуации, когда он находился в данном состоянии, на входе он получил данный символ и соответствующий выходной символ.

¹ Иногда его называют автоматом I рода.

Пример 3.182. Рассмотрим автомат Мили по продаже шоколадок стоимостью 20 рублей, принимающий монеты номиналом в 5 и 10 рублей и возвращающий сдачу, если это необходимо¹.

Состояний автомата четыре: q_0, q_5, q_{10}, q_{15} — 0, 5, 10 и 15 рублей.

Входных сигналов два: σ_5 — 5 рублей и σ_{10} — 10 рублей.

Выходных сигналов три: ω_n — ничего не выдавать, ω_0 — выдать шоколадку и 0 рублей сдачи, ω_5 — выдать шоколадку и 5 рублей сдачи.

На рисунке 3.38a приведена диаграмма состояний, а на рисунке 3.38b — таблица переходов рассматриваемого автомата.

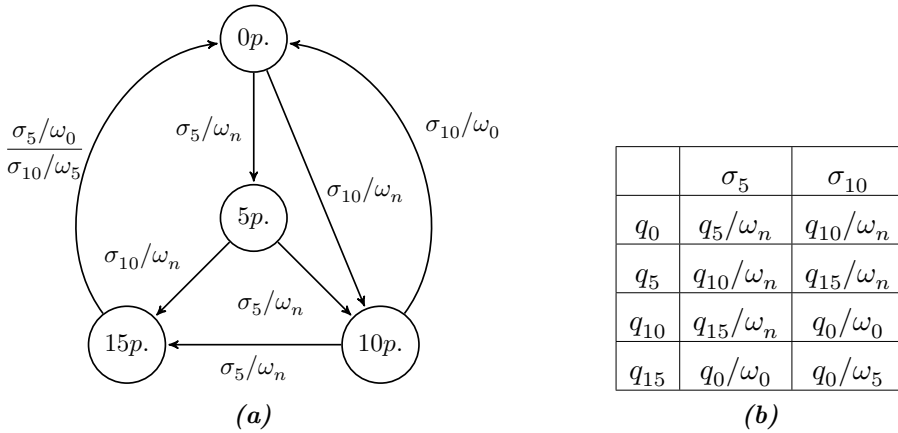


Рис. 3.38.

Определение 3.129. Пусть

$$A = (Q_A, \Sigma, \Omega, \delta_A, \lambda_A, q_0), \quad B = (Q_B, \Sigma, \Omega, \delta_B, \lambda_B, p_0)$$

два автомата с одинаковыми входными и выходными алфавитами. Состояние q автомата A и состояние p автомата B называются *неотличимыми*, если для любого слова $\alpha \in \Sigma^*$ выходные слова при достижении q и p совпадают.

Автоматы A и B называются *эквивалентными*, если для любого состояния автомата A найдется неотличимое от него состояние автомата B и наоборот.

Иногда удобным бывает более простое определение автомата, в котором функция переходов задает следующее состояние, а выход автомата зависит лишь от его текущего состояния.

Определение 3.130. Конечный автомат называется *автоматом Мура* или *автоматом II рода*, если для любого $q \in Q$ значения функции выхо-

¹ Пример взят из *Викиконспекты*: <https://neerc.ifmo.ru/wiki/...>

дов на дугах, входящих в q , совпадают. То есть функция выходов не зависит от входного символа, а только от состояния $\lambda : Q \rightarrow \Omega$.

В графическом представлении автоматов Мура часто вершину состояния $q_i \in Q$ помечают также соответствующим выходным символом $w_j \in \Omega$.

Пример 3.183. Пусть на вход автомата подается последовательность из нулей и единиц. Построим автомат Мура, который возвращает накопленное число единиц по модулю 3.

Как и в предыдущем примере 3.182, начальное состояние обозначим q_0 . Диаграмма состояний рассматриваемого автомата приведена на рисунке 3.39. Выходной сигнал однозначно определяется состоянием, в которое автомат перешел, поэтому выходные сигналы приписаны прямо к состояниям.

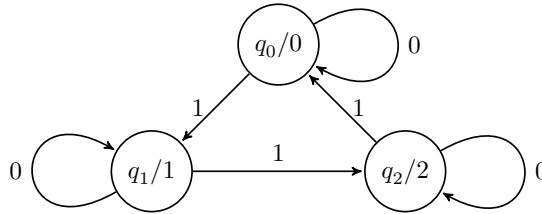


Рис. 3.39.

3.7.3. Примеры автоматов Мили и Мура

Пример 3.184. Реализуем с помощью автомата Мили двоичный последовательный сумматор для одного разряда входного кода.

Автомат имеет два состояния: q_0 — нет переноса единицы, начальное состояние, q_1 — есть перенос. Входной алфавит автомата — это пары битов обоих операндов, последовательно поступающих на сумматор: $(0, 0)$, $(0, 1)$, $(1, 0)$ и $(1, 1)$. Выходной алфавит, результат работы сумматора: $\{0, 1\}$.

Диаграмма состояний рассматриваемого автомата приведена на рисунке 3.40.

Пример 3.185. Построим автомат Мили, управляющий лифтом в трехэтажном доме.

Входной алфавит автомата состоит из нажатия кнопки вызова соответствующего этажа: $\{C_1, C_2, C_3\}$. Выходной алфавит состоит из перемещений на один или два этажа вверх или вниз, а также остановки лифта:

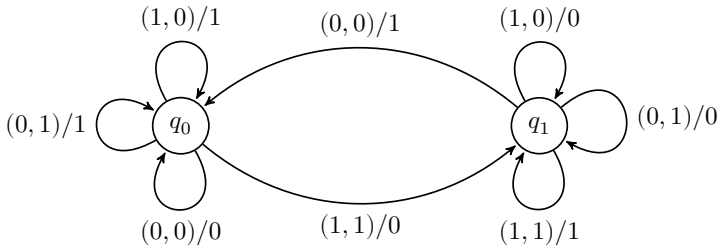


Рис. 3.40.

$\{U_1, U_2, D_1, D_2, S\}$; состояние соответствует этажу, на котором находится автомат: $\{q_1, q_2, q_3\}$; для определенности выберем начальное состояние q_1 .

Диаграмма состояний рассматриваемого автомата приведена на рисунке 3.41.

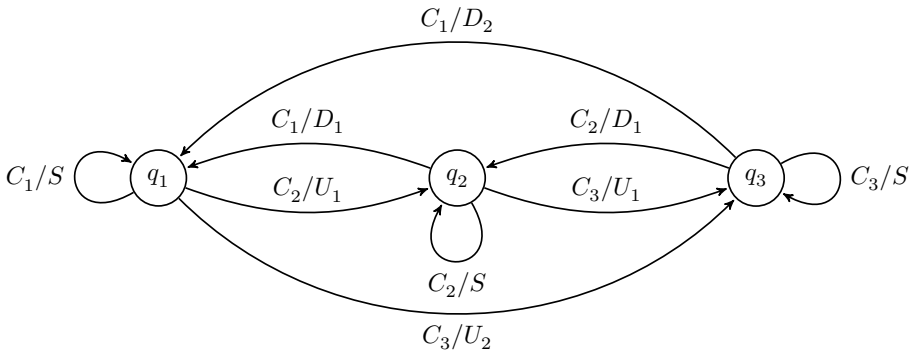


Рис. 3.41.

Пример 3.186. Построим автомат Мили, который удаляет повторяющиеся подряд символы в словах в алфавите $\{a, b, c\}$ (оставляя только один). Например, слово *abbaccaa* должно преобразоваться в слово *abca*.

В выходной алфавит добавим пустой символ ϵ . Пустой символ будем возвращать при удалении символов входного слова. Начальное состояние обозначим q_0 . Диаграмма состояний рассматриваемого автомата приведена на рисунке 3.42.

3.7.4. Эквивалентность автоматов Мили и Мура

Согласно определениям 3.128 и 3.130 автомат Мура есть частный случай автомата Мили. Однако оба автомата являются эквивалентными.

Теорема 3.49 (об эквивалентности автоматов Мили и Мура). Для любого автомата Мили существует эквивалентный ему автомат Мура.

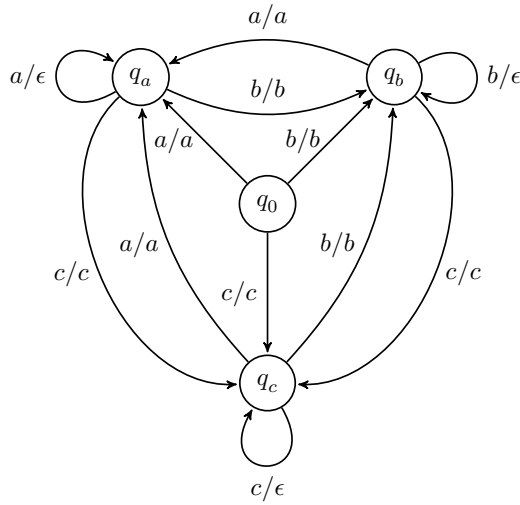


Рис. 3.42.

Доказательство. Преобразование автомата Мура в эквивалентный автомат Мили легко провести в их графическом представлении. Для этого для каждой вершины-состояния символ выходного алфавита Ω , которыми помечена данная вершина, переносим на все дуги, входящие в нее. Очевидно, что получившийся автомат Мили эквивалентен исходному.

На рисунке 3.43 показано данное преобразование для автомата Мура из примера 3.183.

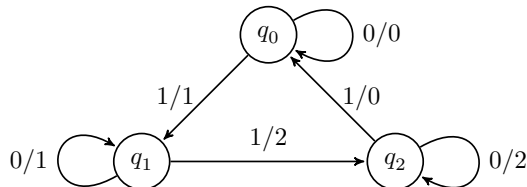


Рис. 3.43.

Рассмотрим алгоритм обратного преобразования автомата Мили в автомат Мура. Для этого выполним действия, обратные действиям при преобразовании автомата Мура в автомат Мили, т. е. выходной символ, которым помечена дуга графа автомата, перенесем в вершину, в которую эта дуга входит. Однако такое преобразование невозможно для случая нескольких входящих в вершину дуг с разными выходными символами. Тогда это состояние необходимо «расщепить» на такое количество состояний, сколько различных входных символов имеется на всех входных ребрах этого состояния, и сопоставить каждому из них свой выходной символ.

Схематически идея «расщепления» состояний показана на рисунке 3.44.

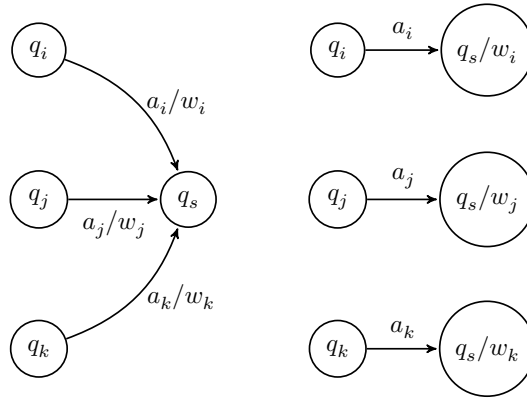


Рис. 3.44.

Рассмотрим эту процедуру на примере автомата Мили на рисунке 3.45.

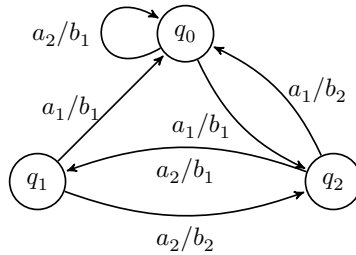


Рис. 3.45.

Каждому состоянию автомата Мили $q_i \in Q$ поставим в соответствие множество пар A_i :

$$A_i = \{(q_i, \omega_j) \mid q_i = \delta(q_m, \alpha_s), \omega_j = \lambda(q_m, \alpha_s), q_m \in Q, \alpha_s \in \Sigma, \omega_j \in \Omega\}. \tag{3.95}$$

Согласно (3.95) для автомата на рисунке 3.45 имеем:

$$A_0 = \begin{cases} (q_0, b_1) = p_0, \\ (q_0, b_2) = p_1, \end{cases} \quad A_1 = \{(q_1, b_1) = p_2\}, \quad A_2 = \begin{cases} (q_2, b_1) = p_3, \\ (q_2, b_2) = p_4. \end{cases} \tag{3.96}$$

Таким образом, получаем состояния автомата Мура: $\{p_0, p_1, p_2, p_3, p_4\}$. Обозначим функции переходов и выходов для автомата Мура как $\hat{\delta}$ и $\hat{\lambda}$. Функция выхода определяется вторым элементом соответствующей пары в (3.96).

$$\hat{\lambda}(p_0) = b_1, \hat{\lambda}(p_1) = b_2, \hat{\lambda}(p_2) = b_1, \hat{\lambda}(p_3) = b_1, \hat{\lambda}(p_4) = b_2.$$

Определим функцию переходов для автомата Мура. Рассмотрим произвольное состояние $q_i \in Q$ и произвольный символ входного алфавита $\alpha \in \Sigma$. Пусть $\delta(q_i, \alpha) = q_s$ и $\lambda(q_i, \alpha) = \omega_m \in \Omega$. Тогда в автомате Мура должны быть переходы из всех состояний A_i в состояние $(q_s, \omega_m) \in A_s$, определенное согласно (3.95), под воздействием символа α . Например, в автомате Мили $\delta(q_0, a_1) = q_2, \lambda(q_0, a_1) = b_1$. Тогда для автомата Мура для состояний $p_0, p_1 \in A_0$ имеем $\hat{\delta}(p_0, a_1) = (q_2, b_1) = p_3$ и $\hat{\delta}(p_1, a_1) = p_3$.

В общем виде можно записать таблицу переходов (таблица 3.59).

Таблица 3.59.

$\delta(q_0, a_1) = q_2, \lambda(q_0, a_1) = b_1$	$(q_2, b_1) = p_3$	$\hat{\delta}(p_0, a_1) = \hat{\delta}(p_1, a_1) = p_3$
$\delta(q_0, a_2) = q_0, \lambda(q_0, a_2) = b_1$	$(q_0, b_1) = p_0$	$\hat{\delta}(p_0, a_2) = \hat{\delta}(p_1, a_2) = p_0$
$\delta(q_1, a_1) = q_0, \lambda(q_1, a_1) = b_1$	$(q_0, b_1) = p_0$	$\hat{\delta}(p_2, a_1) = p_0$
$\delta(q_1, a_2) = q_2, \lambda(q_1, a_2) = b_2$	$(q_2, b_2) = p_4$	$\hat{\delta}(p_2, a_2) = p_4$
$\delta(q_2, a_1) = q_0, \lambda(q_2, a_1) = b_2$	$(q_0, b_2) = p_1$	$\hat{\delta}(p_3, a_1) = \hat{\delta}(p_4, a_1) = p_1$
$\delta(q_2, a_2) = q_1, \lambda(q_2, a_2) = b_1$	$(q_1, b_1) = p_2$	$\hat{\delta}(p_3, a_2) = \hat{\delta}(p_4, a_2) = p_2$

В итоге получаем эквивалентный автомат Мура (рисунок 3.46).

