

Министерство науки и высшего образования Российской Федерации
Ярославский государственный университет им. П. Г. Демидова
Кафедра алгебры и математической логики

Введение в теорию кодирования, сжатия и восстановления информации

Учебно-методическое пособие

Ярославль
ЯрГУ
2020

УДК 004.056.5(075.8)
ББК В182_я73
К14

Рекомендовано
Редакционно-издательским советом университета
в качестве учебного издания. План 2020 года

Рецензент
кафедра алгебры и математической логики
Ярославского государственного университета им. П. Г. Демидова

Казарин, Лев Сергеевич.

К14 Введение в теорию кодирования, сжатия и восстановления информации : учебно-методическое пособие / Л. С. Казарин, М. А. Заводчиков ; Яросл. гос. ун-т им. П. Г. Демидова. – Ярославль : ЯрГУ, 2020. – 112 с.

Настоящее пособие, являющееся основой курса для студентов - математиков, специализирующихся в области защиты информации, предназначено для их ознакомления с основными идеями теории кодирования, хранения и сжатия информации. В наше время это хорошо развитое направление, являющееся источником многочисленных приложений и новых идей в математике, органически связанное со всеми другими курсами этого направления.

УДК 004.056.5(075.8)
ББК В182_я73

© ЯрГУ, 2020

Содержание

I. Элементы теории кодирования	8
1. Основная проблема кодирования	8
1.1. Основные понятия теории кодов	8
1.2. Примеры	10
1.3. Линейные (матричные) коды	11
1.5. Недвоичные коды	14
1.6. Упражнения	16
2. Некоторые сведения из теории групп	17
2.1. Определения	17
2.2. Примеры	18
2.3. Теоремы	20
2.4. Упражнения	21
3. Оценки качества кода.	
Подход к решению задачи	23
3.1. Геометрическая интерпретация	23
3.2. Групповые коды	27
4. Вероятность ошибки декодирования	29
4.1. Понятие о теореме Шеннона	31
4.2. Упражнения	32
5. Снова о линейных кодах	34
5.1. Предварительные замечания	34
5.2. Корректирующая способность линейного кода	35
5.3. Коды Хэмминга	36
5.4. Техника матричного кодирования и декодирования	41
5.5. Дуальный код	42
5.6. Упражнения	43
6. Конечные поля	46
6.1. Аддитивная структура конечных полей	46
6.2. Мультипликативная конечных полей	47

6.3.	Конструирование конечных полей	48
6.4.	Упражнения	50
7.	Полиномиальные и циклические коды	51
7.1.	Полиномиальные коды	51
7.2.	Двоичные циклические коды	54
7.3.	Техническая реализация циклических и полиномиальных кодов	56
7.4.	Кодер для циклического кода	58
7.5.	Декодер для кода Хэмминга	59
7.6.	Упражнения	61
8.	Двоичные БЧХ-коды	61
8.1.	БЧХ-коды, исправляющие 2 ошибки	61
8.2.	Двоичные БЧХ-коды, исправляющие t ошибок	63
8.3.	Общая схема декодера для БЧХ-кода	66
8.4.	Упражнения	67
9.	Заключительные замечания	67
9.1.	БЧХ-коды над полями нечетной характеристики и коды Рида-Соломона	67
9.2.	Латинские квадраты и коды	69
9.3.	Коды Рида – Маллера	69
9.4.	Матрицы Адамара	70
9.5.	Метрика Хэмминга и метрика Ли	71
9.6.	Границы возможного и невозможного в кодировании	72
9.7.	Упражнения	73
II.	Элементы теории сжатия и восстановления информации	74
10.	Методы сжатия информации	74
11.	Энтропия и информация	78
11.1.	Энтропия	78
11.2.	Энтропия двумерной случайной величины	80
11.3.	Условная энтропия	80

11.4. Энтропия и сжатие информации	81
11.4.1. Информация	81
11.4.2. Сжатие данных	82
11.5. Упражнения	83
12.Алгоритм Шеннона–Фэно	84
12.2. Упражнения	87
13.Алгоритм Хаффмана	88
13.2. Упражнения	92
14.Арифметическое кодирование	93
14.2. Адаптивное арифметическое кодирование	95
14.3. Упражнения	96
15.Словарные алгоритмы сжатия информации	96
15.1. LZ77	97
15.2. LZ78	99
15.3. Упражнения	102
16.Преобразование	Барроуза–Уиллера
и RLE	103
16.1. Упражнения	106
17.Приложения	107
17.1. Лабораторные работы	107
Литература	108

Введение

Настоящее пособие предназначено для ознакомления студентов - математиков с основными идеями теории кодирования, проделавшей стремительный путь в своем развитии с начала 50-х годов прошлого столетия. В настоящий момент это хорошо развитое направление, являющееся источником многочисленных приложений и новых идей в математике. Можно указать несколько прекрасных книг, посвященных различным аспектам этой теории, например [7], [11], [4], [3], [10].

Более 30 лет тому назад один из авторов данного пособия опубликовал методическую работу с названием «Теория кодирования», которая оказала определенное влияние на подготовку математиков в ЯрГУ, будучи одним из немногих курсов, имеющих связи как с абстрактными моделями алгебры, так и с современными техническими устройствами, предназначенными для защиты информации от природных помех (то, что ныне фигурирует в массмедиа как «цифровые технологии»). Цель, разумеется, была сделать очень краткий и доступный для недавних школьников курс, содержащий основные понятия, терминологию и подходы к решению задач защиты информации от природных помех. Опыт показал, что пособие усваивается успешно большинством студентов. Однако небольшое количество экземпляров и их полиграфическое качество заставило переписать текст заново, добавив некоторые новые разделы и исправив опечатки, имевшиеся в первом издании. В пособие добавлен важный раздел, посвященный сжатию информации.

В теории передачи информации одной из наиболее важных является следующая проблема. По некоторому каналу связи с шумом передается сообщение. Требуется обеспечить его правильную передачу. Сообщение может быть определено как последовательность знаков некоторого алфавита. При этом шум в канале может искажать передаваемые символы. Ситуация несколько напоминает известную детскую игру «испорченный телефон». Только задача теории кодирования противоположная – не дать шуму исказить сообщение. Если последнее записано очень экономно, то оно менее помехоустойчиво (заметим, что одна из трудностей, стоящих перед исследователями некоторых древних языков – отсутствие в их письменности гласных). Реально все человеческие языки обладают огромной избыточностью, которая позволяет им доносить до нас информацию через большие временные промежутки. Вспомним «Слово о полку

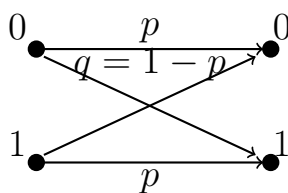


Рис. 1: Двоичный симметричный канал с вероятностью ошибки q

Игорева», пролежавшее в неизвестности более 600 лет и ставшее нам сразу понятным. Имеются, правда, некоторые трудности в идентификации отдельных слов, но общий смысл поэмы понятен.

Итак, наиболее действенный способ борьбы с шумом – введение избыточности. Известная поговорка гласит: «Повторение – мать учения». Здесь в неявном виде заложены как принцип избыточности, так и один из наиболее популярных кодов – код с повторами.

Перейдем к формальной стороне вопроса.

Сообщение – последовательность символов («букв») некоторого конечного алфавита. Чаще всего нам придется в этом пособии иметь дело с алфавитом, состоящим из двух символов – 0, 1.

Передача сообщений ведется по некоторому каналу с шумом. Для алфавита 0, 1 наиболее распространенной моделью является *двоичный симметричный канал*. Пусть в среднем на каждые N переданных символов алфавита приходится M искаженных символов. Тогда отношение $q = \frac{M}{N}$ можно назвать вероятностью ошибки. В этом случае $p = 1 - q$ – вероятность правильной передачи символа. Сверх того, в рассматриваемой модели предполагается, что ошибки в передаче символов появляются независимо одна от другой. Описанная (сильно идеализированная) модель называется двоичным симметрическим каналом. Схематически эту модель принято изображать в следующем виде (рис 1). Реально эта модель не может существовать. Например, чаще всего нарушается предположение о независимости появления ошибок ("Беда приходит не одна"). Кроме того, идеализацией является и допущение о равноправности переходов $0 \rightarrow 0$ и $1 \rightarrow 1$. Наконец, факт существования вероятности ошибки также ничем не обоснован. Тем не менее модель двоичного симметричного канала является очень полезной и позволяет построить содержательную теорию, имеющую важные практические применения.

Часть I.

Элементы теории кодирования

1. Основная проблема кодирования

Это был как раз такой день,
когда надо было, скажем,
написать письмо (подпись —
Кролик), день, когда следовало
все проверить, все выяснить,
все разъяснить и, наконец,
самое главное — что-то
организовать.

А. А. Милн. Винни-Пух
и все-все-все.

1.1. Основные понятия теории кодов

Пусть имеется сообщение $U = \{u_1, u_2, \dots, u_k\}$ ($u_i \in \{0, 1\}$, $1 \leq i \leq k$), которое должно быть передано по двоичному симметричному каналу. Из-за существования помех передача сообщения в чистом виде исключена. Например, если канал связи физически реализуется электромагнитным полем между Землей и спутником, то, накладываясь на внешнее поле, сообщение может измениться до неузнаваемости. Этот канал связи также весьма чувствителен к солнечным вспышкам и атмосферным явлениям. Поэтому сообщение U *кодируется* (вообще говоря, другой) последовательностью $C = \{x_1, x_2, \dots, x_n\}$, по которой можно восстановить U . Последовательность C называется в этом случае *кодовым словом*,

а U — *информационным словом*. При этом число k символов в слове U называется *числом информационных символов*, а число n символов в слове C — *блоковой длиной* кодового слова C . Устройство, осуществляющее нахождение кодового слова по информационному, называется *кодером*.

После того как кодовое слово C передано и принято приемным устройством, оказывается необходимым устранить вкравшиеся ошибки и *декодировать* принятое слово C' , поставив ему в соответствие информационное слово. Техническое устройство, осуществляющее эту операцию, называется *декодером*. Таким образом, общую схему математической модели связи можно представить в следующем виде (рис. 2).

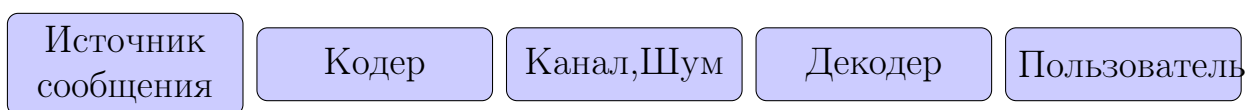


Рис. 2: Схема кодирования

Рис. 2 часто называют гербом теории кодирования. Вообще говоря, декодер не знает ничего о характере шума. Поэтому стратегия декодера будет состоять в том, чтобы выделить наиболее вероятный вид шума и декодировать принятое слово наиболее правдоподобным образом. В том случае, когда принятое слово непременно декодируется каким-либо способом (возможно, неправильно), говорят, что имеется система с *полным декодированием*. Легко представить себе ситуацию, когда принятие даже неправильного решения предпочтительнее отсутствия какого-либо решения. Например, если межпланетная станция передает изображение какого-либо объекта, то у приемника зачастую просто нет возможности уточнить сообщение. Такие ошибки называются *ошибками декодирования*.

Альтернативный метод — те сообщения, в правильном приеме которых нет уверенности — можно либо игнорировать, либо попросить повторить.

Схемы, в которых предусмотрено декодирование не всех сообщений, называются схемами с *неполным декодированием*. Можно представить себе систему связи, в которой неправильное сообщение может иметь катастрофические последствия. Например, получение неправильной команды на космическом корабле. Обычно легко бывает превратить систему с полным декодированием в систему с неполным декодированием, но не наоборот.

При построении кодирующих и декодирующих устройств приходится решать самые разнообразные математические задачи. В частности, необходимо строить коды, обладающие большой защищенностью от шума, достаточно быстрые, т. е. обладающие небольшой избыточностью, и простые в технической реализации (достаточно дешевые). Излишне говорить, что в определенной степени эти требования противоречивы. В теории кодирования строятся алгоритмы и схемы кодирования и декодирования, являющиеся разумными компромиссами для предъявляемых требований.

Ограничимся изучением важного вопроса класса кодов — кодов, имеющих фиксированную блоковую длину. Пусть k — число информационных символов в любом сообщении и пусть любое кодовое слово, которое ставится в соответствие этому сообщению, имеет длину, равную n . Тогда говорят, что имеется код с *блоковой длиной* n и k информационными символами. Число $r = n - k$ характеризует *избыточность кода*, а отношение $R = \frac{k}{n}$ называется *скоростью* кода. Множество всех кодовых слов называется *кодом*. Код с блоковой длиной n и k информационными символами называется (n, k) — *кодом*.

Отметим, что если алфавит кода состоит только из двух символов $\{0, 1\}$, то общее число двоичных слов равно 2^n , в то время как кодовыми из них являются только 2^k слов.

1.2. Примеры

1.2.1.