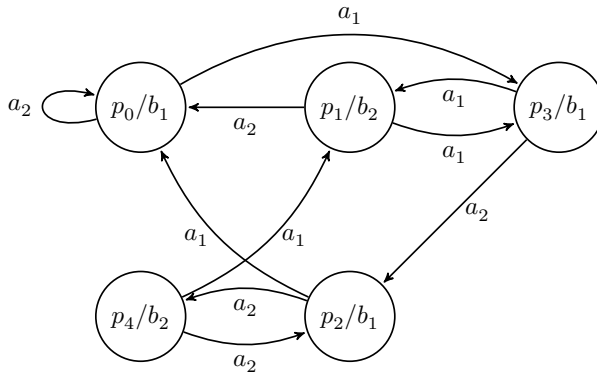


Рис. 3.46.

В качестве начального состояния автомата Мура можно взять любое из состояний p_0 или p_1 , так как оба они порождены состоянием q_0 автомата Мили. ■

Таким образом, можно рассматривать только автоматы Мура.

Для задачи определения принадлежности слова языку достаточно ограничиться случаем $|\Omega| = 2$. Тогда каждому состоянию может соответствовать один из двух выходных символов, будем считать, что тем самым со-

стояния разбиваются на два класса, назовем их *заключительным* и *незаключительным*.

Приходим к понятию детерминированного конечного автомата-распознавателя (раздел 3.7.5).

3.7.5. Основные понятия автоматов-распознавателей

Определение 3.131. Детерминированный конечный автомат-распознаватель A — это пятерка объектов $A = (Q, \Sigma, \delta, q_0, F)$, где:

- Q — конечное *непустое* множество состояний;
- $q_0 \in Q$ — начальное состояние;
- Σ — конечное *непустое* множество входных символов (входной алфавит);
- $\delta : Q \times \Sigma \rightarrow Q$ — *всюду определенное* отображение множества $Q \times \Sigma$ в множество Q , определяющее поведение автомата. Как и ранее (см. определение 3.128), эту функцию будем называть *функцией переходов*;
- $F \subseteq Q$ — множество *заключительных* (финальных) состояний.

Автомат начинает работу в состоянии q_0 , считывая по одному символу входной строки. Считанный символ переводит автомат в новое состояние из Q в соответствии с функцией переходов δ .

Замечание 3.57

Автомат, описанный в определении 3.131, называется *детерминированным*, так как для любой пары $q \in Q, a \in \Sigma$ существует *единственное* состояние $p \in Q$: $p = \delta(q, a)$.

В принципе, для определения последующих действий конечного автомата достаточно знать его текущее состояние и последовательность еще необработанных символов на входной ленте. Этот набор данных называется *конфигурацией автомата*.

Определение 3.132. Слово $w = a_1 \dots a_k$ над алфавитом Σ *допускается* конечным автоматом $M = (Q, \Sigma, \delta, q_0, F)$, если существует последовательность состояний q_1, q_2, \dots, q_n такая, что

$$q_1 = q_0, q_n \in F, \delta(q_i, a_j) = q_{i+1}, 1 \leq i < n, 1 \leq j < k.$$

Определение 3.133. Язык L распознается конечным автоматом A , если каждое слово языка L допускается этим конечным автоматом. При этом язык называется *автоматным* (или *регулярным*) и обозначается L_A .

Определение 3.134. Два состояния автомата называются *эквивалентными*, если для любой строки $\alpha \in \Sigma^*$ результаты ее обработки (допуск или недопуск автоматом), начиная с указанных состояний, совпадают.

Определение 3.135. Автоматы A и B называются эквивалентными, если совпадают распознаваемые ими языки, т. е. $L_A = L_B$.

3.7.6. Примеры автоматов-распознавателей

Рассмотрим диаграммы состояний простейших автоматов. Далее, на рисунках начальное состояние будем отмечать светло-серым цветом, конечные состояния — темно-серым.

Пример 3.187. На рисунке 3.47а приведена диаграмма состояний, а на рисунке 3.47б — таблица переходов детерминированного конечного автомата, допускающего цепочки из 0 и 1, заканчивающиеся символом 0.

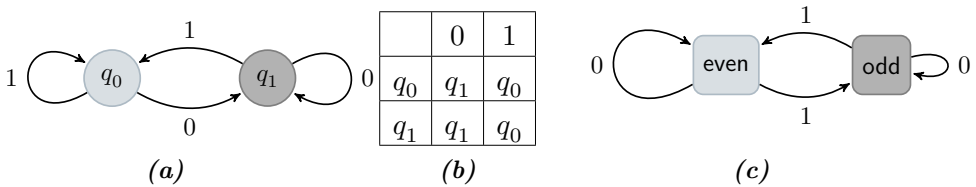


Рис. 3.47.

Пример 3.188. Рассмотрим диаграмму состояний автомата, контролирующего нечетность единиц, то есть распознающего цепочки, состоящих из 0 и 1 и имеющие нечетное число единиц (рисунок 3.47с). Начальное состояние: $q_0 = \text{even}$.

Пример 3.189. Рассмотрим автомат, распознающий слова, содержащие только парные вхождения букв a и b , например aa , $aabbaaaa$, $bbaa$ и т. д. Диаграмма состояний автомата приведена на рисунке 3.48.

Замечание 3.58

Заметим, что автомат из примера 3.189, попав в незаключительное состояние q_3 под воздействием сигналов a, b , не может выйти из него под воздействием тех же входных сигналов (состояние и переходы отмечены на диаграмме 3.48 пунктиром). Будем считать, что переходы автомата в это состояние под воздействием сигналов a, b запрещены. То есть запрещены сигнал b в состоянии q_1 и сигнал a в состоянии q_2 . Далее запрещенные состояния и переходы в диаграмме состояний не изображаем, считая, что если символ входного слова привел к запрещенному переходу, то данное слово не принимается автоматом. Такие состояния (в которые

переходят при обработке любого недопустимого символа) в теории часто называют невозвратными.

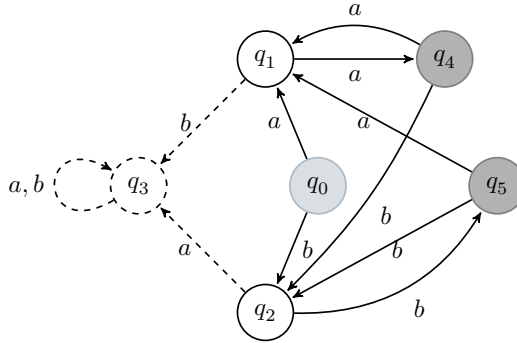


Рис. 3.48.

Пример 3.190. С учетом замечания 3.58 построим автомат, распознающий слова над алфавитом $\{\alpha, \beta\}$, начинающиеся с символов $\alpha\alpha$ и содержащие нечетное число вхождений символа β . Его диаграмма представлена на рисунке 3.49.

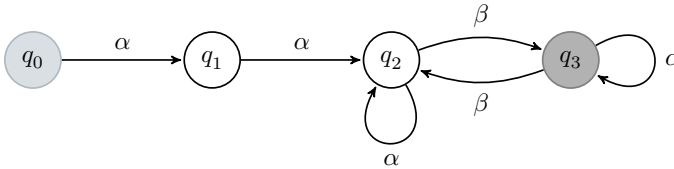


Рис. 3.49.

Пример 3.191. Построим автомат, распознающий слова над алфавитом $\{0, 1\}$, содержащие подслово 00, например $\alpha = 01001$. Его диаграмма представлена на рисунке 3.50.

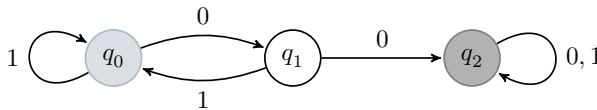


Рис. 3.50.

Пример 3.192. Рассмотрим автомат, распознающий множество вещественных констант ($V_T = \{+, -, 0, \dots, 9, .\}$) (рисунок 3.51). При этом в константе может быть опущен знак и отсутствовать дробная или целая часть (но не обе сразу). Десятичная точка как признак вещественного числа обязательна. При построении диаграммы учитываем замечание 3.58.

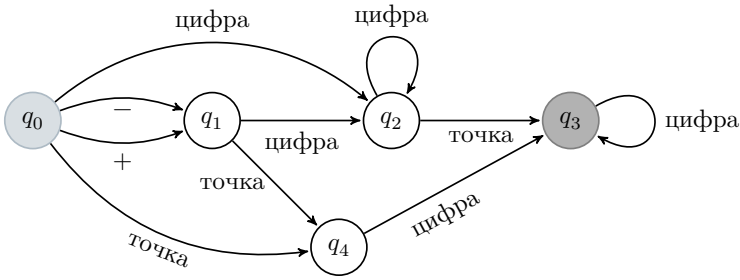


Рис. 3.51.

3.7.7. Недетерминированные автоматы-распознаватели. Детерминизация

Перейдем теперь к недетерминированным автоматам-распознавателям, которые являются обобщением детерминированных.

Определение 3.136. Недетерминированный конечный автомат-распо-

знаватель A — это пятерка объектов $A = (Q, \Sigma, \delta, H, F)$, где:

- Q — конечное *непустое* множество состояний;
- Σ — конечное *непустое* множество входных символов;
- $\delta : Q \times \Sigma \rightarrow 2^Q$ — функция перехода: всюду определенное отображение множества $Q \times \Sigma$ в множество подмножеств Q :

$$\delta(q_i, \alpha_j) = \{q_{i_1}, \dots, q_{i_n}\} \subseteq Q,$$

означает, что из состояния q_i по входному символу α_j можно осуществить переход в любое из состояний q_{i_1}, \dots, q_{i_n} ;

- $H \subseteq Q$ — множество начальных состояний;
- $F \subseteq Q$ — множество заключительных (финальных) состояний.

Замечание 3.59

Отличия от детерминированного автомата (см. определение 3.131) состоят в задании множества начальных состояний H вместо одного q_0 и выходному значению функции перехода δ , которая возвращает подмножество выходных состояний вместо одного.

Для дальнейшего изучения свойств конечных автоматов важное значение имеет следующая теорема.

Теорема 3.50 (о детерминизации). Для любого конечного автомата может быть построен эквивалентный ему детерминированный конечный автомат.

Доказательство. В качестве доказательства приведем два алгоритма: детеминизация объединением и детерминизация вытягиванием для построения детерминированного конечного автомата по недетерминированному. ■

Введем необходимые определения.

Определение 3.137. Состояние $q_i \in Q$ автомата A называется *недостижимым*, если под воздействием любого слова автомат A не переходит в состояние q_i .

Состояния, недостижимые из начального, можно найти поиском в ширину, например, используя следующий алгоритм.

Алгоритм 3.23. Нахождение недостижимых состояний

// Инициализация. Заносим в список L начальное состояние
 $q_0 \rightarrow L$

// Основной цикл

while существуют $q_i \in L, q_j \notin L$ и $\alpha_k \in \Sigma \mid q_j = \delta(q_i, \alpha_k)$ **do**

 // Добавляем в список L новые состояния, достижимые из состояний L

$L \leftarrow L \cup \{q_j\}$

end while

Состояния, не включенные в список L , удаляем со всеми инцидентными им дугами

Рассмотрим теперь алгоритм построения детерминированного конечного автомата по недетерминированному.

На вход алгоритма подается недетерминированный конечный автомат

$$A = (Q, \Sigma, \delta, H, F).$$

В результате работы получаем детерминированный конечный автомат

$$\tilde{A} = (\tilde{Q}, \tilde{\Sigma}, \tilde{\delta}, \tilde{q}_0, \tilde{F}),$$

допускающий тот же язык, что и автомат A .

Алгоритм 3.24. Детерминизация объединением

1: Определяем \tilde{Q} как множество всех подмножеств Q . Обозначим $\tilde{q}_i \in \tilde{Q}$ как

$$\tilde{q}_i = [q_{i_1} \dots q_{i_k}], \quad q_{i_j} \in Q, \quad 1 \leq j \leq k.$$

2: Определяем отображение $\tilde{\delta}$ как объединение состояний:

$$\tilde{\delta}([q_{i_1} \dots q_{i_n}], \alpha) = [q_{j_1} \dots q_{j_m}],$$

где

$$\forall q_{j_s} \in \{q_{j_1}, \dots, q_{j_m}\}, \exists q_{i_p} \in \{q_{i_1}, \dots, q_{i_n}\} : \delta(q_{i_p}, \alpha) = q_{j_s}.$$

3: Определяем начальное состояние \tilde{q}_0 как объединение *всех* начальных состояний H :

$$\tilde{q}_0 = [q_{0_1} \dots q_{0_k}], q_{0_i} \in H, 1 \leq i \leq k, k = |H|. \quad (3.97)$$

4: Определяем множество заключительных состояний \tilde{F} :

$$\tilde{F} = \{\tilde{q}_i \in \tilde{Q} \mid \tilde{q}_i = [\dots q_l \dots], q_l \in F\}.$$

5: Используя алгоритм 3.23, исключаем возможные недостижимые состояния в \tilde{Q} .

Замечание 3.60

Если $|Q| = n$, то после первого шага $|\tilde{Q}| = 2^n$. Таким образом, на шаге 2 необходимо перебрать 2^n возможных состояний нового детерминированного автомата, а отсеечение недостижимых состояний происходит на шаге 5.

Объединение состояний на шаге 2 алгоритма 3.24 иллюстрирует рисунок 3.52

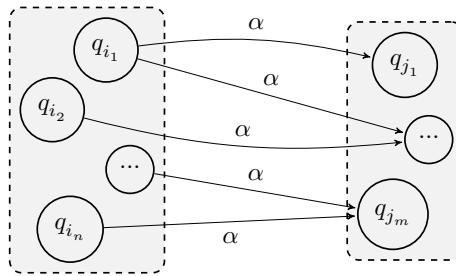


Рис. 3.52.

Пример 3.193. На рисунке 3.53b изображен детерминированный автомат \tilde{A} , построенный по алгоритму 3.24 из недетерминированного автомата A на рисунке 3.53a.

Всего согласно алгоритму 3.24 у нового автомата \tilde{A} должно быть $2^4 = 16$ состояний, но только 4 из них оказываются достижимыми.

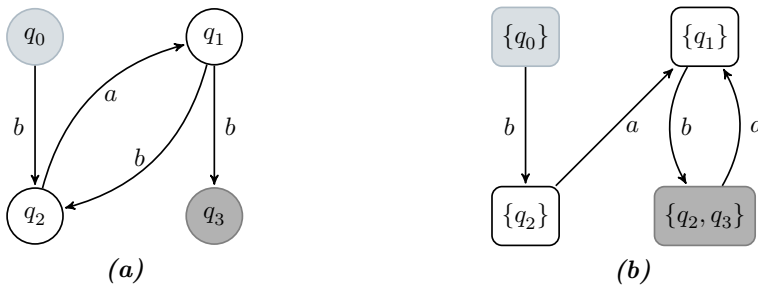


Рис. 3.53.

В алгоритме 3.24 отсеечение недостижимых состояний происходит на последнем шаге. Рассмотрим другой подход, где достижимость состояния нового детерминированного автомата определяется сразу при построении.

Процесс добавления состояний-множеств напоминает расширение списка L в алгоритме 3.23.

Алгоритм 3.25. Детерминизация вытягиванием

// Инициализация. Добавляем начальное состояние.

Начальное состояние $\tilde{q}_0 \in \tilde{Q}$ определяем согласно (3.97).


// Основной цикл

while существует $\tilde{q}_j = \delta(\tilde{q}, \alpha)$, $\tilde{q} \in \tilde{Q}$, $\alpha \in \Sigma \mid \tilde{q}_j \notin \tilde{Q}$ **do**

 // Добавляем новые состояния, непосредственно достижимые из текущего \tilde{Q}

$\tilde{Q} \leftarrow \tilde{Q} \cup \{\tilde{q}_j\}$

end while

 **Замечание 3.61**

На шаге k основного цикла добавляются состояния, достижимые из начального состояния \tilde{q}_0 по пути длиной k .

Пример 3.194. Рассмотрим работу алгоритма 3.25 на примере недетерминированного автомата (рисунок 3.53a). Имеем:

$$\delta_1(\{q_0\}, b) = \{q_2\};$$

$$\delta_1(\{q_2\}, a) = \{q_1\};$$

$$\delta_1(\{q_1\}, b) = \{q_2, q_3\};$$

$$\delta_1(\{q_2, q_3\}, a) = \delta(q_2, a) \cup \delta(q_3, a) = \{q_1\} \cup \emptyset = \{q_1\}.$$

Так как новых состояний-множеств больше не появилось, процедура «вытягивания» на этом заканчивается, и мы получаем граф, изображенный на рисунке 3.53b.

Пример 3.195. Рассмотрим работу алгоритма 3.25 на примере недетерминированного автомата (рисунок 3.54). Имеем:

$$\delta_1(\{q_0\}, a) = \{q_0, q_1\};$$

$$\delta_1(\{q_0, q_1\}, b) = \{q_2\};$$

$$\delta_1(\{q_2\}, a) = \{q_0, q_1, q_3\};$$

$$\delta_1(\{q_0, q_1, q_3\}, b) = \{q_0, q_2\};$$

$$\delta_1(\{q_0, q_2\}, a) = \{q_0, q_3\}.$$

Результат работы алгоритма 3.25 показан на рисунке 3.55.

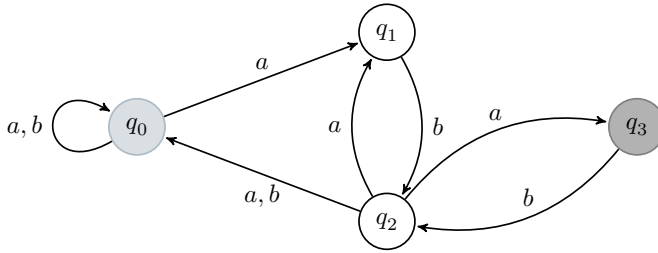


Рис. 3.54.

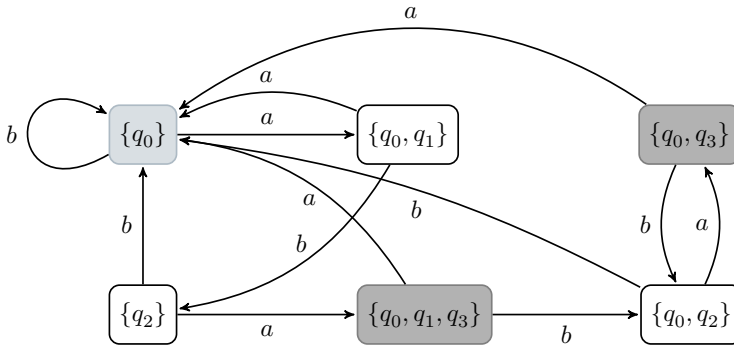


Рис. 3.55.

3.7.8. Теорема о разрастании для автоматных языков

Очевидно, что любой конечный язык может быть распознан конечным автоматом, поэтому *все конечные языки автоматные*. Один и тот же язык может распознаваться разными автоматами. Однако не для всех языков существуют распознающие их конечные автоматы. Иными словами, *существуют и неавтоматные языки*.

Установим некоторое необходимое условие, которому удовлетворяют все автоматные языки. После этого, проверив, что некоторый язык этому условию не удовлетворяет, можно заключить, что он не является автоматным. Очевидно, что это условие имеет смысл формулировать только для бесконечных языков.

Теорема 3.51 (о разрастании для автоматных языков). Пусть L — бесконечный автоматный язык. Тогда существует такая константа n , что любое слово $w \in L$ длиной $|w| > n$ можно разбить на три части $\alpha_1, \alpha_2, \alpha_3$ так, что $w = \alpha_1\alpha_2\alpha_3$ и

- 1) $|\alpha_1\alpha_2| \leq n$;
- 2) $|\alpha_2| > 0$;
- 3) $\forall m \geq 0 \quad w_m = \alpha_1\alpha_2^m\alpha_3 \in L$.

В последнем условии $\alpha_2^0 = \varepsilon$, $\alpha_2^1 = \alpha_2$, $\alpha_2^{i+1} = \alpha_2^i \alpha_2$.

Доказательство. Так как язык L автоматный, то существует детерминированный конечный автомат $A = (Q, \Sigma, \delta, q_0, F)$, распознающий L . Пусть $|Q| = n$ и слово $w = w_1 w_2 \dots w_k \in L$ имеет длину $k > n$. Рассмотрим путь

$$p = (q_0 = q_{i_0}, q_{i_1}, \dots, q_{i_k})$$

в диаграмме автомата A , который принимает слово w . Очевидно, что среди первых $(n + 1)$ состояний этого пути хотя бы одно встречается дважды. Выберем первое из таких состояний — $q \in Q$. Тогда для некоторой пары чисел $l < j \leq n$ имеем $q_{i_l} = q_{i_j} = q$.

Пусть $\alpha_1 = w_1 w_2 \dots w_l$ — это префикс w , который переводит q_0 в $q_{i_l} = q$, $\alpha_2 = w_{l+1} \dots w_j$ — это подслово w , которое переводит $q_{i_l} = q$ в $q_{i_j} = q$, и $\alpha_3 = w_{j+1} \dots w_k$ — это суффикс w , который переводит $q_{i_j} = q$ в $q_{i_k} \in F$.

Части α_1 и α_3 могут быть пустыми, но $|\alpha_2| = j - l \geq 1$. Длина $|\alpha_1 \alpha_2| = j \leq n$. Таким образом, условия (1) и (2) теоремы выполнены. Нетрудно убедиться и в выполнении условия (3).

Действительно, выбросив из пути p цикл $q_{i_l} = q, \dots, q_{i_j}$, получим путь p_0 из q_0 в $q_{i_k} \in F$, который принимает слово $\alpha_1 \alpha_3$, а повторив этот цикл m раз, получим путь p_0 из q_0 в $q_{i_k} \in F$, который принимает слово $\alpha_1 \alpha_2^m \alpha_3$. Следовательно,

$$\forall m \geq 0 \quad w_m = \alpha_1 \alpha_2^m \alpha_3 \in L.$$

■

Замечание 3.62

Содержательно теорема 3.51 утверждает, что у всякого достаточно длинного слова из автоматного языка имеется непустое подслово, которое можно вырезать или повторить сколько угодно раз, оставаясь внутри языка.

Пример 3.196. Рассмотрим автомат с диаграммой, представленной на рисунке 3.56. Проиллюстрируем теорему 3.51 для слова $w = 1101111$. Имеем $n = 4$, $|w| = 7 > 4$. Путь p для слова w :

$$p = q_0 \xrightarrow{1} q_2 \xrightarrow{1} \underbrace{q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_1}_{\text{цикл}} \xrightarrow{1} q_3 \xrightarrow{1} q_3 \xrightarrow{1} q_3.$$

Повторение состояний $q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_1$. Тогда

$$w = \alpha_1 \alpha_2 \alpha_3, \quad \alpha_1 = 11, \quad \alpha_2 = 01, \quad \alpha_3 = 111.$$

При этом

$$|\alpha_1 \alpha_2| = n = 4, \quad |\alpha_2| = 2.$$

Очевидно, что слово

$$\alpha_1 \alpha_2^m \alpha_3 \in L, \forall m \geq 0.$$

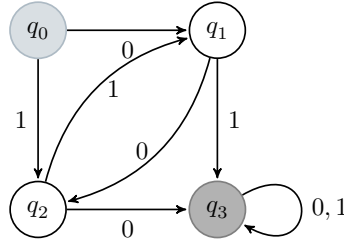


Рис. 3.56.

Замечание 3.63

Теорема 3.51 имеет другие названия в литературе: лемма о накачке, лемма Огдена, pumping lemma.

Как, используя теорему 3.51, доказать, что некоторый язык L не является автоматным? Это можно сделать, используя схему доказательства от противного.

1. Предположим, что L — автоматный язык. Тогда для него имеется константа n из утверждения теоремы 3.51.

2. Определим по n «специальное» слово $w \in L$, $|w| > n$ и докажем, что для любого разбиения $w = \alpha_1 \alpha_2 \alpha_3$, удовлетворяющего условиям (1) и (2) теоремы, найдется такое $k \geq 0$, что слово $w_k = \alpha_1 \alpha_2^k \alpha_3 \notin L$.

3. На основании полученного противоречия делаем вывод, что L не является автоматным языком.

Замечание 3.64

В этой схеме самым сложным является выбор «специального» слова w в пункте (2). Что касается подбора такого $k \geq 0$, для которого $w_k \notin L$, то, как правило, достаточно рассмотреть $k = 0$ или $k = 2$.

Рассмотрим несколько примеров применения данной схемы.

Пример 3.197. Покажем, что язык $L_1 = \{w = 0^k 1^k \mid k \geq 1\}$ не является автоматным.

Предположим, что он автоматный. Тогда для него имеется n из утверждения теоремы 3.51. Рассмотрим «специальное» слово $w = 0^n 1^n$. Очевидно, что $w \in L_1$. Предположим, что существует разбиение $w = \alpha_1 \alpha_2 \alpha_3$, удовлетворяющее условиям (1) и (2) теоремы.

Так как по условию (2) $|\alpha_1\alpha_2| \leq n$, то $\alpha_2 = 0^i$ для некоторого $i > 0$. Но тогда слово $w_0 = \alpha_1\alpha_3 = 0^{n-i}1^n \notin L_1$, что противоречит условию (3) теоремы. Следовательно, язык L_1 не является автоматным.

Пример 3.198. Покажем, что язык правильных скобочных последовательностей L_2 в алфавите $\{(,)\}$ не является автоматным.

В качестве «специального» слова выберем слово $w = ({}^n) \in L_2$. Тогда для всякого разбиения $w = \alpha_1\alpha_2\alpha_3$ такого, что $|\alpha_1\alpha_2| \leq n$, слово $\alpha_2 = ({}^i$ для некоторого $i > 0$. И, как и в предыдущем примере 3.197, слово $w_0 = \alpha_1\alpha_3 = ({}^{n-i}) \notin L_2$, что противоречит условию (3) теоремы. Следовательно, язык L_2 не автоматный.

Упражнение 3.28. С помощью теоремы 3.51 покажите, что язык

$$L_3 = \{w = 0^k 1 0^k \mid k \geq 1\}$$

не является автоматным.

3.7.9. Автоматы и автоматные грамматики

Установим связь между автоматными грамматиками G_3 и недетерминированными автоматами-распознавателями.

Определение 3.138. Порождающая грамматика G и конечный автомат-распознаватель A называются *эквивалентными*, если $L(G) = L(A)$.

Теорема 3.52. Для каждой праволинейной грамматики существует эквивалентный ей конечный автомат.

Доказательство. Каждому нетерминальному символу произвольной праволинейной грамматики G поставим в соответствие одно состояние конечного автомата A . Добавим еще одно состояние T — единственное конечное состояние. Состояние, соответствующее аксиоме грамматики S , будем считать начальным состоянием. Каждому правилу $A \rightarrow aB$ поставим в соответствие команду $Aa \rightarrow B$, а каждому терминальному правилу $A \rightarrow a$ поставим в соответствие команду $Aa \rightarrow T$. Таким образом, выводу цепочки в грамматике

$$S \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \dots \Rightarrow a_1 a_2 \dots a_{k-1} A_{k-1} \Rightarrow a_1 a_2 \dots a_k$$

взаимно однозначно соответствует последовательность команд построенного автомата A :

$$Aa_1 \rightarrow A_1; A_1 a_2 \rightarrow A_2; \dots; A_{k-1} a_k \rightarrow T.$$

Таким образом, язык $L(G) = L(A)$. ■

Пример 3.199. Для грамматики G , заданной следующими правилами вывода:

$$S \rightarrow aS|bB; \quad A \rightarrow aA|bS; \quad B \rightarrow bB|cA,$$

построим эквивалентный ей конечный автомат A .

Граф автомата (рисунок 3.57) будет иметь четыре вершины, три из них помечены нетерминальными символами грамматики S, A, B , четвертая вершина, помеченная символом T , является единственным заключительным состоянием. Начальным состоянием является вершина, соответствующая аксиоме грамматики S .

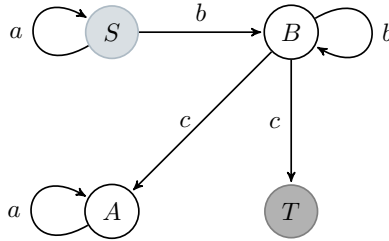


Рис. 3.57.

Согласно теореме 3.52 каждому правилу грамматики поставим в соответствие команду конечного автомата:

- 1) $Sa \rightarrow S$ — в начальном состоянии при поступлении на вход терминального символа a автомат остается в том же состоянии S ;
- 2) $Sb \rightarrow B$ — из начального состояния при поступлении на вход терминального символа b автомат переходит в состояние B ;
- 3) $Bb \rightarrow B$ — в состоянии B при поступлении на вход терминального символа b автомат остается в том же состоянии B ;
- 4) $Bc \rightarrow T$ — из состояния B при поступлении на вход терминального символа c автомат переходит в заключительное состояние T ;
- 5) $Bc \rightarrow A$ — из состояния B при поступлении на вход терминального символа c автомат переходит в состояние A ;
- 6) $Aa \rightarrow A$ — в состоянии A при поступлении на вход терминального символа a автомат остается в этом же состоянии A ;
- 7) $Ab \rightarrow S$ — из состояния A при поступлении на вход терминального символа b автомат переходит в состояние S .

Полученный *недетерминированный* конечный автомат распознает цепочки языка, порождаемые праворекурсивной грамматикой G .

Пример 3.200. Для грамматики G_4 из примера 3.163 построим эквивалентный ей конечный автомат A_4 . Правила вывода для этой грамматики согласно (3.90) имеют вид:

$$S \rightarrow aA, \quad A \rightarrow aA \mid bC, \quad C \rightarrow cC \mid c.$$

Грамматика G_4 является регулярной (праволинейной) грамматикой и порождает язык $L(G_4) = \{a^n b c^m \mid n, m > 0\}$.

Согласно теореме 3.52 каждому правилу грамматики G_4 поставим в соответствие команду конечного автомата. Рассуждая аналогично примеру 3.199, получаем следующий автомат (рисунок 3.58).

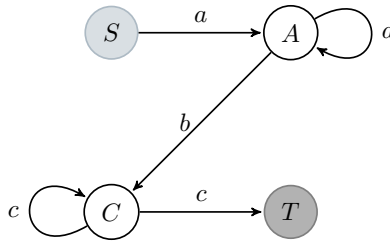


Рис. 3.58.

Полученный *недетерминированный* конечный автомат распознает цепочки языка, порождаемые праворекурсивной грамматикой G_4 , например слово $aaabcc$ из примера 3.163.