Одним из простейших примеров двоичных кодов является *код с повторением* с $k = 1$ и произвольным $r = n - 1$. Пусть $n = 3$. Тогда кодовые слова состоят из двух последовательностей: $C_1 = (0, 0, 0)$ и $C_2 = (1, 1, 1)$. Скорость данного кода $R = \frac{1}{3}$. В общем случае $R = \frac{1}{n}$. Коды с повторением обладают исключительно хорошими корректирующими способностями и очень медлительны. Алгоритм кодирования предельно прост. Информационному (первому) символу приписывается r проверочных символов, совпадающих с информационным.

Алгоритм наиболее правдоподобного декодирования также очень прост. Если вероятность искажения символа $q < \frac{1}{2}$, то принятое слово C' декодируется как $(0, 0, \dots, 0)$, если число нулей в нем больше, чем единиц, и декодируется, как $(1, 1, \dots, 1)$ в противном случае. Если выбрать блоковую длину n достаточно большой, то можно обезопасить себя от ошибки

с очень большой степенью уверенности. Плата за надежность – очень низкая скорость. Если $n = 4$, то в случае приема слова $C' = (0, 1, 0, 1)$ декодирование при помощи «большинства голосов» невозможно. Данная схема оказывается с неполным декодированием.

1.2.2.

Следующий код также весьма популярен и часто используется в приложениях. Он называется *кодом с проверкой на четность*. Блочная длина n произвольна, число информационных символов $k = n - 1$. Первые k символов кодового слова представляют собой информационное слово, а последний, n -й символ, – проверочный. Пусть кодовое слово $C = (\alpha_1, \alpha_2, \dots, \alpha_{n-1}, \alpha_n)$. Тогда символ α_n равен нулю, если число единиц в информационном слове $U = (\alpha_1, \alpha_2, \dots, \alpha_{n-1})$ четно и $\alpha_n = 1$ в противном случае. Например, если $n = 4$, то $k = 3$. Все информационные слова и им соответствующие кодовые легко перечислить:

$$\begin{aligned} U_1 = (0, 0, 0) &\leftrightarrow (0, 0, 0, 0) & U_5 = (1, 0, 1) &\leftrightarrow (1, 0, 1, 0) \\ U_2 = (0, 0, 1) &\leftrightarrow (0, 0, 1, 1) & U_6 = (1, 0, 0) &\leftrightarrow (1, 0, 0, 1) \\ U_3 = (0, 1, 0) &\leftrightarrow (0, 1, 0, 1) & U_7 = (1, 1, 0) &\leftrightarrow (1, 1, 0, 0) \\ U_4 = (0, 1, 1) &\leftrightarrow (0, 1, 1, 0) & U_8 = (1, 1, 1) &\leftrightarrow (1, 1, 1, 1) \end{aligned}$$

Нетрудно проверить, что все кодовые слова содержат четное число единиц. Если в процессе передачи в кодовое слово C вкралось нечетное число ошибок, то принятое слово C' будет содержать нечетное число единиц. Поэтому любое нечетное число ошибок можно обнаружить (хотя может и остаться неясным, в каких позициях эти ошибки были сделаны, а также их точное число). К примеру, если принято слово $C' = (1, 0, 0, 0)$, то переданными могли быть слова C_1, C_5, C_6, C_7 (в случае ровно одной ошибки).

Таким образом, данный код может лишь обнаруживать, но не исправлять ошибки. Преимущество этого кода в его скорости: $R = \frac{n-1}{n} \rightarrow 1$ при $n \rightarrow 1$. В реальных системах кодирования используются идеи, заложенные в обоих примерах.

1.3. Линейные (матричные) коды

Приведенные в п. 1.2 примеры нетрудно обобщить, если ввести в рассмотрение следующие операции над символами 0 и 1:

Таблица сложения Таблица умножения

+	0	1
0	0	1
1	1	0

×	0	1
0	0	0
1	0	1

Операции $+$, \times , заданные этими таблицами, называются двоичными суммой и произведением. В этом случае множество $\mathbb{Z}_2 = \{0, 1\}$ оказывается полем $GF(2)$ ([2], [3]), а кодовые и информационные слова — элементами линейных векторных пространств размерностей n и k соответственно.

Пусть $\mathcal{K}(n, 1)$ — код из примера 1.2.1 $C = (\alpha_1, \alpha_2, \dots, \alpha_n) \in \mathcal{K}$. Тогда $\alpha_2 = \alpha_1$, $\alpha_3 = \alpha_1$, ..., $\alpha_n = \alpha_1$, ($\alpha_1 \in \{0, 1\}$).

Пусть $\mathcal{K}(n, n-1)$ — код из примера 1.2.2 $C = (\alpha_1, \alpha_2, \dots, \alpha_n) \in \mathcal{K}$. Тогда $\alpha_n = \alpha_1 + \alpha_2 + \dots + \alpha_{n-1}$, где $\alpha_1, \alpha_2, \dots, \alpha_{n-1} \in \{0, 1\}$.

Обобщая эти примеры, можно выделить класс (n, k) — кодов с блоковой длиной n и k информационными символами $\alpha_1, \alpha_2, \dots, \alpha_k$, для которых оставшиеся (проверочные) символы являются их линейными комбинациями:

$$\begin{cases} \alpha_{k+1} = a_{11}\alpha_1 + a_{12}\alpha_2 + \dots + a_{1k}\alpha_k \\ \alpha_{k+2} = a_{21}\alpha_1 + a_{22}\alpha_2 + \dots + a_{2k}\alpha_k \\ \vdots \\ \alpha_n = a_{n-k+1}\alpha_1 + a_{n-k+2}\alpha_2 + \dots + a_{n-k+k}\alpha_k, \end{cases}$$

где $A = (a_{ij})$ — матрица размера $(n-k) \times k$ с коэффициентами $a_{ij} \in \mathbb{Z}_2$.

В более симметричном виде эти соотношения можно переписать так:

$$\begin{cases} 0 = a_{11}\alpha_1 + a_{12}\alpha_2 + \dots + a_{1k}\alpha_k - \alpha_{k+1} \\ 0 = a_{21}\alpha_1 + a_{22}\alpha_2 + \dots + a_{2k}\alpha_k - \alpha_{k+2} \\ \vdots \\ 0 = a_{n-k+1}\alpha_1 + a_{n-k+2}\alpha_2 + \dots + a_{n-k+k}\alpha_k - \alpha_n \end{cases}$$

или в матричном виде $HC^t = 0$, где $C = (\alpha_1, \alpha_2, \dots, \alpha_n)$, $H = (A | -I)$, где I — единичная $(n-k) \times (n-k)$ — матрица. Матрица H называется *проверочной матрицей* соответствующего (матричного) линейного кода \mathcal{K} . В частности, оба примера из п. 1.2 являются примерами матричных кодов.

Итак, можно ввести следующее

Определение 1. Код \mathcal{K} называется линейным кодом (с блоковой длиной n и k информационными символами) с проверочной матрицей H размера $(n - k) \times n$, если он состоит из всех тех векторов C пространства \mathbb{Z}_2^n , для которых $HC^t = 0$ (t — операция транспонирования).

Число $n - k$ — число проверок кода \mathcal{K} . Нетрудно показать, что имеет место следующая

Теорема 1. Пусть \mathcal{K} — линейный код с блоковой длиной n . Тогда \mathcal{K} является подпространством линейного векторного пространства \mathbb{Z}_2^n .

1.4.

Пример. Проверочная матрица кода \mathcal{K} имеет вид

$$H = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Как видно из **определения**, число проверок равно 3, $k = 2$. $A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$.

Если $C = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$ — кодовое слово, то $\alpha_3 = \alpha_2, \alpha_4 = \alpha_1 + \alpha_2, \alpha_5 = \alpha_1$. Поэтому **кодowymi** словами будут следующие:

$$\begin{aligned} C_1 &= (0, 0, 0, 0, 0), & C_3 &= (1, 0, 0, 1, 1), \\ C_2 &= (0, 1, 1, 1, 0), & C_4 &= (1, 1, 1, 0, 1). \end{aligned}$$

Удобно, но не очень существенно то, что проверочная матрица H имеет вид $(A | -I)$. В этом случае первые k символов — информационные, а остальные — проверочные и говорят, что A записана в *каноническом виде*.

Если известно сообщение $U = (\alpha_1, \alpha_2, \dots, \alpha_k)$, то для того, чтобы найти кодовое слово $C = (\alpha_1, \alpha_2, \dots, \alpha_k, \alpha_{k+1}, \dots, \alpha_n)$, можно воспользоваться следующим приемом (предполагается, что проверочная матрица H имеет канонический вид). Учитывая, что
$$\begin{pmatrix} \alpha_{k+1} \\ \dots \\ \alpha_n \end{pmatrix} = A \begin{pmatrix} \alpha_1 \\ \dots \\ \alpha_k \end{pmatrix},$$
 получим

$$C^t = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_k \\ \dots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} I \\ -A \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_k \end{pmatrix} = \begin{pmatrix} I \\ -A \end{pmatrix} U^t.$$

Транспонируя это равенство, имеем $C = UG$, где $G = (I | -A^t)$. Здесь I — единичная $k \times k$ -матрица.

Матрица G называется *порождающей* или *кодирующей* матрицей линейного кода. Равенство $C = UG$ означает, что кодовыми словами являются всевозможные линейные комбинации строк матрицы G . Это свойство можно было бы взять в качестве определения линейного кода.

Из определения линейного кода и матриц H и G следует

Теорема 2. Матрицы G и H связаны соотношениями

$$HG^t = 0; \quad GH^t = 0.$$

Отметим, что нулевые матрицы, фигурирующие в теореме 3, не обязаны быть одинаковыми. Матрица HG^t имеет размер $(n-k) \times k$, тогда как GH^t — размера $k \times (n-k)$. Вообще говоря, не требуется, чтобы порождающая матрица G кода \mathcal{K} имела вид $(I | -A^t) = G$. Однако ранг матрицы G обязан равняться k . Если эта ситуация имеет место, то говорят, что \mathcal{K} — систематический линейный код. Легко видеть, что выбор матрицы для систематического линейного кода как раз обеспечивает равенство k для ранга матрицы G .

1.5. Недвоичные коды

Теория кодов, исправляющих ошибки, достаточно далеко продвинута для двоичных кодов и в меньшей степени для кодов с недвоичным алфавитом. Между тем использование более богатого алфавита технически возможно и в большинстве случаев более удобно с практической точки зрения (этим обеспечивается большая скорость передачи). Чаще всего в качестве символов алфавита берутся элементы некоторого конечного поля. Последнее требование возникает в связи с большой разработанностью теории для этого случая. Для конечного поля все определения, введенные выше в связи с линейными кодами, формируются почти

дословно так же, как и для двоичных кодов. В большинстве технических устройств кодовая последовательность символов алфавита подается на модулятор, превращающий эти символы в определенные функции от времени.

Могут быть использованы различные схемы модуляции. Например, для алфавита из символов 0, 1, 2, 3, 4 возможны следующие три типа модуляции: рис.3, рис 4, рис 5.

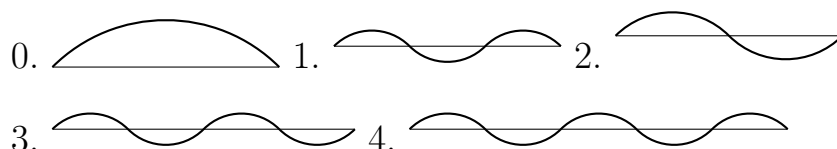


Рис. 3: Ортогональная модуляция

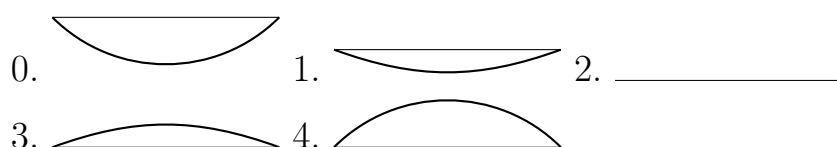


Рис. 4: Амплитудная модуляция

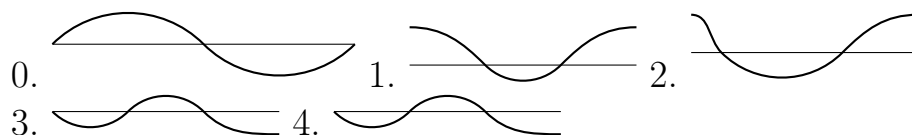


Рис. 5: Фазовая модуляция

Хотя символы алфавита можно представить многими различными способами, в двоичном случае их конкретный выбор несуществен для устройств и декодирования, которые оперируют лишь последовательностями символов алфавита, которые кодер посылает на модулятор, а декодер получает с демодулятора. С другой стороны, в недвоичном случае выбор схемы модуляции может отражаться на характере возможных ошибок в канале. В связи с этим теория недвоичных кодов оказывается более сложной.