Пример 3.201. Построим конечный автомат, распознающий язык

$$L(A) = \{(ab)^n \mid n \in \mathbb{N}\}.$$

Сначала построим саму грамматику G , которая порождает язык $L(A)$:

$$S \rightarrow aA; \quad A \rightarrow bS \mid b.$$

Проверим, действительно ли эта грамматика порождает язык $L(A)$. Для этого построим несколько выводов возможных вариантов цепочек:

- 1) $S \Rightarrow aA \Rightarrow ab$;
- 2) $S \Rightarrow aA \Rightarrow abS \Rightarrow abaA \Rightarrow abab$;
- 3) $S \Rightarrow aA \Rightarrow abS \Rightarrow abaA \Rightarrow ababS \Rightarrow ababaA \Rightarrow ababab$.

и т.д.

Таким образом, грамматика G действительно порождает язык $L(A)$, следовательно, можно построить соответствующий этой грамматике конеч-

ный автомат (рисунок 3.59). Для этого введем заключительное состояние T , начальное состояние соответствует аксиоме грамматики S .

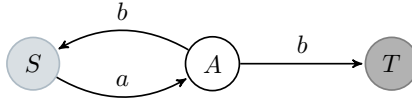


Рис. 3.59.

Запишем преобразование правил вывода грамматики G в команды автомата A :

- 1) $Sa \rightarrow A$ — из состояния S при поступлении на вход терминального символа a автомат переходит в состояние A ;
- 2) $Ab \rightarrow S$ — из состояния A при поступлении на вход терминального символа b автомат переходит в состояние S ;
- 3) $Ab \rightarrow F$ — из состояния A при поступлении на вход терминального символа b автомат переходит в заключительное состояние T .

Таким образом, построенный недетерминированный конечный автомат A распознает заданный язык $L(G)$.

Легко установить и утверждение, обратное к теореме 3.52.

Теорема 3.53. Для произвольного конечного автомата-распознавателя существует эквивалентная порождающая автоматная праволинейная грамматика.

Доказательство. Каждому состоянию произвольного автомата поставим в соответствие нетерминальный символ грамматики, причем начальному состоянию будет соответствовать аксиома грамматики. Тогда для каждой команды $Ac \rightarrow B$ в множество правил грамматики включим правило $A \rightarrow cB$, причем в случае, если B — заключительное состояние, добавим правило $A \rightarrow c$. Эквивалентность исходного конечного автомата и построенной грамматики очевидна. ■

Пример 3.202. Построим порождающую автоматную праволинейную грамматику по недетерминированному конечному автомату-распознавателю на рисунке 3.60.

Введя согласно теореме 3.53 обозначения для порождающей грамматики G , получаем набор правил:

$$q_0 \rightarrow S, q_1 \rightarrow A, \quad G: S \rightarrow aS \mid aA, A \rightarrow bA \mid b.$$

Построенная грамматика G порождает язык

$$L(G) = \{a^n b^m \mid n \geq 0, m > 0\}.$$

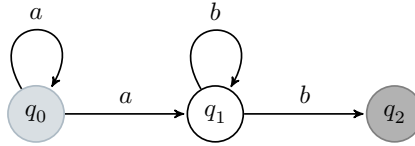


Рис. 3.60.

3.7.10. Минимизация автоматов-распознавателей

Поставим задачу нахождения наименьшего (по количеству состояний) конечного автомата, распознающего данный язык. Введем необходимые определения.

Определение 3.139. На множестве состояний Q автомата A зададим семейство отношений эквивалентности следующим образом:

а) *0-эквивалентность*: для произвольных состояний $q_1, q_2 \in Q$ полагаем $q_1 \stackrel{0}{\equiv} q_2$ тогда и только тогда, когда они оба являются заключительными или оба не являются заключительными;

б) *n-эквивалентность*: при $n \geq 1$ полагаем $q_1 \stackrel{n}{\equiv} q_2$ тогда и только тогда, когда

$$q_1 \stackrel{n-1}{\equiv} q_2, \quad \delta(q_1, \alpha) \stackrel{n-1}{\equiv} \delta(q_2, \alpha), \quad \forall \alpha \in \Sigma. \quad (3.98)$$

Замечание 3.65

Предлагаемый ниже алгоритм 3.26 нахождения минимального автомата был предложен Ауфенкампом и Хоном в [70].

Алгоритм 3.26. Построение минимального автомата

// Инициализация

Удаляем все недостижимые состояния, используя алгоритм 3.23.

Разбиваем множество состояний Q на два подмножества 0-эквивалентных состояний: $S_1 = F$ и $S_2 = Q - F$.

$n \leftarrow 0$

// Основной цикл

do

Разбиваем каждое из подмножеств n -эквивалентных состояний на подмножества $(n + 1)$ -эквивалентных состояний.

$n \leftarrow n + 1$

while пока удается разбить хотя бы одно подмножество

// Получаем набор подмножеств эквивалентных состояний S_1, \dots, S_k

Для множества состояний минимизированного автомата берем по одному представителю каждого из множеств S_i .

При работе алгоритма 3.26 возможны два крайних случая:

1) все состояния исходного автомата окажутся эквивалентными, и тогда в минимальном автомате останется только одно состояние; этот случай проиллюстрирован на рисунке 3.61а (исходный автомат) и б (минимизированный автомат) — это автомат, допускающий произвольные цепочки символов a, b ;

2) итоговое разбиение будет состоять из одноэлементных классов эквивалентности, это означает, что исходный автомат нельзя минимизировать, он уже минимален.

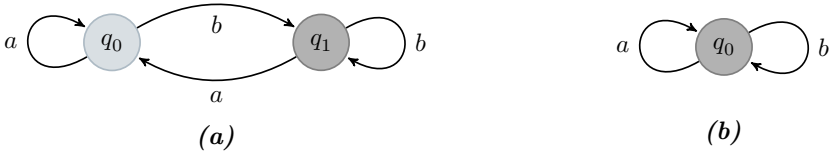


Рис. 3.61.

Замечание 3.66

Алгоритм 3.26 часто называют методом разбиения, а сам процесс приведения автомата к минимальному — редукцией конечного автомата.

Пример 3.203. Рассмотрим процесс минимизации автомата, диаграмма состояний которого представлена на рисунке 3.62.

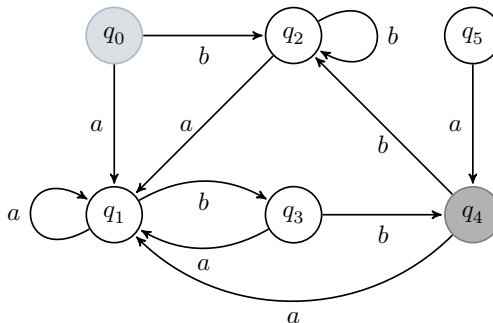


Рис. 3.62.

1 Согласно алгоритму 3.23 вначале произведем поиск и удаление недостижимых состояний. Получаем список достижимых состояний:

$$\{q_0\} \rightarrow \{q_0, q_2\} \rightarrow \{q_0, q_2, q_1\} \rightarrow \{q_0, q_2, q_1, q_3\} \rightarrow \{q_0, q_2, q_1, q_3, q_4\}.$$

Состояние q_5 недостижимо, удаляем его вместе с дугой $[q_5, q_4]$ и производим разбиение множества достижимых состояний автомата на классы эквивалентности.

2 Запишем начальное разбиение множества состояний автомата для отношения $\stackrel{0}{\equiv}$:

$$\{q_4\}, \{q_0, q_1, q_2, q_3\}.$$

3 Заметим, что для состояний q_0, q_1, q_2 автомат переходит в одно из состояний класса эквивалентности предыдущего уровня — $\{q_0, q_1, q_2, q_3\}$, поэтому все они 1-эквивалентны.

Но состояние $\delta(q_3, b) = q_4 \notin \{q_0, q_1, q_2, q_3\}$, поэтому согласно (3.98) состояние q_3 не 1-эквивалентно состояниям q_0, q_1, q_2 . В разбиении для 1-эквивалентности они «разойдутся» по разным классам; разбиение, определяемое отношением $\stackrel{1}{\equiv}$, будет иметь вид:

$$\{q_4\}, \{q_0, q_1, q_2\}, \{q_3\}.$$

4 Далее, для состояний q_0 и q_2 автомат переходит в одно из состояний класса эквивалентности предыдущего уровня $\{q_0, q_1, q_2\}$, поэтому они 2-эквивалентны.

Однако состояние $\delta(q_1, b) = q_3 \notin \{q_0, q_1, q_2\}$, поэтому q_1 не 2-эквивалентно q_0 и q_2 . Произойдет разделение; разбиение, определяемое отношением $\stackrel{2}{\equiv}$, будет иметь вид:

$$\{q_4\}, \{q_0, q_2\}, \{q_1\}, \{q_3\}.$$

5 Заметим, что согласно (3.98) $q_0 \stackrel{3}{\equiv} q_2$. Действительно,

$$q_0, q_2, \delta(q_0, b), \delta(q_2, b) \in \{q_0, q_2\}, \quad \delta(q_0, a), \delta(q_2, a) \in \{q_1\}.$$

Поэтому разбиение на классы 2-эквивалентности и есть искомое разбиение. Таким образом, состояния q_0 и q_2 неразличимы. Полученный минимизированный автомат представлен на рисунке 3.63.

Пример 3.204. Проведем минимизацию автомата, диаграмма состояний которого представлена на рисунке 3.64.

1 Используя алгоритм 3.23, удаляем недостижимые состояния: q_6, q_5 .

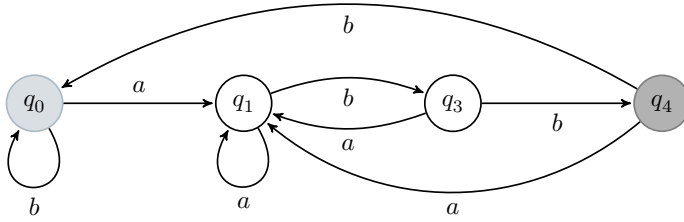


Рис. 3.63.

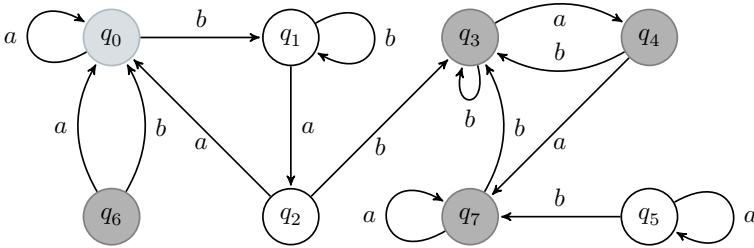


Рис. 3.64.

2 Определяем начальное разбиение множества состояний автомата для отношения \equiv^0 :

$$\{q_0, q_1, q_2\}, \{q_3, q_4, q_7\}.$$

3 Для состояний q_0 и q_1 автомат переходит в одно из состояний класса эквивалентности предыдущего уровня — $\{q_0, q_1, q_2\}$, поэтому они 1-эквивалентны. Но состояние $\delta(q_2, b) = q_3 \notin \{q_0, q_1, q_2\}$, следовательно, q_3 не 1-эквивалентно q_0 и q_1 . Для отношения \equiv^1 получаем новое разбиение:

$$\{q_0, q_1\}, \{q_2\}, \{q_3, q_4, q_7\}.$$

4 Далее, состояния $\delta(q_0, a) = q_0$ и $\delta(q_1, a) = q_2$ не 1-эквивалентны, для отношения \equiv^2 получаем разбиение:

$$\{q_0\}, \{q_1\}, \{q_2\}, \{q_3, q_4, q_7\}.$$

5 Поскольку для всех состояний из множества $\{q_3, q_4, q_7\}$ автомат переходит в одно из этих же состояний, то разбиение на классы 2-эквивалентности и есть искомое разбиение. Таким образом, состояния q_3, q_4, q_7 неразличимы. Минимизированный автомат представлен на рисунке 3.65.

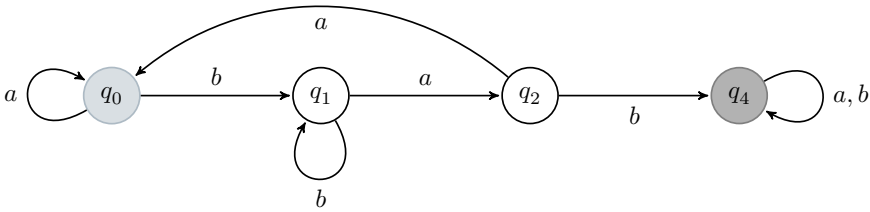


Рис. 3.65.

3.7.11. Конечные автоматы с эpsilon-переходами

Определение 3.140. Эpsilon-переходом называется переход между состояниями, который может быть выполнен автоматом «просто так», без входного символа. На графах и в таблицах такие переходы обычно помечаются символом ϵ .

Одно из наиболее полезных свойств ϵ -переходов — возможность простого объединения нескольких автоматов в один. При этом если на вход подаются детерминированные автоматы, то при объединении может получиться недетерминированный автомат. Рассмотрим этот прием на простом примере.

Пример 3.205. На рисунке 3.66 изображены два исходных конечных автомата.



Рис. 3.66.

На рисунке 3.67 показан результат объединения двух входных автоматов.

Никаких дополнительных ограничений на ϵ -переходы при работе конечного автомата не накладывается.

1. Из одного состояния недетерминированного конечного автомата может выходить сколько угодно как обычных, так и ϵ -переходов.

2. Может существовать цепочка из нескольких последовательных эpsilon-переходов.

3. Автомат может проходить цепочку целиком, частично или не проходить ее вообще (если у автомата есть другие варианты поведения).

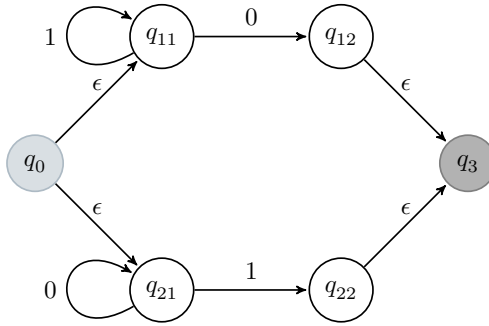


Рис. 3.67.

В общем, граф/таблица переходов автомата выглядят так, как будто во входном алфавите появился еще один символ — ϵ , а его работа так, будто в любом месте обрабатываемой цепочки (в начале, в конце, между символами) находится произвольное количество этих символов. При этом для каждого состояния есть как минимум один переход по ϵ — в себя.

Замечание 3.67

Нетрудно показать, что по недетерминированному автомату с ϵ -переходами можно построить эквивалентный ему конечный детерминированный автомат, но проектировать ϵ -автомат гораздо проще и удобнее.

Действительно, уберем ϵ -переходы следующим образом. Найдем все пары состояний (q_k, q_l) такие, что q_l достижимо из q_k по ϵ -переходам:

$$q_k \xrightarrow{\epsilon} \dots q_i \xrightarrow{\epsilon} \dots \xrightarrow{\epsilon} q_l$$

Для каждого не- ϵ -перехода $\delta(q_l, \alpha) = q_s$ добавим переход $\delta(q_k, \alpha) = q_s$. Кроме того, если $q_l \in F$, добавим и q_k в F . Далее, при необходимости нужно применить один из рассмотренных выше алгоритмов детерминизации (3.24 или 3.25).

Пример 3.206. Уберем ϵ -переходы для объединенного автомата из примера 3.205 (рисунок 3.67). На рисунке 3.68 представлен результат. Детерминизация не потребовалась.

3.7.12. Конечные автоматы и регулярные выражения

Регулярные выражения являются достаточно удобным средством для построения «алгебраических» описаний языков. Они строятся из элементарных выражений с помощью операций *объединения* $(+, |, \cup)^1$, *конкатенации*

¹ В различных описаниях возможно использование одного из данных символов.

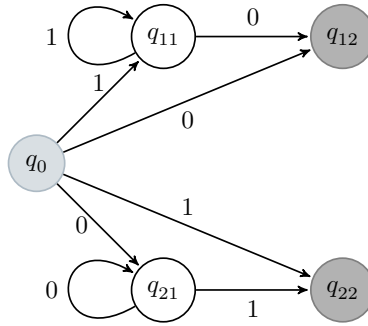


Рис. 3.68.

(сцепления) $()$ и итерации $(*)$. Каждому такому выражению r соответствует представляемый им язык L_r .

Замечание 3.68

Другие общеупотребительные названия регулярных выражений: *Regular Expressions*, *RegExp*. Активно используются в языках программирования *Python*, *Perl*, *Ruby*, *Java*, *.Net* и в текстовых редакторах *Vim*, *EmEdit*.

Введем необходимые определения.

Определение 3.141. Объединение языков $L_1 + L_2$ содержит все слова, содержащиеся или в L_1 , или в L_2 .

Конкатенация (сцепление) L_1L_2 содержит все слова, удовлетворяющие форме vw , где $v \in L_1$, а $w \in L_2$.

Итерация¹ $(L)^*$ языка L образуют все слова, которые можно разбить на несколько подряд идущих слов из L :

$$(L)^* = \{\varepsilon\} \cup \{w \mid (\exists n \geq 0)(w = w_1w_2 \dots w_n), w_i \in L, i \in 1 : n\}.$$

Ее можно представить с помощью степеней:

$$(L)^* = \bigcup_{i=0}^{\infty} L^i.$$

Заметим, что $(L)^*$ включает и пустое слово ε , так как $n = 0$ допустимо по условию.

Введем некоторые соглашения. При записи регулярных выражений будем опускать знак конкатенации $\hat{\cdot}$. Расставим приоритеты операций по уменьшению: итерация, конкатенация, объединение. Это позволит опустить многие скобки.

¹ Иногда ее называют замыканием Клини.

Пример 3.207. Выражение $((1^0)^{(1)^* + 0})$ можно записать как $10(1^* + 0)$.

Дадим строгое определение регулярного выражения. Регулярное выражение над алфавитом Σ определяется рекурсивно следующим образом.

Определение 3.142.

- 1) Пустой символ ϵ является регулярным выражением;
- 2) для любого $a \in \Sigma$, a является регулярным выражением;
- 3) если r и p являются регулярными выражениями, то $(r + p)$, (rp) и r^* тоже являются регулярными выражениями.

Определение 3.143. Два регулярных выражения r и p называются *эквивалентными*, если совпадают представляемые ими языки, т. е. $L_r = L_p$. В этом случае будем писать $r = p$.

Теорема 3.54. Справедливы следующие свойства регулярных операций:

- 1) $r + p = p + r$ (коммутативность объединения);
- 2) $(r + p) + q = r + (p + q)$ (ассоциативность объединения);
- 3) $(rp)q = r(pq)$ (ассоциативность конкатенации);
- 4) $(r^*)^* = r^*$ (идемпотентность итерации);
- 5) $(r + p)q = rq + pq$ (дистрибутивность).

Доказательство. Перечисленные свойства очевидным образом следуют из введенных выше определений. ■

Следствие 3.11. Класс регулярных языков замкнут относительно операций конкатенации и итерации.

Рассмотрим примеры регулярных выражений и представляемых ими языков.

1. Регулярное выражение $(0 + 1)^*$ представляет множество всех слов в алфавите $\{0, 1\}$, включая пустое слово.

2. Множество всех слов в алфавите $\{0, 1\}$, содержащих ровно одно вхождение 1, соответствует регулярному выражению 0^*10^* .

3. Регулярное выражение $11(0 + 1)^*001$ представляет язык, состоящий из всех слов в алфавите $\{0, 1\}$, которые начинаются на 11, а заканчиваются на 001.

Теорема 3.55. Для каждого регулярного выражения r можно эффективно построить такой недетерминированный конечный автомат A , который распознает язык, задаваемый r , т. е. $L_A = L_r$.

Доказательство этой теоремы достаточно техническое, построение автомата A по выражению r проводится индукцией по длине регулярного выражения r .

Следствие 3.12. Для каждого регулярного выражения можно эффективно построить детерминированный конечный автомат, который распознает язык, представляемый этим выражением.

Рассмотрим алгоритм построения недетерминированного конечного автомата на основе регулярного выражения. Используем автоматы с эпсилон-переходами, рассмотренными в разделе 3.7.11.

Будем строить недетерминированный конечный автомат (НКА) как композицию таких автоматов для более простых выражений исходя из определения 3.142. Комбинирование выполняется по правилам, представленным ниже.

1 Правило 1. Для ϵ строим НКА, i — новое начальное состояние, f — новое заключительное состояние. Этот НКА (рисунок 3.69a) распознает язык $L = \{\epsilon\}$.

2 Правило 2. Для $a \in \Sigma$ строим НКА, i — новое начальное состояние, f — новое заключительное состояние. Этот НКА (рисунок 3.69b) распознает язык $L = \{a\}$.



Рис. 3.69.

3 Правило 3. Пусть A_s и A_t представляют собой НКА для регулярных выражений s и t . Для регулярного выражения $s + t$ строим НКА A_{s+t} , i — новое начальное состояние, f — новое заключительное состояние. Этот НКА (рисунок 3.70a) распознает язык $L_s \cup L_t$.

4 Правило 4. Для регулярного выражения st строим НКА A_{st} . Здесь начальное состояние НКА A_s становится новым начальным состоянием A_{st} , заключительное состояние A_t становится заключительным состоянием A_{st} . А конечное состояние A_s и начальное состояние A_t объединяются. Этот НКА (рисунок 3.70b) распознает язык $L_s L_t$.

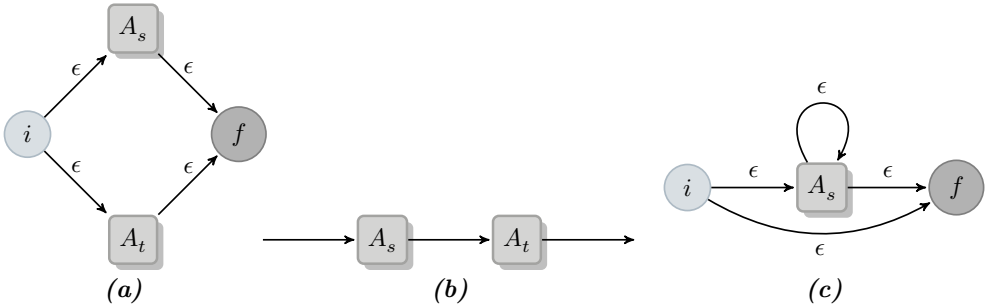


Рис. 3.70.

5 Правило 5. Для регулярного выражения s^* строим НКА A_{s^*} , i — новое начальное состояние, f — новое заключительное состояние. Этот НКА (рисунок 3.70с) распознает язык $L(s^*)$.

Пример 3.208. Используя перечисленные правила, построим НКА по регулярному выражению $r = (ab)^* a^* ba$.

Используя правила 2, 4 и 5, строим НКА A_r (рисунок 3.71) для регулярного выражения $r = (ab)^*$.

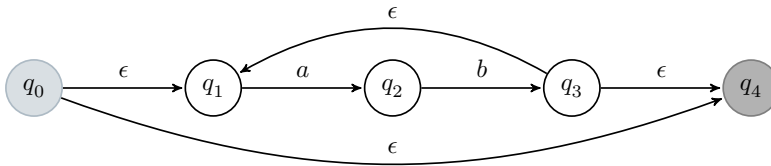


Рис. 3.71.

Используя правила 2 и 5, строим НКА A_s (рисунок 3.72) для регулярного выражения $s = a^*$.

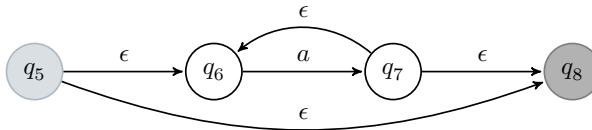


Рис. 3.72.

Используя правила 2 и 4, строим НКА A_t (рисунок 3.73) для регулярного выражения $t = ba$.

Соединяем построенные автоматы A_r , A_s и A_t (рисунок 3.74).

Пример 3.209. Построим детерминированный конечный автомат для регулярного выражения $(a + b)^* a(b + a)$.

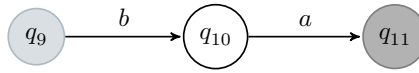


Рис. 3.73.

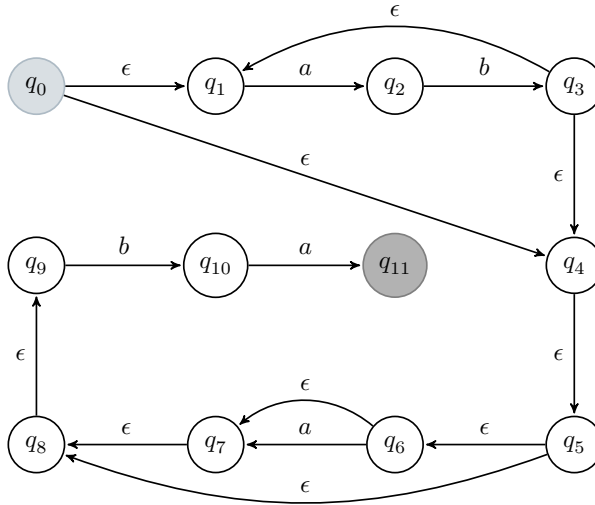


Рис. 3.74.

Построим сначала недетерминированный автомат (рисунок 3.75a), затем детерминизируем его. В данном случае при построении можно обойтись без эпсилон-переходов.

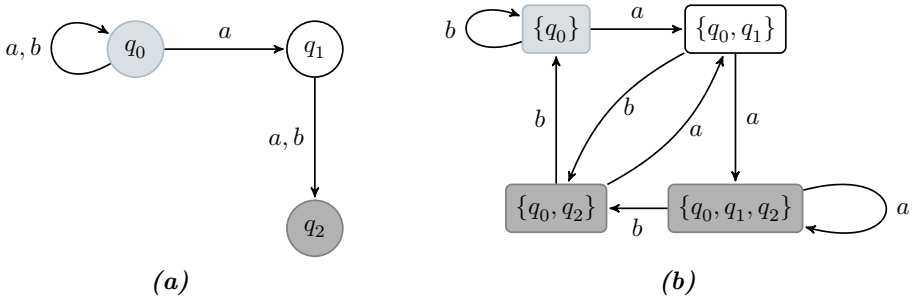


Рис. 3.75.

Применяя алгоритм 3.25 детерминизации вытягиванием, получаем:

$$\begin{array}{ll}
 \delta_1(\{q_0\}, a) = \{q_0, q_1\}; & \delta_1(\{q_0\}, b) = \{q_0\}; \\
 \delta_1(\{q_0, q_1\}, a) = \{q_0, q_1, q_2\}; & \delta_1(\{q_0, q_1\}, b) = \{q_0, q_2\}; \\
 \delta_1(\{q_0, q_2\}, a) = \{q_0, q_1\}; & \delta_1(\{q_0, q_2\}, b) = \{q_0\}; \\
 \delta_1(\{q_0, q_1, q_2\}, a) = \{q_0, q_1, q_2\}; & \delta_1(\{q_0, q_1, q_2\}, b) = \{q_0, q_3\};
 \end{array}$$

$$\delta_1(\{q_0, q_3\}, a) = \{q_0, q_1\}; \quad \delta_1(\{q_0, q_3\}, b) = \{q_0\}.$$

Полученный детерминированный автомат представлен на рисунке 3.75*b*.

Задачи и вопросы

1. Дайте определение автоматов Мили и Мура. В чем их различие?
2. Как преобразовать автомат Мура в эквивалентный ему автомат Мили?
3. Дайте определение недетерминированного конечного автомата-распознавателя.
4. Опишите алгоритм построения недетерминированного конечного автомата по автоматной грамматике.
5. Постройте конечный автомат Мили, который для заданных двух целых двоичных чисел N_1 и N_2 одинаковой длины (меньшее может быть дополнено слева нулями) выдает $\max(X_1, X_2)$. Числа вводятся со старших разрядов, вход автомата — пары разрядов исходных чисел.
6. Приведите пример конечного автомата с ϵ -переходами. Для чего нужны такие автоматы?
7. Постройте конечный автомат, распознающий язык L в алфавите $\{a, b, c\}$, где слово принадлежит языку L , если в нем не встречается буква a .
8. Постройте конечный автомат, распознающий язык

$$L = \{ucv \mid u \in \{a, b\}^*, v \in \{a, b\}^*\}.$$

9. Постройте конечный автомат с входным алфавитом $\{0, 1\}$, допускающий цепочки:
 - а) число единиц четное, а нулей — нечетное;
 - б) между вхождениями единиц четное число нулей;
 - в) за каждым вхождением пары 11 следует 0;
 - г) каждый третий символ — единица;
 - д) имеется по крайней мере одна единица;
 - е) начинаются с 0 и заканчиваются на 1.
10. Постройте конечный автомат с входным алфавитом $\Sigma = \{a, b, c\}$, допускающий все цепочки:

- а) в которых две последние буквы не совпадают;
- б) начинающиеся и заканчивающиеся различными символами;
- в) которые содержат символ b только парами и заканчиваются на c ;
- г) начинающиеся с b , и в цепочке встречается только один раз ac ;
- д) у которых последний символ цепочки уже появлялся раньше.

11. Постройте конечный автомат с входным алфавитом $\Sigma = \{a, b\}$, распознающий язык L :

- а) в каждом слове которого сочетание ab встречается не более двух раз;
- б) в каждом слове которого содержится подслово $bbcc$;
- в) каждое слово которого имеет длину не более 8 и содержит одинаковое число букв a и b .

12. Минимизируйте конечный детерминированный автомат, представленный на рисунке 3.76.

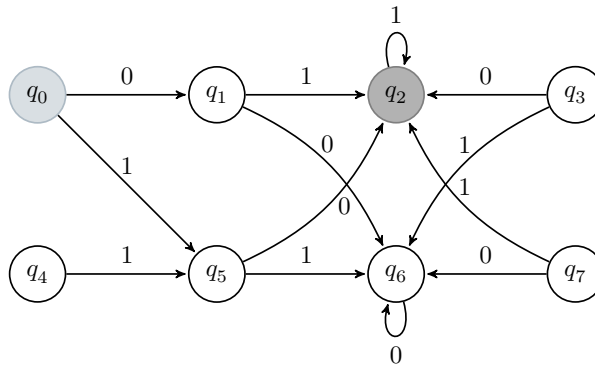


Рис. 3.76.

13. Постройте конечный автомат, распознающий зарезервированные слова языка C++:

- а) *continue*, *class*, *const*;
- б) *private*, *protected*, *public*;
- в) *union*, *using*, *union*;
- г) *double*, *delete*, *default*.