1.6. Упражнения

1. Сколько существует кодовых слов в двоичном коде с повторами, если число информационных символов $k = 3$, а блоковая длина $n = 9$?
2. Сколько существует двоичных слов длины n над алфавитом из d букв?
3. Сколько существует двоичных слов длины n , не содержащих двух нулей подряд?
4. Сколько существует d — буквенных слов длины n , не содержащих двух нулей подряд (нуль содержится среди букв алфавита)?
5. Сколько существует двоичных слов длины n с числом единиц, равным t , не содержащих двух нулей подряд?
6. Пусть двоичный симметрический канал используется для передачи сообщений, имеющих длину n . Какова вероятность того, что все символы сообщения будут приняты правильно? Какова вероятность ошибки во всех символах?
7. Какова должна быть вероятность q ошибки в передаче отдельного символа, чтобы при передаче по двоичному симметрическому каналу блока из 8 символов не менее половины было принято правильно с вероятностью не менее 0,9?
8. Показать, что вероятность ошибки q в двоичном симметричном канале можно считать меньшей чем $\frac{1}{2}$.
9. Для кода из примера 1.4 придумать алгоритм декодирования.
10. Чему равна скорость кода из примера 1.4?
11. Дана матрица H размера $(n - k) \times n$, состоящая из нулей и единиц. Чему равно число двоичных слов C длины n , удовлетворяющих соотношению $HC^t = 0$?
12. Показать, что для любого двоичного линейного кода с проверочной матрицей H существует такая кодирующая матрица G , что $HG^t = 0$.
13. Дать доказательства теорем 1 и 2.

2. Некоторые сведения из теории групп

Тут Неизвестно Кто вошел,
и при свете свечи он и Пух
установились друг на друга.
— Я — Пух, — сказал Винни-Пух.
— А я — Тигра, — сказал Тигра.

А. А. Милн. Винни-Пух
и все-все-все

Нам понадобятся некоторые определения и теоремы теории групп, которые будут использоваться в дальнейшем.

2.1. Определения

Определение 2. *Бинарной операцией на множестве M называется произвольная функция $f : M \times M \rightarrow M$.*

Другими словами, бинарная операция каждой упорядоченной паре элементов $x, y \in M$ ставит в соответствие элемент $z = f(x, y) \in M$.

Определение 3. *Множество G с бинарной операцией \star (результат z применения операции к элементам $x, y \in G$ записывается в виде $z = x \star y$) называется группой, если выполнены следующие аксиомы:*

1. $\forall x, y, z \in G : (x \star y) \star z = x \star (y \star z);$
2. $\exists e \in G : \forall x \in G : (e \star x) = (x \star e) = x;$
3. $\forall x \in G \exists y \in G : x \star y = y \star x = e.$

Определение 4. *Множество G с бинарной операцией называется абелевой (коммутативной) группой, если G — группа и выполнено следующее утверждение:*

$$\forall x, y \in G : x \star y = y \star x.$$

Определение 5. *Группа G называется конечной, если число ее элементов конечно. Порядок конечной группы G — число $|G|$ ее элементов.*

Для любого множества M через $|M|$ обозначается мощность этого множества. В случае конечной мощности — это просто число элементов.

В дальнейшем будем различать *мультипликативные* группы ($\star =$ умножение) и *аддитивные* ($\star =$ сложение). В первом случае знак операции, как правило, опускается. Если противное не оговаривается, то группа считается мультипликативной. Элемент e в этом случае записывается единицей — 1, а элемент y из аксиомы 3 (определение 3) обозначается как x^{-1} (обратный элемент).

Определение 6. Пусть G — группа, $M \subseteq G$. Говорят, что M — подгруппа G , если выполнены следующие условия:

1. $\forall x, y \in M : xy \in M$;
2. $x \in M : x^{-1} \in M$.

Определение 7. Пусть G — группа, M — ее подгруппа, $x \in G$. Тогда $Mx = \{gx | g \in M\}$ — правый смежный класс группы G по подгруппе M .

Определение 8. Пусть G, H — группы (операции в G и H обозначаются одинаково). Говорят, что группа $\Gamma = G \times H$, если $\Gamma = \{(x, y) | x \in G, y \in H\}$ и операция (умножения) на Γ задается следующим образом: $\forall z_1 = (x_1, y_1), z_2 = (x_2, y_2) \in \Gamma$ их произведение $z_3 = z_1 z_2 = (x_1 x_2, y_1 y_2)$.

Определение 9. Группы G и H называются изоморфными, если существует такая биекция $\varphi : G \rightarrow H$, что $\varphi(xy) = \varphi(x)\varphi(y)$ для любых $x, y \in G$.

Определения кольца и поля в [3, 6].

Определение 10. Группа G называется циклической (с образующим элементом x), если существует такой элемент $x \in G$, что для любого $y \in G$ элемент y является некоторой степенью x .

2.2. Примеры

2.2.1.

Наиболее распространенные примеры групп — это поля. Именно $(\mathbb{Q}, +)$; $(\mathbb{R}, +)$; $(\mathbb{C}, +)$ — группы (их называют аддитивными группами соответствующих полей). Стандартная запись: $\mathbb{Q}^+, \mathbb{R}^+, \mathbb{C}^+$. Множество целых чисел по сложению — также группа \mathbb{Z}^+ . Пусть F — поле, $F^* = F \setminus \{0\}$. Тогда F^* — группа по умножению (мультипликативная группа поля F). Отметим, что $\mathbb{Z} \setminus \{0\}$ не является группой по умножению.

2.2.2.

Приведенные выше группы являются абелевыми (коммутативными) группами и \mathbb{Q}^+ , \mathbb{R}^+ , \mathbb{C}^+ , \mathbb{Q}^* , \mathbb{R}^* , \mathbb{C}^* , \mathbb{Z}^+ . Понятно, что абелевыми являются и циклические группы. Распознать абелевость группы легко по ее таблице умножения (таблице Кэли).

2.2.3.

Пусть $\sqrt[n]{1}$ — множество всех корней n -й степени из 1 в \mathbb{C} , рассматриваемое по умножению. Тогда $\sqrt[n]{1}$ — группа, состоящая ровно из n элементов. При этом $\sqrt[n]{1}$ — циклическая группа. Ее образующий элемент — первообразный корень n -й степени из единицы. Группа $\sqrt[n]{1}$ изоморфна \mathbb{Z}_n^+ (аддитивной) группе вычетов по модулю n . При этом изоморфизм осуществляется соответствием φ :

$$\forall k \in \{0, 1, 2, \dots, n-1\} \quad \varphi(k) = \cos \frac{\alpha \pi k}{n} + i \sin \frac{\alpha \pi k}{n}.$$

2.2.4.

Примером группы по сложению может служить также любое линейное векторное пространство над произвольным полем F . В частности, если $F = \mathbb{Z}_2$ и

$$V_n = \{(\alpha_1, \alpha_2, \dots, \alpha_n) \mid \alpha_i \in F, (i = 1, 2, 3, \dots, n)\},$$

то $(V_n, +)$ — абелева группа, являющаяся прямой суммой n циклических групп порядка 2, $|V_n| = 2^n$.

2.2.5.

Диэдральная группа D_n состоит из симметрий правильного n -угольника. Умножение = последовательное выполнение симметрий. Нетрудно вычислить, что $|D_n| = 2n$. Группа $D_n (n > 2)$ некоммутативна (вращения и отражения относительно осей симметрии, проходящих через центр, не коммутируют при $n > 2$).

2.2.6.

Предыдущий пример допускает следующее обобщение. Пусть $\Gamma = (G, V)$ — граф с множеством вершин G и ребер V . Вершины графа Γ нумеру-

ются числами $1, 2, \dots, n$, где $n = |G|$, а ребра — парами чисел: $(i, j) \in V$, если между вершинами i и j есть ребро. Под симметрией графа Γ понимается такая подстановка π его вершин, которая сохраняет ребра, т. е. если $(i, j) \in V$, то $(\pi(i), \pi(j)) \in V$. Под произведением симметрий понимается их последовательное выполнение. Группа всех симметрий графа Γ обозначается $Aut(\Gamma)$ и является подгруппой в группе всех подстановок n символов S_n . Напомним, что $|S_n| = n!$.

2.2.7.

Интересный и нетривиальный пример группы дают все движения кубика Рубика.

2.3. Теоремы

Группы, рассматриваемые в этом разделе, мультипликативные (т. е. по умножению). Простое упражнение: переписать текст раздела для аддитивных (по сложению) групп.

Пусть G — конечная группа, M — ее подгруппа, $x \in G$ и $xM = \{xg | g \in M\}$ — левый смежный класс группы G по M с представителем x . Соответственно, $Mx = \{gx | g \in M\}$ — правый смежный класс группы G по M с представителем x .

Теорема 3. *Множество всех левых (правых) смежных классов группы G по ее подгруппе M образует разбиение множества G (т. е. любые два левых (правых) смежных класса либо не пересекаются, либо совпадают, а объединение всех левых (правых) смежных классов дает всю группу G).*

Определение 11. *Если множество левых (правых) смежных классов группы G по ее подгруппе M конечно, то их число называется индексом группы M в подгруппе G и обозначается $|G : M|$. Оно не зависит от того, рассматриваются ли левые или правые смежные классы.*

Нетрудно проверить, что любые два левых (правых) смежных класса группы G по одной и той же ее подгруппе состоят из одинакового числа элементов. Поэтому справедлива следующая

Теорема 4 (Лагранж). *Для любой конечной группы G и ее подгруппы M справедливо следующее соотношение $|G| = |G : M| |M|$.*

Порядком элемента x конечной группы G назовем такое наименьшее положительное целое число m , что $x^m = 1$. Очевидно, что множество всех степеней элемента x образует циклическую подгруппу H группы G (порожденную x). Обозначение $H = \langle x \rangle$. При этом m есть порядок (число элементов) циклической подгруппы $H = \langle x \rangle$, порожденной x .

В частности, справедлива

Теорема 5. *Порядок элемента конечной группы G делит порядок группы G . Если $x \in G$, то $x^{|G|} = 1$.*

В качестве следствия имеем такое утверждение:

Теорема 6. *Всякая группа простого порядка является циклической.*

Циклические подгруппы в группах играют немаловажную роль. Отметим следующий факт.

Теорема 7. *Всякая подгруппа циклической группы является циклической. Любая циклическая группа порядка n изоморфна \mathbb{Z}_n^+ . Строение конечных абелевых групп полностью сводится к строению циклических групп и их прямых произведений.*

Теорема 8 (Основная теорема о конечных абелевых группах). *Всякая конечная абелева группа является прямым произведением циклических групп, порядки которых являются степенями простых чисел.*

С помощью теоремы 8 (с точностью до изоморфизма) можно перечислить все абелевы группы данного порядка. Например, с точностью до изоморфизма имеется лишь 2 типа абелевых групп порядка 60, именно: $\mathbb{Z}_3^+ \times \mathbb{Z}_4^+ \times \mathbb{Z}_5^+$ и $\mathbb{Z}_3^+ \times \mathbb{Z}_2^+ \times \mathbb{Z}_2^+ \times \mathbb{Z}_5^+$.

2.4. Упражнения

1. Доказать теоремы 3-7 из п. 2.3
2. Определим центр группы G следующим образом $Z(G) = \{x \in G \mid \forall g \in G \ xg = gx\}$. Доказать, что $Z(G)$ — подгруппа группы G и $Z(G) = G \Leftrightarrow G$ — абелева группа.
3. Найти все подгруппы групп S_3 и D_4 .

4. Показать, что множество всех невырожденных $n \times n$ -матриц с коэффициентами из поля F образует группу $GL(n, F)$. Найти порядок $GL(n, \mathbb{Z}_2)$.
5. Найти среди чисел $n \leq 25$ такие, что любая группа порядка n будет циклической.
6. Найти порядок группы движений кубика Рубика.
7. Дать оценку наибольшего порядка элемента группы S_n .
8. Показать, что группа G абелева, если $g^2 = 1$ для любого элемента группы $g \in G$.
9. Будет ли группой по сложению множество всех целых чисел, делящихся на фиксированное число n ?
10. Пусть на множестве всех ненулевых вещественных чисел \mathbb{R}^* задана операция \circ по следующему правилу: $x \circ y = x/y$. Будет ли множество \mathbb{R}^* , \circ — группой?
11. Пусть G — конечная группа, M — ее подгруппа, $x \in G$ и $xM = \{xg | g \in M\}$ — левый смежный класс группы G по M с представителем x . Доказать, что существует такая система представителей $\{x_1, x_2, x_3, \dots, x_k\}$, где $k = |G : M|$, что $G = \cup_{i=1}^k Mx_i = \cup_{i=1}^k x_iM$.
12. В каком случае \mathbb{Z}_n (с операциями сложения и умножения) будет полем?
13. Доказать, что для любого делителя m числа n в циклической группе порядка n существует одна и только одна подгруппа порядка m .
14. Верно ли обращение теоремы Лагранжа?
15. Доказать малую теорему Ферма: если p — простое число и a — такое целое число, что $(p, a) = 1$, то $a^{p-1} - 1$ делится на p .
16. Верно ли обращение малой теоремы Ферма?
17. Сколько подгрупп у аддитивной группы n -мерного векторного пространства над \mathbb{Z}_2 ?
18. Найти число попарно неизоморфных абелевых групп порядка 720.

19. Найти число попарно неизоморфных абелевых групп порядка 1024.
20. Существуют ли бесконечные группы, все нетривиальные (т. е. отличные от группы \mathbb{Z}^+) подгруппы которых имеют конечный индекс?

3. Оценки качества кода.

Подход к решению задачи

— Пожалуй, местами он даже лучше!

— Гораздо лучше! — хором сказали

Пух и Пятачок.

— Вот вам пример того, что можно

сделать, если не лениться, — сказал Иа. — Тебе понятно, Пух?

Тебе понятно, Пятачок? Во-первых,

Смекалка, и во-вторых, —

Добросовестная Работа. Ясно?

А. А. Милн. Винни-Пух

и все-все-все

3.1. Геометрическая интерпретация

Пусть V_n — множество всех слов над алфавитом $\{0, 1\}$ длины n . Код \mathcal{K} — некоторое подмножество V_n . Нас интересует вопрос о том, как должен быть расположен код \mathcal{K} в V_n , чтобы быть устойчивым относительно не слишком больших помех.

Для того чтобы уяснить ситуацию, обратимся к геометрической интерпретации. Будем рассматривать V_n как гиперкуб в пространстве \mathbb{R}^n . Точнее, множество всех слов длины n можно отождествить с вершинами этого гиперкуба.

Пусть $a \in V_n$. Если лишь в одной позиции вектора a произойдет искажение, то вектор a превратится в вектор a' , лежащий на одном с ребре a (т. е. a' будет расположен в соседней вершине). Если число искаженных позиций будет не более t , то между исходным вектором a и искаженным вектором a' существует путь по ребрам гиперкуба длины $\leq t$. Таким образом, если у нас есть уверенность в том, что большого числа искажений при передаче быть не может, естественно разместить вершины, отвечающие кодовым словам кода \mathcal{K} , чтобы они были расположены друг от друга как можно дальше.

Попробуем найти хорошие коды при $n = 3$ и $n = 2$.

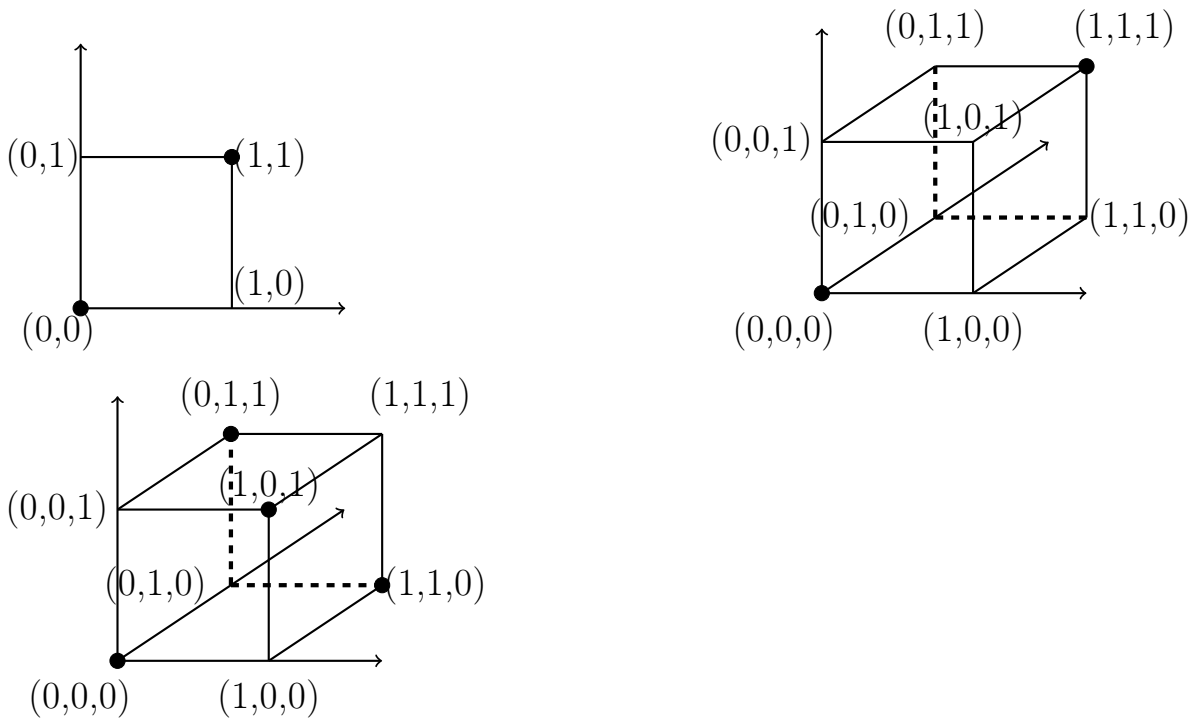


Рис. 6: Коды при $n = 2$ и $n = 3$

Решения представлены на рис. 6. При $n = 2$ имеется (с точностью до симметрии) единственное решение $\mathcal{K}_0 = \{(0,0), (1,1)\}$. При $n = 3$ имеется (опять с точностью до симметрии) два решения: $\mathcal{K}_1 = \{(0,0,0), (1,1,1)\}$ и $\mathcal{K}_2 = \{(0,0,0), (0,1,1), (1,1,0), (1,0,1)\}$. Любое из указанных решений содержит наибольшее возможное число кодовых слов, отвечающих одному из требований:

- а) ни одна из одиночных ошибок не пройдет незамеченной (коды \mathcal{K}_0 и \mathcal{K}_2) или
- б) можно исправить любую одиночную ошибку (код \mathcal{K}_1).

В случае кодов \mathcal{K}_0 и \mathcal{K}_1 мы без труда узнаем знакомые нам коды с повтором, а в случае \mathcal{K}_2 — код с проверкой на четность (код \mathcal{K}_0 также будет кодом с проверкой на четность). Теперь становится понятным, почему эти коды обладают определенными корректирующими способностями.

Формализуем теперь интуитивно понятную причину успеха в конструировании кодов $\mathcal{K}_0, \mathcal{K}_1, \mathcal{K}_2$.

Пусть $a = (\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n) \in V_n$ и $w(a) = \sum_{i=1}^n \alpha_i$ ($\alpha_i \in \mathbb{Z}_2$). Функция $w : V_n \rightarrow R$, определенная только что, называется *весом Хэмминга* (слова a). Если теперь использовать операцию сложения в V_n , как в векторном пространстве над полем \mathbb{Z}_2 , то расстояние Хэмминга между словами $a, b \in V_n$ можно определить следующим образом:

$$\rho(a, b) = w(a - b).$$

Иными словами, $\rho(a, b)$ — число позиций, в которых слова a и b не совпадают. Это в точности минимальная длина пути между a и b по ребрам гиперкуба V_n . Введенное таким образом расстояние отвечает всем требованиям, предъявляемым к расстояниям в математике, т. е. справедлива

Теорема 9.

- 1) $\forall a, b \in V_n \rho(a, b) = \rho(b, a)$;
- 2) $\forall a, b \in V_n \rho(a, b) = 0 \Leftrightarrow a = b$;
- 3) $\forall a, b, c \in V_n \rho(a, b) + \rho(b, c) \geq \rho(a, c)$.

Если в коде $\mathcal{K} \subseteq V_n$ существуют два слова, расстояние между которыми равно единице, то можно подобрать такую одиночную ошибку, что она переводит одно кодовое слово в другое. Поэтому-то такой код не способен обнаруживать любую одиночную ошибку. В общем случае справедлива

Теорема 10. *Для того чтобы двоичный код \mathcal{K} с блоковой длиной $n \geq t + 1$ мог обнаруживать любые ошибки в $\leq t$ позициях, необходимо и достаточно, чтобы минимальное расстояние между двумя словами было не меньше $t + 1$.*

Например, $(n, 1)$ -код с повторением может обнаруживать до $n - 1$ ошибки. Код \mathcal{K}_2 , рассмотренный выше, обнаруживает любую одиночную ошибку.

Мы получили возможность ввести одну из наиболее важных характеристик корректирующей способности кода — его минимальное расстояние.

Определение 12. Минимальное расстояние кода \mathcal{K} называется числом $d(K) = \min_{\substack{a, b \in \mathcal{K} \\ a \neq b}} \rho(a, b)$.

Минимальные расстояния кодов $\mathcal{K}_0, \mathcal{K}_1, \mathcal{K}_2$ равны соответственно 2, 3 и 2.

Теорема 11. Для того чтобы двоичный код $\mathcal{K} \subseteq V_n$ с блоковой длиной $n \geq 2t + 1$ давал возможность направлять все ошибки в $\leq t$ позициях, необходимо и достаточно, чтобы минимальное расстояние этого кода было не меньше $2t + 1$.

Доказательство. Пусть $S_t(a) = \{b \in V_n | \rho(a, b) \leq t\}$ — сфера радиуса t , описанная вокруг слова $a \in \mathcal{K}$ и пусть $S_t(a') = \{b \in V_n | \rho(a', b) \leq t\}$ — сфера радиуса t , описанная вокруг слова $a' \in \mathcal{K}$. Пусть $\rho(a, a') \geq 2t + 1$.

Из теоремы 9 и условия теоремы 10 следует, что эти сферы не пересекаются. Если под действием шума слово $a \in \mathcal{K}$ подверглось искажению, но число искаженных позиций не превосходит t , то полученное слово $a' \in S_t(a)$. Следовательно, мы можем декодировать слово a' с помощью ближайшего к нему кодового слова a .

Обратно, если существуют два слова $a, b \in \mathcal{K}$, для которых $\rho(a, b) \leq 2t$, то существует такое слово $c \in V_n$, что $\rho(a, c) \leq t$ и $\rho(b, c) \leq t$. Поэтому можно подобрать так шум, искажающий $\leq t$ позиций слова a , что получится слово c . Однако информации о том, что произошло t ошибок, явно недостаточно, чтобы однозначно декодировать слово c (оно могло получиться и в результате искажения слова b). \square

Из сказанного следует, что имеются следующие естественные задачи в теории линейных кодов, исправляющих ошибки.

А. По заданным параметрам n (блоковая длина) и k (число информационных символов) построить код, обладающий наибольшим возможным расстоянием.

Б. По заданным минимальному расстоянию d и числу информационных символов k (или данному числу кодовых слов) построить код, обладающий наименьшей блоковой длиной (обладающий наименьшей избыточностью, или, что одно и то же, наибольшей скоростью).

В. Что можно сказать о соотношениях, связывающих параметры n, k и d ?

Это — наиболее интригующие задачи в теории кодирования. К сожалению, полное решение их до сих пор не получено. Во всяком случае,

большая часть этой бурно развивающейся науки направлена на решение этих задач. Помимо этого, возникают различные вспомогательные, но весьма важные проблемы, связанные с простотой, стоимостью и эффективностью реализации разработанных алгоритмов. Наконец, если теория двоичных кодов достаточно развита, этого еще нельзя сказать об обобщениях на случай не двоичных алфавитов.

Из дальнейшего будет ясно, как решаются указанные задачи в некоторых конкретных случаях.

3.2. Групповые коды

Определение 13. Пусть V_n — n -мерное векторное пространство над полем \mathbb{Z}_2 . Код $\mathcal{K} \subseteq V_n$ называется групповым, если множество элементов этого кода образует группу по сложению.

Теорема 12. Если код $\mathcal{K} \subseteq V_n$ является групповым, то минимальное расстояние кода \mathcal{K} равно наименьшему весу ненулевых слов кода \mathcal{K} .

Доказательство. Очевидно из соотношения $\rho(a, b) = w(a - b)$ и аксиом группы. \square

Теорема 12 сильно облегчает вычисление минимального расстояния кода. Действительно, если код \mathcal{K} имеет M слов, то для определения минимального расстояния необходимо в общем случае посчитать всевозможные попарные расстояния между кодовыми словами (всего $\frac{M(M-1)}{2}$ расстояний). В случае же группового кода надо лишь перебрать $M - 1$ ненулевых элементов кода. Отметим, что любой линейный код является групповым. Так как V_n , множество всех двоичных слов длины n , также является группой по сложению, то множество всех слов группового кода $\mathcal{K} \subseteq V_n$ будет подгруппой группы V_n .

Введем теперь полезное усовершенствование в нашу схему. Будем считать, что при передаче кодового слова $c \in \mathcal{K}$ шум в канале просто прибавляет к нему *шумовое слово* e . В результате получается принятое слово y , т. е.

$$y = c + e.$$

Эта модель позволяет объяснить, какие ошибки при передаче останутся незамеченными.

Теорема 13. Если код \mathcal{K} является групповым, то ошибка передачи останется не замеченной в том и только в том случае, когда соответствующее шумовое слово является кодовым.

Допустим, что мы имеем дело с двоичным симметричным каналом, в котором ошибки проходят независимо с вероятностью $q < \frac{1}{2}$. Воспользовавшись схемой Бернулли, легко убедиться, что имеет место

Теорема 14. Вероятность появления в канале фиксированного шумового слова e веса w при передаче блока длины n равна $q^w(1 - q)^{n-w}$. Вероятность того, что при передаче произойдет ровно w ошибок, равна $C_n^w q^w(1 - q)^{n-w}$.

Так как $q < \frac{1}{2}$, то из теоремы 14 ясно, что вероятность появления фиксированного шумового слова веса t меньше, чем вероятность появления фиксированного шумового слова веса $t - 1$ и т. д.

Поэтому стратегия декодирования по максимуму правдоподобия сводится к следующему: если принятое слово — y , то выбирается вектор ошибок e (шумовое слово) с наименьшим весом и декодируется в такое кодовое слово c , что $y = c + e$. Декодирование полным перебором заключается в сравнении y со всеми словами группового кода \mathcal{K} и выборе ближайшего кодового слова. Если число кодовых слов невелико, эта стратегия разумна. Но если их много, то она не годится. Одной из целей теории кодирования является разработка быстрых алгоритмов декодирования.