14. Для грамматик, заданных следующими правилами вывода, постройте эквивалентные им конечные автоматы:

- а) $S \rightarrow 0A$; $A \rightarrow 0A|1S|0$;

$$\text{б) } S \rightarrow 0C|0D; C \rightarrow 0D|0S|1; D \rightarrow 1C|1S|0;$$

$$\text{в) } S \rightarrow bS|aA|a|b; A \rightarrow aA|bS|a;$$

$$\text{г) } S \rightarrow aA|bB|aC; A \rightarrow bA|bB|c; B \rightarrow aA|cC|b; C \rightarrow bB|bC|a.$$

15. Для недетерминированного конечного автомата A :

$$A = (Q, \Sigma, \delta, q_0, F), \quad Q = \{q_0, q_1, q_2, q_3\}, \quad \Sigma = \{7, 8, 9\}, \quad F = \{q_3\},$$

$$\left\{ \begin{array}{l} \delta(q_0, 7) = \{q_0, q_1\}, \\ \delta(q_1, 8) = \{q_1, q_2\}, \\ \delta(q_2, 9) = \{q_3\}, \\ \delta(q_3, 9) = \{q_3\} \end{array} \right.$$

постройте эквивалентный ему детерминированный конечный автомат.

16. Дан детерминированный конечный автомат A :

$$A = (Q, \Sigma, \delta, q_0, F), \quad Q = \{q_0, q_1, q_2, q_3\}, \quad \Sigma = \{0, 1\}, \quad F = \{q_3\},$$

$$\left\{ \begin{array}{l} \delta(q_0, 1) = q_0, \\ \delta(q_0, 0) = q_1, \\ \delta(q_1, 1) = q_0, \\ \delta(q_1, 0) = q_2, \\ \delta(q_2, 1) = q_0, \\ \delta(q_2, 0) = q_3, \\ \delta(q_3, 1) = q_3, \\ \delta(q_3, 0) = q_3. \end{array} \right.$$

Применив теорему 3.51 о разрастании для автоматных языков, постройте представление слова $w = 0010001$ в виде $z = \alpha_1\alpha_2\alpha_3$.

17. Применив теорему 3.51 о разрастании для автоматных языков, покажите нерегулярность следующих языков:

$$\text{а) } \{0^n 1^m 2^n \mid n, m \in N\};$$

$$\text{б) } \{0^n 1^m \mid n \leq m, n, m \in N\};$$

$$\text{в) } \{0^n 1^{2n} \mid n \in N\};$$

г) множество цепочек из нулей и единиц вида ww , где некоторая цепочка w повторяется дважды.

18. Докажите, что язык над алфавитом $\Sigma = \{a, b\}$, содержащий только слова, в котором букв a больше, чем букв b , невозможно задать никаким конечным автоматом.

19. По заданному регулярному выражению постройте диаграмму переходов конечного автомата. Если конечный автомат является недетерминированным, то преобразуйте его в детерминированный. Если конечный автомат не является минимальным, то минимизируйте его.

а) $a^2(ab)^*c|d$;

б) $(d|a)^2(ab)^*c$;

в) $c^+((d|a)^2ab)^*$;

г) $(a|b|c)^+abc(a|b|c)^*$;

д) $a^*b^*c^*|a^+b^+c^+$.

СПИСОК ЛИТЕРАТУРЫ

1. Айзерман М. А., Гусев Л. А., Розоноэр Л. И., Смирнова И. М., Таль А. А. Логика. Автоматы. Алгоритмы. — М. : Гос. изд-во физ.-мат. лит., 1963.

2. Айерлэнд К., Роузен М. Классическое введение в современную теорию чисел. — М. : Мир, 1987.

3. Биркгоф Г. Теория решеток. — М. : Наука, 1984.

4. Блейхут Р. Теория и практика кодов, исправляющих ошибки. — М. : Мир, 1986.

5. Брауэр В. Введение в теорию конечных автоматов. — М. : Радио и связь, 1987.

6. Булос Дж., Джефффри Р. Вычислимость и логика. — М. : Мир, 1994.

7. Бухштаб А. А. Теория чисел. — М. : Просвещение, 1966.

8. Cobham A. The intrinsic computational difficulty of functions // In Proceedings of the 1964 Congress for Logic, Methodology, and the Philosophy of Science. — North-Holland, 1964. — P. 24–30.

9. Edmonds J. Paths, trees and flowers // Canadian Journal of Mathematics. — 1965. — Vol. 17. — P. 449–467.

10. Введение в криптографию / под ред. Яценко В. В. — М. : МЦНМО, 1998.

11. *Виноградов И. М.* Основы теории чисел. — М. : Наука, 1972.
12. *Гилл А.* Введение в теорию конечных автоматов. — М. : Наука, 1966.
13. *Харари Ф.* Теория графов. — 5-е изд., доп. — URSS, 2018.
14. *Горбатов В. А., Горбатов А. В., Горбатова М. В.* Теория автоматов. — М. : Высш. шк., 2008.
15. *Грэхем Р., Кнут Д., Паташник О.* Конкретная математика. — М. : Мир, 1998.
16. *Жегалкин И. И.* О технике вычислений предложений в символической логике. : мат.сб. — 1927. — С. 9–28.
17. *Заде Л. А.* Понятие лингвистической переменной и его применение к принятию приближенных решений. — М. : Мир, 1976.
18. *Карпов Ю. Г.* Теория автоматов. — СПб. : Питер, 2003.
19. *Касьянов В. Н.* Лекции по теории формальных языков, автоматов и сложности вычислений. — М. : Вильямс, 2002.
20. *Клини С.* Математическая логика. — М. : Мир, 1978.
21. *Клоксин У.* Программирование на языке Пролог / Клоксин У., Меллиш К. — М. : Мир, 1987.
22. *Кнут Д.* Искусство программирования для ЭВМ. Получисленные алгоритмы. : в 3 т. — М. : Мир, 1977. — Т. 2.
23. *Кнут Д.* Искусство программирования. Т.4, А: Комбинаторные алгоритмы. Ч. 1. — М. : Вильямс, 2018.
24. *Коньшева Л. К., Назаров Д. М.* Основы теории нечетких множеств. — СПб. : Питер, 2011.
25. *Кофман А.* Введение в теорию нечетких множеств. — М. : Радио и связь, 1982.
26. *Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л., Штайн К.* Алгоритмы: построение и анализ. — 3-е изд. — Вильямс, 2013.
27. *Кристофидес Н.* Теория графов. Алгоритмический подход. — М. : Мир, 1997.
28. *Кудрявцев В. Б., Алешин С. В., Подколзин А. С.* Введение в теорию автоматов. — М. : Наука, 1985.

29. *Кузнецов О. П., Адельсон-Вельский Г. М.* Дискретная математика для инженера. — М. : Энергоатомиздат, 1988.
30. *Лавров И. А., Максимова Л. Л.* Задачи по теории множеств, математической логике и теории алгоритмов. — М. : Наука, 1984.
31. *Леоненков А.В.* Нечеткое моделирование в среде MATLAB и fuzzyTECH. — СПб. : БХВ-Петербург, 2005.
32. *Липский В.* Комбинаторика для программистов. — М. : Мир, 1988.
33. *Лихтарников Л.М., Сукачева Т.Г.* Математическая логика. — М. : Лань, 1999
34. *Агафонова И. В., Малоземов В. Н.* Алгебраическая нормальная форма булевых функций и ее быстрое вычисление. Избранные главы дискретного гармонического анализа и геометрического моделирования /под ред. проф. Малоземова В. Н. — СПб. : Изд-во ВВМ, 2014. — Ч. 2. — С. 582–586.
35. *Мальцев А. И.* Алгоритмы и рекурсивные функции. — М. : Наука, 1965.
36. *Марков А. А., Нагорный Н. М.* Теория алгоритмов. — М. : ФАЗИСТ, 1996.
37. *Мелихов А. Н.* Ориентированные графы и конечные автоматы. — М. : Наука, 1971.
38. *Мендельсон Э.* Введение в математическую логику. — М. : Наука, 1984.
39. *Минский М.* Вычисления и автоматы. — М. : Мир, 1971.
40. *Могиленко А. В., Балусев А. В.* Элементарные понятия теории нечетких множеств. — Новосибирск, 2003.
41. *Мозговой М.* Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход. — М. : Наука и техника, 2006.
42. *Нефедов В. Н., Осипова В. А.* Курс дискретной математики. — М. : Изд-во МАИ, 1992.
43. *Фон Нейман Дж.* Вероятностная логика и синтез надежных организмов из ненадежных компонент, Автоматы /под ред. Шеннона К. Э. и Маккарти Дж. — М. : ИЛ, 1956.

44. *Новак В., Перфильева И., Мочкорж И.* Математические принципы нечеткой логики. — М. : Физматлит, 2006.
45. *Новиков Ф. А.* Дискретная математика для программистов. — 3-е изд. — СПб. : Питер, 2019.
46. *Оре О.* Теория графов. — М. : Наука, 1980.
47. *Пентус А. Е., Пентус М. Р.* Математическая теория формальных языков. — М. : БИНОМ, 2006
48. *Поликарпова Н. И., Шальто А. А.* Автоматное программирование. — СПб. : Питер, 2008.
49. *Поздняков С. Н., Рыбин С. В.* Дискретная математика. — М. : Академия, 2007.
50. *Романовский И. В.* Дискретный анализ. — СПб. : Невский диалект, 1999.
51. *Рыжов А. П.* Элементы теории нечетких множеств и ее приложений. — М. : Диалог, 2003.
52. *Штовба С. Д.* Введение в теорию нечетких множеств и нечеткую логику. — Винница : Континент-Прим, 2003.
53. *Смаллиан Р. М.* Принцесса или тигр? — М. : Мир, 1985.
54. *Супрун В. П.* Табличный метод полиномиального разложения булевых функций // Кибернетика. — 1987. — № 1. — С. 116–117.
55. *Супрун В. П.* Основы теории булевых функций. — М. : Лепанд / URSS, 2017. — 208 с.
56. *Стенли Р.* Перечислительная комбинаторика. — М. : Мир, 1990.
57. *Грахтенброт Б. А., Барздинь Я. М.* Конечные автоматы (поведение и синтез). — М. : Наука, 1970.
58. *Турчин В. Ф.* Алгоритмический язык рекурсивных функций (РЕФАЛ). — М. : ИПМ АН СССР, 1968.
59. *Уилсон Р.* Введение в теорию графов. — М. : Мир, 1977.
60. *Фудзисава Т., Касами Т.* Математика для радиоинженеров. Теория дискретных структур. — М. : Радио и связь, 1984.
61. *Холл М.* Комбинаторика. — М. : Мир, 1970.

62. Хомский Н. Синтаксические структуры // Новое в лингвистике. — М., 1962. Вып. II.
63. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. — 2-е изд. — М. : Вильямс, 2015.
64. Фляйшнер Г. Эйлеровы графы и смежные вопросы. — М. : Мир, 2002.
65. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. — М. : Мир, 1983.
66. Черч А. Введение в математическую логику. — М. : Изд-во иностр. лит., 1960.
67. Эндрюс Г. Теория разбиений. — М. : Наука, 1982.
68. Яблонский С. В. Введение в дискретную математику. — М. : Наука, 1997.
69. Omondi A., Premkumar B. Residue Number Systems: Theory and Implementation, 2007.
70. Aufenkamp D. D., Hohn F. E. Analysis of Sequential Machines // IRE Transactions on Electronic Computing. — Dec. 1957. — Vol. EC-6, I. 4. — P. 276–285.
71. Bron C., Kerbosh J. Algorithm 457 — Finding all cliques of an undirected graph. // Comm. of ACM — 1973. — Vol. 16. — P. 575–577.
72. De Bruijn N. G. A computational problem // Koninklijke Nederlandse Akademie v. Wetenschappen. — 1946. — Vol. — 49. — P. 758–764.
73. Diffie W., Hellman M. E. New Directions in Cryptography // IEEE Transactions on Information Theory. — 1976 — Vol. IT-22. — P. 644–654.
74. Euler L. Solutio problematis ad geometriam situs pertinentes // Implementation Academiae Petropolitanae. — 1736. — Vol. 8. — P. 128–140.
75. Fleury M. Deux problemes de geometrie de situation // Journal de mathematiques elementaires. — 1883.
76. Gardner M. A new kind of cipher that would take millions of years to break // Scientific American. — 1977. — P. 120–124.

-
77. *Van Lint J. H.* A Course in Combinatorics. — Cambridge University Press, 2001.
78. *Martelo S., Toth P.* Knapsack problems. — Great Britain — Wiley, 1990.
79. *Misra J., Gries D.* A constructive proof of Vizing's theorem // Information Processing Letters. — 1992. — Vol. 41, I. 3. — P. 131–133
80. *Simmons G. J., Norris M. J.* Preliminary comment on the M.I.T. public key cryptosystem // Cryptologia. — 1977.

СПИСОК АЛГОРИТМОВ

1.1. Решето Эратосфена	19
1.2. Метод пробных делителей	21
1.3. Метод Ферма	23
1.4. Вычисление квадратного корня в \mathbb{Z}	26
1.5. Сложение p -ичных чисел	29
1.6. Умножение p -ичного числа на цифру	30
1.7. Сдвиг p -ичного числа на k разрядов влево	30
1.8. Перемножение p -ичных чисел	30
1.9. Перевод из p -ичной записи в q -ичную в p -ичной арифметике	31
1.10. Перевод из p -ичной записи в q -ичную в q -ичной арифметике	32
1.11. Возведение числа в натуральную степень	32
1.12. Модификация алгоритма 1.11	33
1.13. Классический алгоритм Евклида	36
1.14. Бинарный алгоритм Евклида	38
1.15. Обобщенный алгоритм Евклида	41
1.16. Решение диофантового уравнения	63
1.17. Решение линейного сравнения	85
1.18. Сложение двух многочленов	107
1.19. Умножение многочленов	107
1.20. Первый алгоритм деления многочленов	108
1.21. Второй алгоритм деления многочленов	113
1.22. Разложение многочлена на свободные от квадратов множители	121
1.23. Редукция многочлена P по Q	137
1.24. Редукция многочлена P по базису Гребнера P_1, \dots, P_n	138

1.25. Алгоритм Бухбергера	140
1.26. Полиномиальное декодирование	169
1.27. Генерация k -элементных подмножеств	174
1.28. Генерация всех подмножеств n -элементного множества (1) .	176
1.29. Генерация всех подмножеств n -элементного множества (2) .	179
1.30. Генерация всех подмножеств n -элементного множества (3) .	180
1.31. Перечисление декартова произведения	181
1.32. Вычисление элементов набора (i_1, i_2, \dots, i_n)	181
1.33. Перечисление декартова произведения	182
1.34. Переход от одной перестановки к следующей	182
1.35. Генерация перестановок	186
1.36. Генерация разбиений	201
1.37. Задача о неограниченном рюкзаке	203
1.38. Задача о рюкзаке 0–1	204
2.1. Поиск в глубину в графе «DFS»	274
2.2. Поиск в ширину в графе «BFS»	276
2.3. Поиск компонент связности неориентированного графа . . .	279
2.4. Нахождение всех мостов графа	280
2.5. Поиск компонент сильной связности	285
2.6. Нахождение замкнутой эйлеровой цепи (два стека)	290
2.7. Нахождение замкнутой эйлеровой цепи (Флери)	293
2.8. Построение графа де Брюина	299
2.9. Нахождение гамильтонова цикла	306
2.10. Построение стягивающего дерева	312
2.11. Построение кода Прюфера	317
2.12. Построение дерева по коду Прюфера	318
2.13. Распознавание двудольности графа	330
2.14. Брона — Кербоша	335
2.15. Последовательная раскраска	348
2.16. Построение наибольшего паросочетания в двудольном графе	371
2.17. Прима или ближайшего соседа	374
2.18. Краскала	379
2.19. Модифицированный алгоритм Краскала	380
2.20. Форда — Беллмана	388
2.21. Форда — Беллмана: нахождение дерева кратчайших путей	392
2.22. Дейкстры	397
2.23. Флойда	404
2.24. Алгоритм Ли: распространение волны	412

2.25. Форда — Фалкерсона	419
2.26. Максимальный поток в плоской транспортной сети	422
2.27. Рекурсивный алгоритм «перебор с возвратом»	429
2.28. Метод ветвей и границ	431
2.29. Разбиение на классы эквивалентности	443
2.30. Топологическая сортировка	447
2.31. Алгоритм Уоршелла	455
3.1. Приведение к ДНФ	479
3.2. Первый алгоритм приведения к СДНФ	481
3.3. Второй алгоритм приведения к СДНФ	482
3.4. Первый алгоритм приведения к СКНФ	485
3.5. Второй алгоритм приведения к СКНФ	486
3.6. Метод минимизирующих карт	493
3.7. Метод Куайна — Мак-Класки	498
3.8. Метод резолюций для логики высказываний	506
3.9. Стратегия насыщения уровней	509
3.10. Метод равносильных преобразований для полинома Жегалкина	530
3.11. Метод треугольника Паскаля для полинома Жегалкина	532
3.12. Быстрое вычисление полинома Жегалкина	536
3.13. Приведение к предваренной нормальной форме	578
3.14. Приведение к сколемовской нормальной форме	580
3.15. Инициализация метода резолюций	584
3.16. Алгоритм унификации множества литералов	586
3.17. Метод резолюций для логики предикатов	589
3.18. Удаление непродуктивных нетерминалов	674
3.19. Удаление «неиспользуемых» нетерминалов	675
3.20. Разбор постфиксной записи арифметического выражения	681
3.21. Вычисление значений в постфиксной форме	682
3.22. Перевод инфиксной формы в постфиксную	684
3.23. Нахождение недостижимых состояний	700
3.24. Детерминизация объединением	700
3.25. Детерминизация вытягиванием	702
3.26. Построение минимального автомата	710

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

- Автомат
 - Детерминизация 699
 - Мили
 - — определение 689
 - — примеры 691
 - — способы задания 689
 - Мура
 - — определение 690
 - — примеры 691
 - — способы задания 691
 - Регулярные выражения 715
 - автоматные грамматики 706
 - детеминированный
 - — определение 696
 - детерминизация вытягиванием 702
 - детерминизация объединением 700
 - лемма о накачке 703
 - минимизация 710
 - невозвратное состояние 697
 - правила построения для регулярного выражения 718
 - распознаватель
 - — определение 696
 - — примеры 697
 - редукция 710
 - с эpsilon-переходами 713
- Алгебраическая система 569
- Алгоритм
 - Брона — Кербоса 335
 - Бухбергера 140
 - Дейкстры 397
 - Евклида
 - — анализ 46–51
 - — бинарный 38
 - — для многочленов 116
 - — классический 36
 - — обобщенный 41
 - Задача о неограниченном рюкзаке 203
 - Задача о рюкзаке 0–1 204
 - Зыкова 357
 - Косарайю и Шарира 285
 - Краскала 379
 - — модифицированный 380
 - Куайна — Мак-Класки 498
 - Ли 411
 - Прима 374
 - Уоршелла 455
 - Ферма 23
 - Флери 293

- Флойда 404
- Форда — Беллмана
 - нахождения дерева кратчайших путей 392
- Форда — Беллмана 388
- Хаффмана 171
- Эратосфена 19
- ближайшего соседа 374
- возведения числа в степень 32
- волновой 411
- вычисления квадратного корня 26
- генерации
 - подмножеств 174
 - разбиений числа 201
- деления многочленов 108, 113
- детерминизации вытягиванием 702
- детерминизации объединением 700
- для метода ветвей и границ 431
- жадный 173
- метода резолюций 506, 589
- минимизирующих карт 492
- нахождения гамильтонова цикла 306
- нахождения максимального потока в плоской транспортной сети 422
- нахождения мостов графа 280
- нахождения недостижимых состояний 700
- нахождения эйлеровой цепи 290
- перебора с возвратом 429
- перевода из p -ичной записи в q -ичную в p -ичной арифметике 31
- перевода из p -ичной записи в q -ичную в q -ичной арифметике 32
- перемножения чисел в p -ичной системе счисления 30
- поиска
 - в глубину 274
 - в ширину 276
- поиска компонент связности в графе 279
- поиска компонент сильной связности в графе 285
- полиномиального декодирования 169
- последовательной раскраски 348
- построение кода Прюфера 317
- построения
 - каркаса 312
 - наибольшего паросочетания 371
 - построения графа де Брюина 299
 - построения дерева по коду Прюфера 318
 - построения корневого дерева 314
 - построения минимального автомата 710
 - построения полинома Жегалкина
 - метод неопределенных коэффициентов 531
 - метод равносильных преобразований 530
 - метод треугольника Паскаля 532
- приведения
 - к дизъюнктивной нормальной форме 479
 - к предваренной нормальной форме 578
 - к совершенной

- дизъюнктивной нормальной форме 481, 482
- — к совершенной конъюнктивной нормальной форме 485, 486
- пробных делителей 21
- разбиения на классы эквивалентности 443
- разложения многочлена на свободные от квадратов множители 121
- распознавания двудольности графа 330
- решения
 - — диофантова уравнения 63
 - — линейного сравнения 85
- сдвига числа в p -ичной системе счисления 30
- сколемизации 580
- сложения в p -ичной системе счисления 29
- сложения многочленов 106
- топологической сортировки 447
- умножения многочленов 107
- умножения на цифру в p -ичной системе счисления 30
- шифрования RSA 95
- Алгоритм RSA
 - затемняющий множитель 99, 103
 - ослепление 99
 - схема с нотариусом 99
 - циклическая атака 100
- Алгоритм Маркова 649
 - вычислимая функция 650
 - подстановка 650
 - схема 649
 - формула подстановки 649
- Алфавит 298
- Базис
 - Бандлера — Кохоута 625
 - Вебера 625
 - Гребнера 140
 - Лукашевича — Гилеса 625
 - замкнутого класса 551
 - идеала 131
- Байеса
 - Формула 255
- Биграф 329
- Бинарное отношение 436
 - Парето 446
 - антирефлексивное 438
 - антисимметричное 438
 - асимметричное 438
 - достижимости 443
 - замыкание 452
 - иррефлексивное 438
 - порядка 444
 - — в решетке 514
 - — линейного 445
 - — нестрогого 445
 - — строгого 445
 - — частичного 445
 - предпорядка 444
 - рефлексивное 437
 - симметричное 438
 - согласованное 446
 - толерантности 444
 - транзитивное 439
 - транзитивное замыкание 453
 - эквивалентности 442, 471
- Бинарные диаграммы решений 556
 - упорядоченные 556
- Бинарные отношения
 - нечеткие 619
 - — антирефлексивность 622
 - — антисимметричность 623
 - — асимметричность 623
 - — граф 620

- — котранзитивность 624
- — матрица 620
- — рефлексивность 622
- — свойства 622
- — сильная антирефлексивность 623
- — сильная полнота 623
- — сильная рефлексивность 622
- — симметричность 623
- — слабая антирефлексивность 623
- — слабая полнота 623
- — слабая рефлексивность 622
- — способы задания 619
- — транзитивность 623
- Бином Ньютона 154, 160
- Божественная пропорция 68, 236
- Буква 298
- Булеан 515
- Булева алгебра 515
 - Кантора 516
 - Линденбаума — Тарского 516
 - двойственная 518
- Бэктрекинг (перебор с возвратом) 428
- Вероятность
 - независимые события 252
 - определение 251
 - полная система событий 252
 - условная 253
- Вершина
 - источник 414
 - корень дерева 313
 - лист дерева 313
 - насыщенная в паросочетании 367
 - покрытая в паросочетании 367
 - свободная в паросочетании 367
 - сток 414
- Вхождение переменной
 - свободное 569
 - связанное 569
- Вывод 589
 - резолютивный 504
- Высказывание 463, 566
- Высказывания
 - нечеткие 624
 - — логические операции 625
 - — логические формулы 627
 - — степень равносильности формул 627
- Гипотеза
 - Кармайкла 81
- Грамматика 663
 - автоматная 666
 - алфавит нетерминальных символов 663
 - алфавит терминальных символов 663
 - вывод 664
 - грамматический разбор 669
 - классификация Хомского 665
 - контекстно-зависимая 665
 - контекстно-свободная 666, 667
 - — непродуктивные нетерминалы 673
 - начальный символ 663
 - правила вывода 663
 - регулярная 666
 - свободная 665
 - синтаксический анализатор 670
 - — однозначность ветвления 670
 - — синтаксическая диаграмма 671
 - язык 665
 - язык Дика 666
- Граница 328
 - Хемминга 163
- Грань 341
- Граф

- k -раскрашиваемый 345
- Герца 282
- Дирака 304
- Оре 304
- Петерсена 338, 344
- Ферре 199
- ациклический 310
- бикомпонента 282
- бинарного отношения 440
- вершина 263
 - — валентность 264
 - — висячая 264
 - — звезда 264, 267
 - — изолированная 264
 - — концевая 264
 - — периферийная 268
 - — разделяющая 279
 - — степень 264
 - — точка сочленения 279
 - — центральная 268
 - — шарнир 279
- ветвь 321
- взвешенный 373
- гамильтонов 303
- гомеоморфизм 343
- грань 341
- двудольный 329
- де Брюина 298
- дерево 310
 - — DFS-дерево 312
 - — код Прюфера 317
 - — корневое 313
- диаметр 268
- дополнение 333
- дуга 267
- звезда 329
 - — порядок 329
- изоморфизм 337
- инварианты 337
- история 269
 - — каркас 311
 - — клика 333
 - — максимальная 333
 - — наибольшая 333
 - — кликовое число 333
 - — компоненты реберной двусвязности 279
 - — компоненты связности 278
 - — конденсации 282
 - — коранг 320
 - — кубический 264
 - — лес 310
 - — множество смежности 264
 - — мост 279
 - — мультиграф 263
 - — нагруженный 373
 - — независимое множество 333
 - — максимальное 333
 - — наибольшее 333
 - — неориентированный 263
 - — нуль-граф 264
 - — обратный 333
 - — ориентированный 267
 - — отождествление вершин 357
 - — планарный 339
 - — критерий Вагнера 344
 - — критерий Понтрягина — Куратовского 344
 - — плоский 339
 - — подграф 264
 - — полный 264
 - — полугамильтонов 303
 - — полуэйлеров 289
 - — псевдограф 263, 281
 - — путь 264
 - — вершинно-простой 264
 - — направленный 267
 - — простой 264
 - — реберно-простой 264
 - — эйлеров 289

- радиус 268
- разбиение ребра 343
- раскраска
 - — вершинная 345
 - — реберная 352
- расстояние между вершинами 268
- реберное хроматическое число 352
- реберно отделимый 279
- реберно связный 279
- реберный 353
- ребро 263
 - — ветвь 321
 - — кратное 263
 - — мост 279
 - — обратное 281
 - — отделимости 279
 - — перешеек 279
 - — петля 263
 - — прямое 281
 - — разделяющее 279
 - — хорда 321
- регулярный 264
- связности булевой функции 525
- связный 278
- сильная компонента 282
- сильносвязный 282
- слабосвязный 282
- стягивание вершин 357
- стягивание ребра 344
- транспонированный 283
- турнир 308
 - — 1-парадоксальный 310
 - — транзитивный 309
- узел 267
- укладка
 - — плоская 339
 - — трехмерная 340
- хорда 321
 - — хроматическая редукция 355, 357
 - — хроматический индекс 352
 - — хроматический многочлен 354
 - — формула удаления и стягивания 355
 - — хроматическое число 345
 - — цепь 264
 - — длина 268
 - — цикл 264
 - — главный 321
 - — цикломатическое число 320
 - — четный 329
 - — число независимости 333
 - — число паросочетания 367
 - — Эйлеров 289
 - — эксцентриситет 268
- Группа 71
 - абелева 71
 - классы смежности 218
 - коммутативная 71
 - подгруппа 218
 - подстановок 217
 - симметрий тетраэдра 221
 - система образующих 220
- Делитель
 - наибольший общий 14
 - — линейное представление 39
 - нуля 72
- Деньги электронные 102
- Дерево 310
 - DFS-дерево 312
 - каркас 311
 - код Прюфера 317
 - корень 314
 - корневое 313
 - — алгоритм построения 314
 - — потомок 313
 - — предок 313
 - минимальное остовное 373

- ориентированное 314
- остов 311
- остовое 311
- покрывающее граф 311
- стягивающе 311
- Джон фон Нейман 688
- Диаграмма
 - Ферре 199
 - Хассе 451
 - Юнга 199
- Дивергенция потока 415
- Дизъюнкт 483
 - хорновский 593
- Дизъюнкция 464
 - простая 483
 - элементарная 483
 - — конституента нуля 487
- Динамическое программирование
 - Задача о рюкзаке 202
 - Задача о рюкзаке
 - — Неограниченный рюкзак 203
 - — Рюкзак 0–1 204
- Дискретная случайная величина
 - дисперсия 259
 - закон распределения 258
 - математическое ожидание 259
 - независимость 258
 - определение 258
- Дополнение 515
 - по Сугено 625
 - по Ягеру 625
- Дробь
 - непрерывная 43
 - подходящая 45
 - цепная 43
 - — бесконечная 54
 - — наилучшие приближения 58
 - — применения 60–61
- Задача
 - Кислера 464, 467, 468
 - Монмора 209
 - коммивояжера 432
 - о Ханойских башнях 189
 - о встречах 209
 - о домино 296
 - о заполнении домино 231
 - о кенигсбергских мостах 269, 292
 - о путешественнике 469
 - о разбиении числа 198
 - о размене монет 149
 - о рыцарях и лжецах 478
 - о футбольной команде 152
 - о числе беспорядков 209
 - о числе перестановок 151
 - построения генома 301
 - расстановки ферзей 426
- Закон логики высказываний 471
 - ассоциативности 472
 - двойного отрицания 472
 - де Моргана 472
 - идемпотентности 472
 - исключенного третьего 472
 - коммутативности 472
 - контрапозиции 472
 - противоречия 472
- Замыкание
 - булевых функций 526
 - транзитивное 453, 664
- Запись
 - инфиксная 522
 - польская
 - — обратная 677
 - — прямая 677
 - постфиксная 677
 - префиксная 522, 678
- Золотое сечение 68, 236
- Идеал 131
 - мономиальный 140
- Изоморфизм 337