В связи со сказанным выше возникает понятие *стандартного расположения кода* (таблица декодирования). Пусть код $\mathcal{K} \subseteq V_n$ состоит из слов $c_1 = 0, c_2 = 0, \dots, c_m$. Если декодер принимает слово $y \in V_n$, то возможный вектор ошибок должен принадлежать смежному классу $\mathcal{K} + y$ (см. определение 7, отметим, что здесь запись аддитивна). С другой стороны, ясно, что для любого $g \in \mathcal{K} + y$ существует такое слово $c \in \mathcal{K}$, что $c + g = y$. Верное и обратное, если $c' + g' = y$, где $c' \in \mathcal{K}$, то $\mathcal{K} + g' = \mathcal{K} + g$, т. е. возможные векторы ошибок — это в точности элементы смежного класса $\mathcal{K} + y$.

Расположим все кодовые слова в первой строке таблицы:

$$c_1 = 0, c_2, c_3, \dots, c_m.$$

Далее выписываются элементы смежных классов:

$$y, c_2 + y, c_3 + y, \dots, c_m + y.$$

При этом в первый столбец записывается слово наименьшего веса в данном смежном классе. Оно называется *лидером* смежного класса, а первый столбец таблицы — *столбцом лидеров*.

Приведем примеры.

Стандартное расположение кода \mathcal{K}_1 .

Код	(0, 0, 0)	(1, 1, 1)
Смежный класс	(0, 0, 1)	(1, 1, 0)
Смежный класс	(0, 1, 0)	(1, 0, 1)
Смежный класс	(1, 0, 0)	(0, 1, 1)

Аналогичным образом можно привести стандартное расположение для кода из примера 1.2.2

Код	(0, 0, 0, 0)	(0, 0, 1, 1)	(0, 1, 0, 1)	(0, 1, 1, 0)	(1, 0, 0, 1)
	(1, 0, 1, 0)	(1, 1, 0, 0)	(1, 1, 1, 1)		
Смежный класс	(0, 0, 0, 0)	(0, 0, 1, 1)	(0, 1, 0, 1)	(0, 1, 1, 0)	(1, 0, 0, 1)
	(1, 0, 1, 0)	(1, 1, 0, 0)	(1, 1, 1, 1)		

Обратим внимание на то, что выбор лидеров смежных классов во втором случае не однозначен.

Стандартное расположение используется для декодирования. Например, если принятое слово y для кода \mathcal{K} равно $(0, 1, 1)$, то, отыскав это слово в таблице и найдя в том же столбце и первой строке соответствующее кодовое слово, получим наиболее вероятное переданное слово $c \in \mathcal{K}$ (в данном примере $c = (1, 1, 1)$). Декодирование с помощью стандартного расположения будет декодированием по методу максимального правдоподобия.

Справедлива следующая

Теорема 15. *Групповой код \mathcal{K} со схемой декодирования посредством лидеров исправляет в точности те ошибки, которые являются лидерами смежных классов. При этом кодовое слово, стоящее в данном столбце, является ближайшим ко всем словам этого столбца.*

4. Вероятность ошибки декодирования

Определение 14. Вероятностью ошибки декодирования для данной схемы декодирования называется вероятность появления ошибки на выходе декодера.

Пусть имеется M кодовых слов C_1, C_2, \dots, C_M , которые могут быть посланы с равной вероятностью. Тогда

$$P_{\text{ош}} = \frac{1}{M} \sum_{i=1}^M P \{ \text{выход декодера} \neq C_i \mid C_i \text{ было послано} \}.$$

Если декодировать, пользуясь стандартным расположением, то $P_{\text{ош}} = P \{ l \neq \text{лидер смежного класса} \}$. Тогда, используя схему Бернулли и формулу полной вероятности, имеем

$$P_{\text{ош}} = 1 - \sum_{i=0}^M \alpha_i q^i (1 - q)^{n-i}$$

(n – блоковая длина слова, q – вероятность ошибки при передаче одного символа).

Если минимальное расстояние кода $d = 2t + 1$, то по теореме 11 он может исправлять $\leq t$ ошибок. Поэтому каждый вектор ошибок веса $\leq t$ будет лидером своего смежного класса. Отсюда $\alpha_i = C_n^i$ ($0 \leq i \leq t$).

В общем случае распределение чисел α_i известно для немногих кодов.

Если вероятность q в канале очень мала, то хорошей аппроксимацией для кодов с минимальным расстоянием $d = \alpha t + 1$ будет формула

$$P_{\text{ош}} \approx 1 - \sum_{i=0}^n C_n^i q^i (1 - q)^{n-i}.$$

Если $\alpha_i = 0$ при $i > t$, то эта формула становится точной, а соответствующий код называется *совершенным*.

Геометрически это означает, что все n -мерное пространство V_n двоичных последовательностей длины n разбито на непересекающиеся шары радиуса t (с центрами в кодовых словах). Таким образом, в теории кодирования возникает задача плотнейшей упаковки сфер.

Содержательно задача о наиболее плотной упаковке в теории кодирования может быть сформулирована так. Какое наибольшее число кодовых слов может содержать код с блоковой длиной n и минимальным расстоянием d ?

Лишь сравнительно недавно было доказано, что все совершенные коды известны [7]. В то же время существует большое число вопросов, относящихся к кодам, близким к совершенным; с некоторыми из них мы познакомимся вскоре.

Если код \mathcal{K} используется лишь для обнаружения ошибок, то декодер ошибается лишь в том случае, когда вектор ошибок является ненулевым кодовым словом. Поэтому если код содержит A_i кодовых слов веса i , то вероятность ошибки равна

$$P_{\text{ош}} = \sum_{i=1} A_i q^i (1-q)^{n-i}$$

(n – блоковая длина кода).

Так как некоторые из информационных символов могут быть правильными, даже если декодер выдает неправильное слово, то имеется другая полезная мера качества декодирования $P_{\text{симв}}$ – вероятность ошибки на символ.

Определение 15. Пусть код \mathcal{K} содержит M кодовых слов C_1, C_2, \dots, C_M . Первые k символов каждого слова $C_i = C_i^1, C_i^2, \dots, C_i^k$ являются информационными, и пусть $\hat{C} = (\hat{C}^1, \hat{C}^2, \dots, \hat{C}^n)$ – слово на выходе декодера. Тогда

$$P_{\text{симв}} = \frac{1}{kM} \sum_{j=1}^k \sum_{i=1}^M P(\hat{C}_j \neq C_i^j \text{ было послано}).$$

Если при декодировании используется стандартное расположение, то $P_{\text{симв}}$ не зависит от того, какое слово было послано [7]. Можно показать также, что для (n, k) -кода

$$\frac{1}{k} P_{\text{ош}} \leq P_{\text{симв}} \leq P_{\text{ош}}.$$

Для большинства кодов значение $P_{\text{симв}}$ неизвестно.

4.1. Понятие о теореме Шеннона

В общем случае было бы интересно узнать, сколь малым может быть сделано значение $P_{\text{симв}}$ для (n, k) -кода с фиксированной скоростью $R = \frac{k}{n}$. Ответ на этот вопрос был получен в 1948 году в знаменитой статье К. Шеннона, с которой, собственно, и началась теория кодов, исправляющих ошибки. Шеннон показал, что с каждым каналом связано измеряемое в битах в секунду число C , называемое пропускной способностью канала. Число $C(P) = 1 + p \log_2 p + q \log_2 q$ для двоичного симметрического канала.

Теорема 16 (Шеннон). Для любого $\varepsilon > 0$ и $R < C(P)$ существует такое N , что найдется двоичный (n, k) -код со скоростью передачи $\frac{k}{n} \geq R$, $n > N$, вероятность ошибки которого $P_{ош} < \varepsilon$.

Теорема Шеннона говорит лишь о существовании хороших кодов, но из нее никак нельзя извлечь способ их построения. Несмотря на явную неэффективность, теорема Шеннона показала, что с экономической точки зрения выгоднее строить системы связи, используя кодирование, чем улучшать технические характеристики канала. В скором времени Хеммингом были открыты коды, исправляющие одиночные ошибки (1950 год), а после небольшой заминки, после усиленных поисков в 1960 году были открыты коды, исправляющие кратные ошибки (Боуз, Рой-Чоудхури и Хоквингем).

4.2. Упражнения

1. Пусть по двоичному симметричному каналу передаются строки длины 10. Какова вероятность того, что ровно 3 символа будут приняты неправильно? Какова вероятность того, что не более 3 символов будут приняты неправильно? Сколько существует слов, отличающихся от данного не более, чем в 3 позициях?
2. Доказать, что двоичный (n, k) -код будет совершенным (для некоторого d) тогда и только тогда, когда для некоторого $m \in \mathbb{N}$ выполнено:

$$1 + n + \dots + C_n^m = 2^{n-k}.$$

3. Рассмотрим $(9, 8)$ -код с проверкой на четность. Какова вероятность того, что при приеме кодового слова не будет обнаружена ошибка передачи?
4. Пусть \mathcal{K} – $(9, 3)$ -код с повторением (троекратным). Какова вероятность того, что при приеме кодового слова не будет обнаружена ошибка передачи?
5. Найти множество из 16 двоичных слов длины 7, каждое из которых находится от другого на расстоянии не меньше чем 3.
6. Найти максимальное количество королей на доске размера $n \times n$, которые не бьют друг друга.

7. Можно ли найти 32 двоичных слова длины 8 таких, что каждое отличается от другого слова не меньше, чем в 3 позициях?
8. $(8, 4)$ -код \mathcal{K} состоит из слов вида $a = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta_1, \beta_2, \beta_3, \beta_4)$, где $\alpha_1 + \alpha_2 = \beta_1, \alpha_3 + \alpha_4 = \beta_2, \alpha_1 + \alpha_3 = \beta_3, \alpha_2 + \alpha_4 = \beta_4$. Является ли этот код групповым? Чему равно минимальное расстояние этого кода?
9. Чему равна вероятность того, что ошибки в сообщении останутся не обнаруженными для (n, k) -кода с минимальным расстоянием d ?
10. Определим для двоичных векторов x и y длины n их пересечения как вектор $x * y = (x_1 y_1, x_2 y_2, \dots, x_n y_n)$, где $x = (x_1, \dots, x_n)$, $y = (y_1, \dots, y_n)$. Показать, что
- $$w(x + y) = w(x) + w(y) - 2w(x * y).$$
11. Показать, что если u, v, w, x – четыре вектора, любая пара из которых находится друг от друга на расстоянии d , где d – четное число, то имеется в точности один вектор, который находится на расстоянии $\frac{d}{2}$ от каждого из векторов u, v и w .
12. Найти минимальное расстояние для кода (mk, k) с m – кратным повторением и для $(n, n - 1)$ -кода с проверкой на четность.
13. Порождающая матрица G кода \mathcal{K} имеет вид $G = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$. Найти его минимальное расстояние.

5. Снова о линейных кодах

Забравшись еще выше, он сказал (про себя):

«Конечно, это не так легко, что говорить!...»

А еще повыше он сказал:

— Ведь слезать тоже придется. Задом.

А еще-еще повыше:

— И это будет трудновато...

— Если не упасть...

А. А. Милн. Винни-Пух

и все-все-все.

5.1. Предварительные замечания

В п.1.3 мы определили (n, k) -код как множество n -мерных двоичных векторов C , удовлетворяющих соотношению $HC^t = 0$, где $H = \{0, 1\}$ — матрица размера $(n - k) \times n$, или (двойственным образом) как множество n -мерных двоичных векторов C , удовлетворяющих соотношению $C = UG$, где U — информационное слово длины k , а G — матрица с коэффициентами из $\{0, 1\}$ размера $k \times n$. При этом ничего не было сказано о структуре матриц H и G , кроме связывающего их соотношения $HG^t = 0$; $GH^t = 0$. Если в качестве проверочной матрицы H взять, например, нулевую матрицу размера $(n - k) \times n$, то соотношению $HC^t = 0$ будут удовлетворять все n -мерные двоичные векторы. Если в качестве кодирующей матрицы G взять нулевую, то множество векторов $C = UG$ будет состоять только из нулевого вектора. В обоих случаях предложенная модель находится в противоречии с интуицией. Действительно, чтобы исправлять ошибки, код \mathcal{K} должен быть собственным подмножеством в V_n (это требование нарушается в случае выбора нулевой матрицы V). Для того чтобы передавать различные сообщения, надо уметь восстанавливать по кодовому слову $c \in \mathcal{K}$ информационное слово U . Поэтому примем меры предосторожности, чтобы такие ситуации не появлялись.

Если V_k — множество всех k -мерных двоичных векторов, (информационных векторов) то выбор матрицы G (кодирующей матрицы) должен обеспечивать возможность однозначно находить прообраз кодового слова $UG = C$. Для этого необходимо (и достаточно) потребовать, что-

бы ранг матрицы G был равен k . Простейший прием, обеспечивающий это свойство, – сделать код систематическим.

Аналогичным образом ранг проверочной матрицы H должен быть равен $n - k$. Отметим, что роли матриц G и H различны. Матрица G служит для нахождения по информационному слову соответствующего кодового слова, а матрица H служит для выявления ошибок, возникших при передаче сообщения (т. е. кодового слова) по каналу связи и возможного последующего его исправления.

5.2. Корректирующая способность линейного кода

Пусть \mathcal{K} — линейный код с блоковой длиной n и проверочной матрицей H . Корректирующую способность кода можно оценить с помощью следующей теоремы.

Теорема 17. *Код \mathcal{K} имеет минимальное расстояние $\geq d$ тогда и только тогда, когда любые $d - 1$ столбец матрицы H линейно независимы (над полем \mathbb{Z}_2).*

Доказательство. По теореме 12 минимальное расстояние кода \mathcal{K} равно минимальному весу его ненулевых кодовых слов. Пусть \mathcal{K} имеет минимальное расстояние d и пусть $H = (h_1, h_2, \dots, h_n)$ — проверочная матрица кода \mathcal{K} , где h_1, h_2, \dots, h_n — столбцы матрицы H . Если некоторые из t столбцов матрицы H линейно зависимы, то

$$\varepsilon_{i1}h_{i1} + \varepsilon_{i2}h_{i2} + \dots + \varepsilon_{it}h_{it} = 0,$$

где $\varepsilon_{ij} \in \{0, 1\}$, $1 \leq i_1 < i_2 < \dots < i_t \leq n$, и существует вектор x , у которого на местах с номерами ij , где $\varepsilon_{ij} = 1$, стоят единицы, а остальные координаты — нули.

Для такого x , очевидно, $Hx^t = 0$, т. е. $x \in \mathcal{K}$. Так как $w(x) \geq d$, то $t \geq d$. Стало быть, любые $d - 1$ столбцов матрицы H линейно независимы (над полем \mathbb{Z}_2).

Обратно, пусть любые $d - 1$ столбец матрицы H линейно независимы над \mathbb{Z}_2 . Если $c \in \mathcal{K}$, то из сказанного выше следует, что

$$w(C) \geq d.$$

□

Теорема 18 (Граница Синглтона). Пусть \mathcal{K} — линейный (n, k) -код с минимальным расстоянием d . Тогда $n - k \geq d - 1$.

Доказательство. Число $r = n - k$ — ранг проверочной матрицы H . Так как максимальное число линейно независимых столбцов матрицы H не может превосходить $n - k$, то

$$d - 1 \leq n - k.$$

□

Коды, для которых эта граница достигается, т. е. $n - k = d - 1$, называются МДР-кодами (разделимыми кодами с минимальным расстоянием). Для двоичных кодов их нетрудно перечислить — это в точности $(n, 1)$ -коды с повтором и $(n, n - 1)$ -коды с проверкой на четность. Существует естественное обобщение МДР-кодов на недвоичный случай. Здесь ситуация сложнее [7].

5.3. Коды Хэмминга

Построим один специальный класс кодов, исправляющих одиночные ошибки, обладающий многими важными свойствами, — код Хэмминга. Из теоремы 12 следует следующее простое утверждение.

Теорема 19. *Линейный код исправляет все одиночные ошибки тогда и только тогда, когда все столбцы его проверочной матрицы различны между собой и отличны от нуля.*

Возникает естественный вопрос: как использовать матричный код для исправления одиночных ошибок? Обратимся к исходной модели.

Пусть c — кодовое слово линейного (n, k) -кода \mathcal{K} , а $H = (h_1, h_2, \dots, h_n)$ — проверочная матрица \mathcal{K} , e — шумовое слово. Так как мы интересуемся одиночными ошибками, то будем предполагать, что вес вектора ошибок e равен 1. Пусть $y = c + e$ — принятое слово. Тогда $Hy^t = Hc^t + He^t$, ибо $Hc^t = 0$. Пусть $e = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$ и $\varepsilon_i = 1$, а все остальные ε_j ($1 \leq j \neq i \leq n$) равны нулю. Тогда $He^t = \varepsilon_1 h_1 + \dots + \varepsilon_n h_n = h_i$. Таким образом, $Hy^t = h_i$, где i — номер позиции, в которой произошла ошибка. Алгоритм исправления ошибки теперь очень просто построить. Для этого надо вычислить Hy^t , где y — принятое слово. Если $h_i = Hy^t$, то ошибка произошла в i -ой позиции. Исправляем ошибку и получаем кодовое слово $c = y - e$.

В связи со сказанным имеет смысл ввести

Определение 16. Пусть H — проверочная матрица линейного (n, k) -кода, R — принятое слово. Тогда слово S , вычисляемое по формуле $S = HR^t$, будет называться синдромом ошибки.

Теорема 20. Пусть \mathcal{K} — линейный (n, k) -код с проверочной матрицей H . Синдром ошибок S равен сумме столбцов матрицы H , соответствующих тем позициям, в которых произошли ошибки. Два вектора x, y находятся в одном смежном классе тогда и только тогда, когда они имеют один и тот же синдром.

Из теоремы 20 следует, что существует взаимнооднозначное соответствие между синдромами и смежными классами V_n по \mathcal{K} . Синдром содержит всю информацию, которую имеет приемник об ошибках.

Если все столбцы матрицы H , выбранной в соответствии с теоремой 12, различны, то между столбцами матрицы H и одиночными ошибками существует взаимнооднозначное соответствие. Легко поэтому построить алгоритм исправления ошибок по синдрому. Отметим, что этот алгоритм, вообще говоря, будет неполным (могут быть ошибки кратности большей чем 1, тогда синдром не равен ни одному из столбцов матрицы H).

Поставим теперь такую задачу. Найти код наибольшей длины с данным числом проверочных символов r , исправляющих любую одиночную ошибку. Понятно, что существует $2^r - 1 = n$ столбцов r -мерных векторов, удовлетворяющих теореме 17. Поэтому существует проверочная матрица H размера $r \times 2^r - 1$, решающая поставленную задачу. Соответствующий код (открытый в 1950 году) принято называть *кодом Хэмминга*.

5.3.1.

Пример. Построим код Хэмминга для $n = 7$. Проверочная матрица кода

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Допустим, что принятое слово $y = (1, 0, 1, 0, 1, 0, 0)$. Покажем, как исправить одиночную ошибку в этом слове. Синдром ошибок

$$S = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Отсюда видно, что S совпадает с седьмым столбцом проверочной матрицы H . Следовательно, ошибка произошла в седьмой позиции. Подобным образом обстоит дело с кодами Хэмминга для больших размерностей.

Для того чтобы сделать процедуру исправления ошибок более изящной, будем интерпретировать синдром ошибок как номер позиции, в которой произошла ошибка. Например, в предыдущем случае (пример 5.3.1)

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \leftrightarrow 1 \cdot 2^2 + 1 \cdot 2 + 1 = 7.$$

Конечно, таким образом упорядочиваются и столбцы проверочной матрицы. Как видно из построения кода Хэмминга, его блоковая длина $n = 2^r - 1$, а число информационных символов $k = 2^r - r - 1$.

В качестве проверочных удобно взять символы с номерами $2^0 = 1, 2^1, 2^2, \dots, 2^{r-1}$, а в качестве информационных — остальные. Пусть дано информационное слово $U = (\alpha_1, \alpha_2, \dots, \alpha_k)$. Тогда составляется кодовое слово $C = (\beta_1, \beta_2, \dots, \beta_n)$ с неопределенными вначале элементами $\beta_1, \beta_2, \beta_2, \dots, \beta_{2^r-1}$, которые находятся из системы уравнений $HC^t = 0$. Так как в каждой строке H имеется лишь один ненулевой элемент, соответствующий проверочному символу, то эта система имеет единственное решение, которое легко находится.

5.3.2.

Пример. Информационный вектор $U = (1, 0, 1, 1)$, требуется найти кодовое слово, соответствующее этому информационному. Коэффициенты $\beta_1, \beta_2, \beta_4$ определим из системы уравнений $HC^t = 0$, т.е.

$$\begin{cases} \beta_4 + 1 \cdot 0 + 1 \cdot 1 + 1 \cdot 1 = 0 \\ \beta_2 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 = 0 \\ \beta_1 + 1 \cdot 0 + 1 \cdot 1 + 1 \cdot 1 = 0 \end{cases}$$

Отсюда $\beta_4 = 0, \beta_2 = 1, \beta_1 = 0$ и $C = (0, 1, 1, 0, 0, 1, 1)$.

Теорема 21. $(\alpha^r - 1, \alpha^r - r - 1)$ — код Хэмминга является совершенным. Минимальное расстояние кода Хэмминга равно 3.

Доказательство. Действительно, в проверочной матрице кода Хэмминга любые два столбца линейно независимы, и существуют три линейно независимых столбца (например, столбцы, соответствующие первым трем позициям). По теореме 17 минимальное расстояние кода Хэмминга равно 3. \square

Для того чтобы убедиться, что код Хэмминга является совершенным, надо показать, что шары радиуса 1, описанные вокруг каждого кодового слова, попарно не пересекаются и их объединение дает все множество двоичных слов длины $n = 2^r - 1$.

Первое уже доказано. Займемся подсчетом числа слов в шарах. Шар радиуса 1, описанный вокруг каждого кодового слова, соответствует одной позиции. Таких позиций будет $n = 2^r - 1$. Поэтому общее число двоичных слов в каждом шаре радиуса 1 равно $n + 1 = 2^r$. Всего кодовых слов 2^k , где $k = 2^r - r - 1$ — число информационных символов. Поэтому общее число двоичных слов, попавших во все шары радиуса 1, равно

$$(n + 1)2^k = 2^r 2^{2^r - r - 1} = 2^{2^r - 1} = 2^n,$$

т. е. объединение всех шаров радиуса 1, описанных вокруг кодовых слов, дает все пространство V_n .

Коды Хэмминга являются высокоскоростными. Действительно, скорость кода Хэмминга $R = \frac{k}{n} = \frac{2^r - r - 1}{2^r - 1} = (1 - \frac{r}{2^r - 1}) \rightarrow 1$ при $r \rightarrow 1$. Таким образом, по сравнению с кодами с проверкой на четность, коды Хэмминга не намного сложнее, но обладают существенно большими корректирующими способностями. Кроме того, для кодов Хэмминга имеется простой алгоритм кодирования и декодирования.

Для того чтобы иметь возможность выявлять ошибки более высокой кратности используется *расширенный код Хэмминга*. Чтобы его построить, применяется следующая конструкция.

Пусть $C = (\alpha_1, \alpha_2, \dots, \alpha_n)$ — кодовое слово кода Хэмминга ($n = 2^r - 1$) с проверочной матрицей H . Добавим к матрице H еще один столбец $(0, 0, \dots, 1)^t$ и одну (последнюю) строку $(1, 1, \dots, 1)$. Тогда получится матрица \hat{H} — проверочная матрица расширенного кода Хэмминга. Полученный код имеет блоковую длину $n + 1 = 2^r$ и то же число информационных символов $k = 2^r - r - 1$. Соответствующее кодовое слово есть слово

$\hat{C} = (\alpha_0, \alpha_1, \dots, \alpha_n)$, где символы α_i удовлетворяют дополнительному соотношению

$$\alpha_0 + \alpha_1 + \alpha_2 + \dots + \alpha_n = 0.$$

Алгоритм (неполного) декодирования для этого кода также достаточно прост. Пусть принятое слово $\hat{Y} = (\beta_0, \beta_1, \dots, \beta_n)$.

1. проверяем, выполнено ли соотношение $\hat{H}\hat{Y}^t = 0$. Если да, то считаем, что $Y = (\beta_1, \beta_2, \dots, \beta_n)$ — кодовое слово; 2. допустим, что случай 1) не выполнен. Тогда проверяем выполнение соотношения $\beta_0 + \beta_1 + \dots + \beta_n = 0$. Если оно выполнено, то слово \hat{Y} получилось в результате четного числа ошибок из кодового слова. Поэтому происходит отказ от декодирования. Если $\beta_0 + \beta_1 + \dots + \beta_n = 1$, то имеется нечетное число ошибок; 3. допустим, что число ошибок нечетно. Находим синдром ошибки для усеченного вектора $Y = (\beta_1, \dots, \beta_n)$. Если $S = HY^t = 0$, то ошибка в нулевой позиции. Если $S = HY^t \neq 0$, то декодируем слово так же, как это делалось в случае обычного кода Хэмминга.

5.3.3.

Пример. Матрица проверки на четность расширенного $(8, 4)$ кода Хэмминга:

$$\hat{H} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Требуется декодировать слово $\hat{Y} = (1, 1, 1, 0, 1, 0, 1, 0)$.

Решение. Найдем $\hat{S} = \hat{H}\hat{Y}^t = \begin{pmatrix} * \\ 1 \end{pmatrix}$. Так как последний элемент синдрома равен 1, то имеется нечетное число ошибок.

$$S = HY^t = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Поэтому надо исправить позицию в Y , соответствующую β_1 , т. е. искомое кодовое слово $\hat{C} = (1, 0, 1, 0, 1, 0, 1, 0)$. Для того чтобы теперь декодировать сообщение, нужно устранить проверочные символы, соответствующие 0, 1, 2 и 4 позициям. Таким образом, посланное сообщение — $(0, 0, 1, 0)$.