- булевых алгебр 517
- Импликация 464
 - Бандлера — Кохоута 626
 - Вади 626
 - Геделя 626
 - Лукашевича 626
 - Лукашевича — Гилеса 626
 - Мамдани 626
- Интерпретация формулы
 - логики высказываний 467
 - — выполняющая 476
 - логики предикатов 569
- Иррациональность квадратичная 55
- Квадрат Дюрфи 199
- Квантификация переменной 567
- Класс
 - булевых функций
 - — Поста 548
 - — замкнутый 526
 - — линейных 534
 - — монотонных 544
 - — полный 528
 - — предполный 550
 - — самодвойственных 541
 - — сохраняющих единицу 527
 - — сохраняющих ноль 527
 - вычетов 73, 75, 96
 - грамматик 665, 667
 - частично рекурсивных функций 646
 - эквивалентности 443
- Кограница 328
- Код
 - Грея 188
 - Шеннона — Фано 171
 - беспрефиксный 171
 - линейный 165
 - — циклический 165
 - плотно упакованный 165
 - совершенный 165
 - с проверкой на четность 170
- Кодовое расстояние
 - Хемминга 162
 - минимальное 165
- Кольцо 71
 - ассоциативное 72
 - булево 518
 - вычетов 75
 - коммутативное 72
 - многочленов 108
 - с единицей 72
- Конечный автомат
 - Детерминизация 699
 - Мили
 - — определение 689
 - — примеры 691
 - — способы задания 689
 - Мура
 - — определение 690
 - — примеры 691
 - — способы задания 691
 - Регулярные выражения 715
 - автоматные грамматики 706
 - детеминированный
 - — определение 696
 - с эпсилон-переходами 713
- Константа
 - предметная 568
- Контакт 488
- Контактная схема 488
 - замкнутая 488
 - разомкнутая 488
- Континуанта 46
- Конъюнкт 479
- Конъюнкция 464
 - простая 479
 - элементарная 479
 - — конституента единицы 483
- Коцикл 324

- главный 325
- Коэффициент
 - биномиальный 83, 160
- Коэффициенты Безу 40
- Кратное наименьшее общее 14
- Критерий
 - Поста 548
 - Эйзенштейна 124
- Лемма
 - Огдена 703
 - Эйлера 47
 - о моделировании 528
 - о накачке 703
 - о нелинейной функции 535
 - о немонотонной функции 546
 - о несамодвойственной функции 542
 - о переименовании связанных переменных 575
 - о подстановке 473
 - о рукопожатиях 264
- Лес 310
- Литерал 479, 579
 - контрарный 503
- Матрица
 - весов 373
 - инцидентий 270
 - отношения 440
 - смежности 271
- Машина Тьюринга 632
 - активная зона 634
 - геделева нумерация 644
 - команда 632
 - композиция 641
 - конфигурация 633
 - правильно вычисляющая функцию 635
 - проблема останова 643
 - программа 633
 - разветвление 642
 - реализация цикла 642
 - функциональная схема 633
- Мера
 - Егера 611
 - Коско 611
- Метод
 - гарвадский 491
 - минимизирующих карт 491
 - неопределенных коэффициентов 491
 - разбиения 710
 - раскраски 443
- Мили
 - автомат
 - — определение 689
 - — примеры 691
 - — способы задания 689
- Минимизация
 - автоматов-распознавателей 710
- Многочлен
 - Лорана 131
 - Чебышева 239
 - зацепление 139
 - многих переменных 130
 - — линейный 132
 - мультииндекс 130
 - над полем 106
 - неприводимый 120
 - одночлен 130
 - ошибок 167
 - порождающий 166
 - примитивный 120
 - разложение на свободные от квадратов множители 120
 - редукция 136
 - свободный от квадратов 120
 - синдромный 167
 - степень 106
 - факториальный 192
 - характеристический 237

- Множества
 — логические операции
 — — Дизъюнктивная сумма 609
 — — контрастная интенсификация 607
 — — концентрирование 607
 — — размывание 607
 — функция принадлежности
 — — S -образная функция 616
 — — Метод прямого оценивания 617
 — — Методы построения 617
 — — гауссова 614
 — — колоколообразная 614
 — — трапецидальная 613
 Множество
 — линейно упорядоченное 450
 — предметное 566
 — унифицируемое 585
 — частично упорядоченное 450
 Модель 570
 Модулярная арифметика 93
 Моном 138
 Мура
 — автомат
 — — определение 690
 — — примеры 691
 — — способы задания 691
 Нечеткие
 — бинарные отношения 619
 — — антирефлексивность 622
 — — антисимметричность 623
 — — асимметричность 623
 — — граф 620
 — — котранзитивность 624
 — — матрица 620
 — — рефлексивность 622
 — — свойства 622
 — — сильная антирефлексивность 623
 — — сильная полнота 623
 — — сильная рефлексивность 622
 — — симметричность 623
 — — слабая антирефлексивность 623
 — — слабая полнота 623
 — — слабая рефлексивность 622
 — — способы задания 619
 — — транзитивность 623
 — высказывания 624
 — — логические операции 625
 — — логические формулы 627
 — — степень равносильности формул 627
 Нечеткие множества 602
 — α -уровня 603
 — Ближайшее четкое 603
 — Выпуклая комбинация 610
 — Кардинальное число 603
 — Мощность 603
 — Ядро 603
 — высота 603
 — декомпозиция 605
 — логические операции 606
 — — Дизъюнктивная сумма 609
 — — контрастная интенсификация 607
 — — концентрирование 607
 — — размывание 607
 — мера Егера 611
 — мера Коско 611
 — мера нечеткости 611
 — нормальное множество 603
 — носитель 603
 — оператор увеличения нечеткости 611
 — операции 606
 — отношения включения 606
 — субнормальное множество 603
 — точка перехода 603

- умножение на число 610
- функция принадлежности 603, 613
- — S -образная функция 616
- — Метод попарных сравнений 617
- — Метод прямого оценивания 617
- — Методы построения 617
- — Частотный метод 617
- — Шкала Саати 618
- — гауссова 614
- — колоколообразная 614
- — сплайн-функция 615
- — трапецидальная 613
- Область
- декодирования 163
- Область действия квантора 568
- Оператор
- минимизации 647
- примитивной рекурсии 647
- суперпозиции 647
- Опровержение 504
- Остов 311
- Отрицание 464
- Пакет символьных вычислений
- Maple 144–145, 250
- Паросочетание 367
- максимальное по включению 367
- максимальное по размеру 367
- наибольшее 367
- совершенное 367
- Переменная
- предметная 566
- существенная 523
- фиктивная 523
- Перестановка 157
- Перечисление
- декартова произведения 180, 182
- перестановок 182–187
- подмножеств 175–180
- Подпись электронная 101
- Подформула 467
- Позиционные системы счисления
- алгоритмы арифметических действий 29–31
- алгоритмы перевода 31–32
- факториальная запись 29
- Поле 72
- Полная система вычетов 73
- Пометка вершины 385, 388, 389, 393
- инициализация 404
- модификация 386, 397, 403
- Порядок
- антилексикографический 173
- лексикографический 173, 524
- обратный лексикографическому 174
- Поток в сети 414
- Правило резолюций
- в логике высказываний 503
- в логике предикатов 588
- Предикат 566
- Принцип
- Дирихле 155
- включения-исключения 206
- двойственности 475, 484, 486, 539
- нормализации 651
- Производящая функция 155, 233
- для чисел Каталана 245
- для чисел Фибоначчи 235
- числа разбиений 234
- Пролог
- база знаний 593
- запрос 593

- правило 593
- факт 593
- Путь
 - гамильтонов 400
 - кратчайший 385–386
 - эйлеров 292
- Разложение Шеннона 555
- Размещение 157
 - без повторений 157
 - с повторениями 158
- Разрез 324
 - $s - t$ -разрез 416
 - величина потока через разрез 416
 - минимальный 324
 - пропускная способность 416
 - простой 324
- Регулярные выражения 715
 - замыкание Клини 716
 - итерация языков 716
 - конкатенация языков 716
 - объединение языков 716
 - свойства 717
 - эквивалентность 717
- Редукция конечного автомата 710
- Резольвента 503
 - бинарная 588
- Решетка 513
 - дистрибутивная 514
 - ограниченная 515
 - с дополнением 515
- Связки логические 464
- Семантические сети 459
 - неоднородные 459
 - однородные 459
- Сеть 414
 - транспортная 414
- Сигнатура 568
- Символ функциональный 568
- Система остаточных классов 93
- Склейка дизъюнкта 588
- Сколемизация 579
- Слово 298
 - длина 298
 - префикс 298
 - суффикс 298
- Соединение контактов
 - параллельное 488
 - последовательное 488
- Соотношение Безу 40
- Сочетание 157
 - без повторений 157
 - с повторениями 158
- Список инцидентности 272
- Сравнение по модулю 73
 - линейное 85
- Стиль календарный 60
- Стратегия метода резолюций
 - вычеркивания 510
 - насыщения уровней 509
- Субфакториал 209
- Схема Горнера 114
- Таблица
 - Поста 551
 - истинности 464, 468
- Тезис
 - Тьюринга 643
 - Черча 648
- Теорема
 - Безу 114
 - Бернсайда 224
 - Вандермонда 193
 - Визинга
 - — о вершинной раскраске 346
 - — о реберной раскраске 352
 - Вильсона 88
 - Дирака 304
 - Евклида 20
 - Зыкова 355
 - Кенига

- — критерий двудольности графа 330
- — о реберной раскраске двудольного графа 353
- — о сумме степеней двудольного графа 332
- Лагранжа 55
- Ламе 49
- Лежандра 207
- Оре 303
- Понтрягина — Куратовского 344
- Поста 548
- Редери — Камиона 309
- Ферма (малая) 83
- Форда — Фалкерсона 418
- Хивуда 346
- Цекендорфа 68
- Чебышева 20
- Эйлера 264, 342
- Эйлера — Ферма 83
- китайская об остатках
 - — для многочленов 118
 - — для чисел 89
- об эквивалентности автоматов Мили и Мура 692
- о декомпозиции 605
- о детерминизации 699
- о полноте метода резолюций 505
- о разрастании для автоматных языков 703
- Терм 568
- Тожество Коши 161
- Треугольник Паскаля 148, 161
- Турнир 308
 - 1-парадоксальный 310
 - транзитивный 309
- Унификатор 585
 - наиболее общий 585
- Уравнение
 - диофантово 61
 - рекуррентное 233
 - — неоднородное 243
 - — однородное 236
 - — порядок 233
- Форма Бэкуса — Наура 667
- Формула
 - Байеса 255
 - Бине 236, 239
 - Гаусса 80
 - Лагранжа 119
 - логики высказываний 466
 - — выполнимость 476
 - — дизъюнктивная нормальная форма 479, 489
 - — конъюнктивная нормальная форма 483, 489
 - — логическое следствие 475
 - — минимальная дизъюнктивная нормальная форма 491
 - — противоречие 471
 - — равносильность 470
 - — совершенная дизъюнктивная нормальная форма 480
 - — совершенная конъюнктивная нормальная форма 484
 - — с тесными отрицаниями 479
 - — тавтология 471
 - логики предикатов 568
 - — замкнутая 569
 - — логическое следствие 576
 - — общезначимая 573
 - — предваренная нормальная форма 578
 - — предложение 569
 - — пренексная нормальная форма 578
 - — равносильность 573

— — сколемовская нормальная форма 579
— обращения Мебиуса 212, 248
Функция
— Мебиуса 82, 210
— Эйлера 78
— булева 521
— — Вебба 522
— — Шеффера обобщенная 551
— — вес 524
— — геометрическое представление 525
— — голосования 524
— — двойственная 539
— — замыкание 526
— — книжал Куайна 522
— — линейная 534
— — мажоритарная 524
— — монотонная 544
— — неравнозначности 522
— — область истинности 524
— — полином Жегалкина 529
— — полная 551
— — самодвойственная 540
— — сохраняющая единицу 527
— — сохраняющая ноль 527
— — стрелка Лукасевича 522
— — стрелка Пирса 522
— — суперпозиция 526
— — четности 543
— — штрих Шеффера 522
— мультипликативная 81
— односторонняя 94
— оценочная для целевой функции 430
— расстояния 162
— рекурсивная
— — общерекурсивная 648
— — примитивно-рекурсивная 648
— — частично рекурсивная 648

— с секретом 95
Хемминга
— вес кодового слова 165
— граница 163
— кодовое расстояние 162
Цепочка 298
Цикломатическое число 320
Числа
— Белла 191
— Каталана 198
— Стирлинга
— — второго рода 190
— — первого рода 192
— Фибоначчи 66, 239
— — соотношение Кассини 67
— — система счисления 68
— бесквадратные 19
— взаимно простые 15
— каноническое представление 17
— простые 17
— свободные от квадратов 19
Число
— Фидия 68, 236
Эквиваленция 464

Сергей Витальевич РЫБИН
**ДИСКРЕТНАЯ МАТЕМАТИКА
И ИНФОРМАТИКА**
Учебник

Зав. редакцией
литературы по информационным технологиям
и системам связи *О. Е. Гайнутдинова*
Ответственный редактор *В. В. Яески*
Корректор *Т. А. Кошелева*
Выпускающий *Т. А. Быченкова*

ЛР № 065466 от 21.10.97
Гигиенический сертификат 78.01.10.953.П.1028
от 14.04.2016 г., выдан ЦГСЭН в СПб

Издательство «ЛАНЬ»
lan@lanbook.ru; www.lanbook.com
196105, Санкт-Петербург, пр. Юрия Гагарина, д. 1, лит. А
Тел./факс: (812) 336-25-09, 412-92-72
Бесплатный звонок по России: 8-800-700-40-71

ГДЕ КУПИТЬ

ДЛЯ ОРГАНИЗАЦИЙ:

*Для того, чтобы заказать необходимые Вам книги, достаточно обратиться
в любую из торговых компаний Издательского Дома «ЛАНЬ»:*

по России и зарубежью
«ЛАНЬ-ТРЕЙД». 196105, Санкт-Петербург, пр. Юрия Гагарина, д. 1, лит. А
тел.: (812) 412-85-78, 412-14-45, 412-85-82; тел./факс: (812) 412-54-93
e-mail: trade@lanbook.ru; ICQ: 446-869-967

www.lanbook.com
пункт меню «Где купить»
раздел «Прайс-листы, каталоги»

в Москве и в Московской области
«ЛАНЬ-ПРЕСС». 109387, Москва, ул. Летняя, д. 6
тел.: (499) 722-72-30, (495) 647-40-77; e-mail: lanpress@lanbook.ru

в Краснодаре и в Краснодарском крае
«ЛАНЬ-ЮГ». 350901, Краснодар, ул. Жлобы, д. 1/1
тел.: (861) 274-10-35; e-mail: lankrd98@mail.ru

ДЛЯ РОЗНИЧНЫХ ПОКУПАТЕЛЕЙ:

интернет-магазин
Издательство «Лань»: <http://www.lanbook.com>
магазин электронных книг
Global F5: <http://globalf5.com/>

Подписано в печать 20.10.21.
Бумага офсетная. Гарнитура Школьная. Формат 70×100^{1/16}.
Печать офсетная/цифровая. Усл. п. л. 60,78. Тираж 30 экз.

Заказ № 1260-21.

Отпечатано в полном соответствии с качеством
предоставленного оригинал-макета в АО «Т8 Издательские Технологии».
109316, г. Москва, Волгоградский пр., д. 42, к. 5.