Заметим, что, хотя коды Хэмминга были открыты давно (в 1950 году), понадобилось около 10 лет, прежде чем были открыты коды, исправляющие две ошибки и более.

5.4. Техника матричного кодирования и декодирования

Как уже отмечалось, матричный код может быть задан проверочной матрицей H размера $(n - k) \times n$. В этом случае все кодовые слова — в точности решения системы линейных уравнений

$$HX^t = 0.$$

Множество всех решений этой системы линейных уравнений образует в пространстве V_n подпространство \mathcal{K} размерности k (напомним, что ранг H равен $n - k$). Поэтому существует базис \mathcal{K} , состоящий из k векторов g_1, g_2, \dots, g_k (фундаментальная система решений однородной системы с матрицей H). Тогда из строк g_1, g_2, \dots, g_k можно составить матрицу G . Если $U = (u_1, u_2, \dots, u_k)$ — произвольный k -мерный двоичный вектор, то

$$UG = C = u_1g_1 + u_2g_2 + \dots + u_kg_k \in \mathcal{K}.$$

Таким образом, мы имеем возможность сопоставить каждому сообщению U кодовое слово C умножением на матрицу G . Матрица G и есть порождающая матрица кода \mathcal{K} . Для того чтобы символы сообщения не перемешивались, код можно сделать систематическим.

5.4.1.

Пример. Построим кодирующую матрицу G для раширенного $(8, 4)$ кода Хэмминга (пример 5.3.3). Нетрудно видеть, что искомая матрица G может быть выбрана следующим образом:

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

При этом 1, 2, 3, 5 позиции кодового слова C , которое будет получаться с помощью умножения на G , информационные. Остальные — проверочные.

Тогда, например, информационное слово $U = (1, 0, 1, 1)$ будет закодированно так:

$$UG = (1, 0, 1, 1) \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} = (1, 0, 1, 0, 1, 0, 1, 0).$$

При декодировании сначала будут исправлены неверные позиции, а затем прочитаны позиции с номерами 1, 2, 3, 5.

Для декодирования при помощи матричного кода полученного слова можно воспользоваться следующим алгоритмом:

- 1) вычислить синдром $S(R)$ принятого вектора R ;
- 2) по синдрому определить из таблицы лидеров смежных классов соответствующий лидер E ;
- 3) найти кодовое слово $C = R + E$;
- 4) найти по кодовому слову C информационное слово U .

Разумеется, алгоритм декодирования расширенного кода Хэмминга проще в реализации, так как он не требует запоминания таблицы лидеров смежных классов, которая может быть достаточно обширной. Поэтому реально существующие коды основаны в большинстве своем на иной технике, которая станет ясной из дальнейшего.

5.5. Дуальный код

Пусть V_n — по-прежнему n -мерное пространство двоичных слов длины n , $x = (\alpha_1, \dots, \alpha_n)$, $y = (\beta_1, \dots, \beta_n) \in V_n$. Определим скалярное произведение x и y обычным образом:

$$(x, y) = \sum_{i=1}^n \alpha_i \beta_i,$$

где операция сложения производится в поле \mathbb{Z}_2 . Если $(x, y) = 0$, то векторы x и y называются ортогональными.

Определение 17. Пусть \mathcal{K} — линейный (n, k) -код в V_n . Тогда дуальный (ортогональный) к нему код \mathcal{K}^\perp определяется следующим образом:

$$\mathcal{K}^\perp = \{U \in V_n \mid (U, C) = 0 \ \forall C \in \mathcal{K}\}.$$

Таким образом, дуальный код состоит из всех векторов, ортогональных к любому вектору из \mathcal{K} .

Если $\mathcal{K} \subseteq \mathcal{K}^\perp$, то код \mathcal{K} называется слабо самодуальным. Если же $\mathcal{K} = \mathcal{K}^\perp$, то \mathcal{K} (строго) *самодуальный*. Например, самодуальным является расширенный $(8, 4)$ -код Хэмминга.

Теорема 22. Пусть \mathcal{K} — линейный (n, k) -код с проверочной матрицей H и кодирующей матрицей G . Тогда \mathcal{K}^\perp является $(n, n - k)$ -кодом с кодирующей матрицей H и проверочной матрицей G .

Доказательство. Доказательство следует из определения дуального кода, проверочной матрицы и кодирующей матрицы с учетом замечаний в п. 5.1 \square

Отметим следующий любопытный факт. Коды $(n, 1)$ -код с повторением и $(n, n - 1)$ -код с проверкой на четность дуальны друг другу. Код $(2, 1)$ с повторением самодуален. Если n четно, то $(n, 1)$ -код с повторением слабо самодуален.

Операция дуализации на множестве кодов оказывается очень полезной. Она приводит зачастую к построению новых важных кодов. Например, код Рида-Маллера первого порядка определяется как дуальный код к расширенному коду Хэмминга.

5.6. Упражнения

1. Определить положение одиночной ошибки в искаженном слове $(1, 1, 0, 0, 0, 1, 1)$ кода Хэмминга.
2. Пусть $(1, 1, 0, 1, 0, 1, 1)$ и $(1, 1, 0, 0, 1, 1, 1, 1)$ — искаженные слова расширенного кода Хэмминга. Какое из этих слов содержит одиночную, а какое — двойную ошибку? В случае одиночной ошибки определить ее положение.
3. Построить проверочную матрицу для кода Хэмминга с блоковой длиной 15. Сколько кодовых слов содержит этот код? Сколько информационных и сколько проверочных символов в кодовом слове?
4. Чему равно минимальное расстояние для расширенного кода Хэмминга?

5. Найти проверочную матрицу для кода, дуального к расширенному коду Хэмминга с блоковой длиной 16.
6. Пусть линейный двоичный код \mathcal{K} содержит хотя бы одно слово нечетного веса. Доказать, что число таких слов составляет ровно половину от числа всех кодовых слов (указание: воспользоваться теоремой Лагранжа).
7. Как изменится расстояние двоичного линейного кода при добавлении ко всем строкам проверочной матрицы одной общей проверки на четность.
8. Двоичный $(8, 4)$ -код задан проверочной матрицей

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

Найти его кодирующую матрицу и минимальное расстояние.

9. Построить $(14, 10)$ -код с исправлением одиночных ошибок, дав таблицу кодовых слов. Какова вероятность правильного приема (при двоичном симметричном канале)?
10. Показать, что $(3n, n)$ -код с декодированием «по большинству голосов» не может быть совершенным.
11. Пусть код \mathcal{K} имеет проверочную матрицу $H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$. Задать таблицей все сообщения и их коды.
12. Вычислить вероятность обнаружения и исправления ошибок для кода из предыдущей задачи.
13. Пусть код задан кодирующей матрицей

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

и используется для обнаружения ошибок, канал двоичный симметричный. Какова вероятность ошибочного приема?

14. Код \mathcal{K} называется эквивалентным коду \mathcal{K}^* , если существует биекция между кодовыми словами \mathcal{K} и \mathcal{K}^* , сохраняющая расстояние. Показать, что всякий линейный код эквивалентен коду с систематической кодирующей матрицей.
15. Выписать проверочную и порождающую матрицы для $(n, 1)$ -кода с повторением.
16. Построить таблицу синдромов и стандартное расположение для кода \mathcal{K} , заданного проверочной матрицей $H = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$. Найти параметры d и $R = \frac{k}{n}$ для этого кода.
17. Какова может быть наименьшая блоковая длина линейного кода \mathcal{K} с 3 информационными символами, исправляющего все двойные ошибки?
18. Пусть проверочная матрица кода \mathcal{K} имеет вид $H = (A|I)$. Когда \mathcal{K} — самодуальный код?
19. Показать, что в двоичном самодуальном коде каждое кодовое слово имеет четный вес.
20. Для кода с проверочной матрицей $H = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$ найти лидеры смежных классов, содержащих слова $\mathcal{Y}_1 = (1, 1, 1, 0, 1, 0)$ и $\mathcal{Y}_2 = (0, 1, 0, 1, 1, 1, 1)$.
21. Доказать, что число k информационных символов для любого двоичного линейного (n, k) -кода с минимальным расстоянием $d \geq (2t + 1)$ удовлетворяет соотношению $n - k \geq \log_2(1 + C_n^1 + C_n^2 + \dots + C_n^t)$ (граница Хэмминга).
22. Чему равно минимальное расстояние расширенного кода Хэмминга?

6. Конечные поля

И каждый в ИКСПЕДИ-
ЦИИ
Ужасно был бы рад
Узнать, что значит Полюс
И с чем его едят.

А. А. Милн. Винни-
Пух
и все-все-все.

6.1. Аддитивная структура конечных полей

Напомним определение поля.

Определение 18. *Полем называется такое множество F , наделенное двумя операциями (сложением и умножением), для которых выполнены следующие аксиомы:*

- 1) F — абелева группа по сложению;
- 2) $F^* = F \setminus \{0\}$ — абелева группа по умножению;
- 3) сложение и умножение связаны законами дистрибутивности, т. е. для любых $a, b, c \in F$ выполнено

$$(a + b)c = ac + bc.$$

Поле называется конечным, если оно состоит из конечного числа элементов. Нас будут интересовать только конечные поля.

Определение 19. *Число p называется характеристикой поля F , если оно является таким наименьшим положительным натуральным числом, что сумма p единиц равна нулю.*

Существование числа p для конечного поля F следует из аксиомы 1 и аддитивной версии теоремы 5.

Теорема 23. *Характеристика p конечного поля F всегда является простым числом. Для любого $a \in F$ $\underbrace{(a + a + \dots + a)}_{p \text{ раз}} = 0$.*

Теорема 24. *В любом поле F характеристики p*

$$(x + y)^p = x^p + y^p$$

для любых $x, y \in F$.

Множество вычетов по модулю простого числа p — \mathbb{Z}_p — всегда поле характеристики p . Позднее мы увидим, что существуют и другие конечные поля.

Теорема 25. *Определим число $m \in F$ как элемент конечного поля F вида*

$\underbrace{1 + 1 + \dots + 1}_{m \text{ раз}}$. Тогда множество всех чисел поля F характеристики

p образует поле F_p из p элементов (оно называется простым подполем поля F).

Теорема 26. *Число элементов конечного поля характеристики p равно степени числа p .*

Приведем здесь очень простое доказательство этого факта. Действительно, F является конечномерным векторным линейным пространством над F_p . Поэтому F изоморфно m -мерному пространству векторов – строк с элементами из F_p . Отсюда и следует теорема 26.

6.2. Мультипликативная конечных полей

По умножению $F^* = F \setminus \{0\}$ является группой. F^* называется *мультипликативной группой* поля F . Наиболее важный факт, касающийся строения конечного поля, содержит

Теорема 27. *Мультипликативная группа конечного поля циклическая.*

Таким образом, любое конечное поле F содержит такой элемент f , что любой ненулевой элемент поля F есть степень элемента f .

Любой элемент поля F с указанным свойством называется *примитивным* элементом поля F . Фактически теорема 27 утверждает существование примитивных элементов у конечного поля F . Из теоремы 5 следует такое утверждение.

Теорема 28. *Пусть F — конечное поле. $|F| = q$. Тогда $x^q - x = 0$ для любого $x \in F$.*

Теорему 28 часто называют обобщением малой теоремы Ферма.

6.3. Конструирование конечных полей

Пусть $F[x]$ — кольцо всех многочленов одной переменной над полем F . Как обычно, через $\deg f$ будем обозначать *степень* многочлена $f \in F[x]$. Степенью нулевого многочлена считается $-\infty$.

Определение 20. Многочлен $f \in F[x]$ будем называть неприводимым над полем F , если $\deg f > 0$ и из равенства $hg = f$, где $f, g, h \in F[x]$, с необходимостью следует, что либо $\deg h = 0$, либо $\deg g = 0$.

Неприводимые многочлены в кольце всех многочленов играют приблизительно ту же роль, что простые числа в кольце целых чисел.

6.3.1.

Примеры. Многочлен $f = x^2 + x + 1$ неприводим над полем \mathbb{Z}_2 . Многочлен $g = x^2 + 1 = (x + 1)^2$ приводим над \mathbb{Z}_2 .

Как и в случае целых чисел, можно рассматривать деление с остатком в кольце многочленов. Большая часть арифметики многочленов является прямым аналогом арифметики целых чисел.

Теорема 29. Для любых многочленов f и $g \in F[x]$, где $g \neq 0$, существуют такие многочлены h и $r \in F[x]$, что $f = hg + r$, где $\deg r < \deg g$.

Будем говорить, что многочлены f и $h \in F[x]$ сравнимы по модулю многочлена $g \in F[x]$, если $f - h$ делится на g с ненулевым остатком (запись $f \equiv h \pmod{g}$).

Теорема 30 (Свойства сравнений). Если $f \equiv h \pmod{g}$ и $u \equiv v \pmod{g}$, то $f \pm u \equiv v \pm h \pmod{g}$ и $fu \equiv vh \pmod{g}$.

Наибольшим общим делителем многочленов f и g называется такой нормированный (т. е. имеющий коэффициент при старшем члене, равный 1) многочлен d , что d делит f и g и делит любой общий делитель многочленов f и g . Можно доказать, что d является многочленом наивысшей степени с этим свойством (запись: $d = (f, g)$).

Отыскивать наибольший общий делитель можно с помощью алгоритма Евклида. Многочлены f и g взаимно просты, если их наибольший общий делитель равен 1. Если $(f, g) = d$, то из алгоритма Евклида следует, что существуют такие многочлены u, v , что $fu + vg = d$. Как и в случае целых чисел, можно рассматривать многочлены по модулю некоторого фиксированного многочлена.

Пусть $g \in F[x] - \{0\}$. Тогда $\overline{F[x]}$ — множество вычетов по модулю многочлена g будет состоять из всех многочленов f , для которых $\deg f < \deg g$, а операции умножения и сложения производятся над остатками по модулю многочлена g .

6.3.2.

Пример. Рассмотрим множество вычетов по модулю неприводимого многочлена $g = x^2 + x + 1$ над полем \mathbb{Z}_2 . Множество $\overline{\mathbb{Z}_2[x]}$ в этом случае состоит из следующих многочленов: $a_1 = 0, a_2 = x, a_3 = x + 1, a_4 = 1$ (все многочлены степени $< 2 = \deg g$ над \mathbb{Z}_2).

+	a_1	a_2	a_3	a_4	\times	a_1	a_2	a_3	a_4
a_1	a_1	a_2	a_3	a_4	a_1	a_1	a_1	a_1	a_1
a_2	a_2	a_1	a_4	a_3	a_2	a_1	a_3	a_4	a_2
a_3	a_3	a_4	a_1	a_2	a_3	a_1	a_4	a_2	a_3
a_4	a_4	a_3	a_2	a_1	a_4	a_1	a_2	a_3	a_4

Сложение не нуждается в комментариях — это обычное сложение многочленов. Умножение на $0 = a_1$ и $1 = a_4$ происходит также обычным образом. Для того чтобы найти $a_2 a_2 = xx = x^2$, надо сначала вычислить обычное произведение многочленов, а затем найти остаток от деления полученного произведения на многочлен g .

Действительно, $a_2^2 = x^2 = 1(x^2 + x + 1) + (x + 1) = 1 \cdot g + r$ (напомним, что $x + x = 2x = 2 \cdot 1x = 0 \cdot x$). Остаток $r = a_3$. Более удобно поставить в соответствие каждому многочлену из $\overline{F[x]}$ вектор, перечислив в порядке убывания коэффициенты при $x^2, x, x^0 = 1$ и т. д. Тогда $a_1 = (0, 0), a_2 = (1, 0), a_3 = (1, 1), a_4 = (0, 1)$. В этом случае таблица сложения получается просто покомпонентным сложением векторов над \mathbb{Z}_2 . Более удобной оказывается и таблица умножения.

Пример 6.3.2 — первый пример конечного поля, отличного от полей вычетов по модулю простого числа (\mathbb{Z}_4 полем не является). Оказывается, что этот пример не случаен.

Теорема 31. Пусть $g \in F[x]$ — неприводимый многочлен. Тогда $K = \overline{F[x]}$ — множество вычетов по модулю многочлена g является полем. Если $m = \deg g$, и F — конечное поле характеристики p , состоящее из q элементов, то $|K| = q^m$.

Таким образом, для того чтобы построить конечное поле из p^m элементов (p — простое число), достаточно найти неприводимый многочлен

степени m и построить поле вычетов по модулю этого неприводимого многочлена. Оказывается, что любое конечное поле может быть построено таким образом.

Теорема 32. *Для каждого простого числа p и произвольного целого $k > 0$ существует единственное конечное поле порядка p^k . Это поле называется полем Галуа порядка p^k и обозначается через $GF(p^k)$.*

«Единственность» означает единственность с точностью до изоморфизма (т. е. взаимно однозначного соответствия, сохраняющего операции). На самом деле существует много различных реализаций конечного поля $GF(p^k)$, зависящих от выбора неприводимого над \mathbb{Z}_p многочлена g степени k . Для прикладника эти реализации различны (они могут требовать различного технического обеспечения), но с точки зрения алгебраиста — это одно и то же поле. В доказательстве теоремы 32 существенную роль играет такой важный факт.

Теорема 33. *Для любого целого $k > 0$ и простого числа p существует неприводимый многочлен $f \in \mathbb{Z}_p[x]$ степени k .*

6.4. Упражнения

1. Доказать со всеми подробностями теоремы 23–25.
2. Доказать теорему 27 (указание: использовать следующие факты): многочлен $f \in F[x]$ степени > 0 имеет в любом поле $k \supseteq F$ не больше корней, чем его степень; конечная абелева группа является прямым произведением циклических подгрупп (теорема 8).
3. Найти все неприводимые над полем \mathbb{Z}_2 многочлены степени ≤ 4 .
4. Построить таблицы сложения и умножения для поля из 9 элементов.
5. Доказать, что если F — поле из p^m элементов и K — его подполе из p^k элементов, то k делит m .
6. Доказать, что $x^{p^m} - x$ является произведением всех неприводимых над \mathbb{Z}_p нормированных многочленов, степени которых делят n .
7. Найти число примитивных элементов у поля из 16 элементов.

8. Показать, что множество вычетов по модулю многочлена $g = x^4 + x^3 + x + 1 \in F[x]$ не может быть полем ни для какого поля F .
9. Вычислить наибольший общий делитель d чисел 108996 и 76219, представив его в виде $d = 108996s + 76219t$ ($t, s \in \mathbb{Z}$).
10. Вычислить таблицу обратных элементов в \mathbb{Z}_3 .
11. В поле вычетов по модулю неприводимого над \mathbb{Z}_2 многочлена $x^5 + x^2 + 1$ найти все примитивные элементы.
12. Доказать, что если p — простое число, то $(p-1)! + 1$ делится на p .
13. Разложить на неприводимые множители над полем \mathbb{Z}_3 многочлен $x^4 + x^3 + x + 1$.
14. Доказать, что любой нормированный многочлен степени > 0 на произвольном поле F однозначно с точностью до порядка сомножителей разлагается в произведение неприводимых нормированных многочленов.
15. Доказать, что для любого поля F и $f = x^n - 1$, $g = x^m - 1 \in F[x]$ $x^n - 1 \mid x^m - 1$ тогда и только тогда, когда $n \mid m$. Что можно сказать в общем случае о $(x^n - 1, x^m - 1)$?

7. Полиномиальные и циклические коды

— Ой, Тигра, — радостно сказал он, —
мы уже на самой верхушке?
— Нет, — сказал Тигра.

А. А. Милн. Винни-Пух
и все-все-все.

7.1. Полиномиальные коды

Здесь будет описан один специальный класс кодов — полиномиальные.

Пусть дано множество сообщений $U = \{(\alpha_0, \alpha_1, \dots, \alpha_{k-1}) \mid \alpha_i \in \mathbb{Z}_2, i \in \{0, 1, 2, \dots, k-1\}\}$. Каждому сообщению $u = (\alpha_0, \alpha_1, \dots, \alpha_{k-1})$ (формально) можно поставить в соответствие многочлен $u(x) = \alpha_0 + \alpha_1 x + \dots +$

$\alpha_{k-1}x^{k-1}$ (в данном случае удобно нумеровать коэффициенты сообщений с 0).

Определение 21. Пусть $g(x) = g_0 + g_1x + \dots + g_mx^m \in \mathbb{Z}_2[x]$, причем $g_0 \neq 0$ и $g_m \neq 0$. Полиномиальный код с кодирующим многочленом $g(x)$ кодирует слово сообщения $u(x) = \alpha_0 + \alpha_1x + \dots + \alpha_{k-1}x^{k-1}$ многочленом $c(x) = c_0 + c_1x + \dots + c_{n-1}x^{n-1}$, где $n = m + k$ и $c(x) = u(x)g(x)$.

Если $g_0 = 0$, то все кодовые слова будут начинаться с 0, а если $g_m = 0$, то все кодовые слова будут заканчиваться 0, так что условия, наложенные на $g(x)$, достаточно естественны.

7.1.1.

Пример. Пусть информационные слова являются двоичными словами длины 3, $g(x) = x + 1$ — кодирующий многочлен. Построим полиномиальный код, отвечающий этому многочлену. Перечислим сначала все информационные слова и поставленные в соответствие им многочлены:

$$\begin{array}{llll} 000 \longleftrightarrow 0 & 110 \longleftrightarrow x+1 & 001 \longleftrightarrow x^2 & 011 \longleftrightarrow x^2+x \\ 100 \longleftrightarrow 1 & 010 \longleftrightarrow x & 101 \longleftrightarrow x^2+1 & 111 \longleftrightarrow x^2+x+1. \end{array}$$

Теперь перечислим все кодовые слова:

$$\begin{array}{ll} 0000 \longleftrightarrow 0 & 0110 \longleftrightarrow x^3+x^2 \\ 1100 \longleftrightarrow x+1 & 1111 \longleftrightarrow x^3+x^2+x+1 \\ 1010 \longleftrightarrow x^2+1 = (x+1)^2 & 0101 \longleftrightarrow x^3+x \\ 0110 \longleftrightarrow x^2+x & 1001 \longleftrightarrow x^3+1. \end{array}$$

Можно убедиться, что построенный полиномиальный код нам уже знаком, это $(4, 3)$ -код с проверкой на четность.

Вообще, $(n, n-1)$ -код с проверкой на четность является полиномиальным кодом с кодирующим многочленом $g(x) = x + 1$.

На самом деле, полиномиальные коды входят в класс матричных кодов.

Теорема 34. Полиномиальный код с кодирующим многочленом $g(x) = g_0 + g_1x + \dots + g_mx^m$ является матричным кодом с кодирующей матрицей

G размера $k \times n$ ($n = m + k$) вида

$$G = \begin{pmatrix} g_0 & g_1 & \cdots & g_m & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & g_{m-1} & g_m & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & g_0 & \cdots & g_m \end{pmatrix}.$$

Действительно, в j -й строке стоят коэффициенты произведения x^j на $g(x)$. Специальный вид матрицы G облегчает физическую реализацию соответствующего кода.

Из теоремы 34 следует, в частности, что любой полиномиальный код является групповым. Поэтому на основании теоремы 12 справедлива

Теорема 35. *Минимальное расстояние полиномиального кода с кодирующим многочленом $g(x)$ равно минимальному весу ненулевых многочленов вида $g(x)u(x)$.*

Определение 22. Экспонентой многочлена $f(x)$ называют такое наименьшее целое положительное число n , что $f(x)|x^n - 1$. Обозначение $n = e(f(x))$.

Теорема 36. *Если экспонента кодирующего многочлена $g(x)$ полиномиального кода K не меньше его блочной длины n , то минимальное расстояние кода K не меньше трех.*

Доказательство. Очевидно, что если минимальное расстояние между кодовыми словами меньше трех, то существует кодовое слово $C(x)$ вида x^i или $x^i + x^j$, где $0 \leq i < j \leq n$. Из определения кодирующего многочлена ясно, что он взаимно прост с многочленом x^i . Поэтому $C(x) = x^i + x^j + x^i(1 + x^{j-1})$, где $g(x)$ делит $x^{j-1} + 1 = x^m - 1$ ($m = j - i$). По условию теоремы это невозможно, противоречие. \square

Из теоремы 36 следует, что при достаточно общих условиях легко построить полиномиальный код, умеющий распознавать до двух ошибок. Двойной ошибкой будем называть ошибки в двух смежных позициях.

Теорема 37. *Пусть кодирующий многочлен полиномиального кода K имеет вид $g(x) = (1 + x)h(x)$, где $e(h(x)) > n$ (n — блочная длина K). Тогда можно обнаружить любую комбинацию из двух простых или двух двойных ошибок.*

Отметим одно немаловажное преимущество полиномиальных кодов: для хранения информации о полиномиальном (n, k) -коде достаточно всего-навсего одной строки длины $m + 1$, в то время как для матричных кодов требуется хранить матрицу размера $k \times n$ (или $(n - k) \times n$). Как будет видно из дальнейшего, полиномиальные коды достаточно просты в реализации.

7.2. Двоичные циклические коды

В разделе 4 было показано, как строить коды с блоковой длиной $n = 2^r - 1$ (r — число проверочных позиций) и $n - r = 2^r - r - 1$ информационными символами. У такого кода (кода Хэмминга) все $2^r - 1$. Но столько же ненулевых элементов в поле $GF(2^r)$. Оказывается, что удобно представлять столбцы этой матрицы в виде последовательных степеней примитивного элемента α из поля $GF(2^r)$. Пусть $g(x)$ — неприводимый многочлен над \mathbb{Z}_2 степени r , корнем которого является α (существование $g(x)$ нетрудно установить [3]).

7.2.1.

Пример. Пусть α — корень неприводимого над \mathbb{Z}_2 многочлена $x^3 + x + 1$. В поле $GF(8)$ любой элемент, не лежащий в \mathbb{Z}_2 , является примитивным. Тогда многочлены степени ≤ 2 от α находятся во взаимно однозначном соответствии с элементами поля $GF(8)$. Так,

$$\begin{array}{lll} 0 & = & 0 \cdot \alpha & \longleftrightarrow & (0, 0, 0) \\ 1 & = & \alpha^0 & \longleftrightarrow & (1, 0, 0) \\ \alpha & = & \alpha^1 & \longleftrightarrow & (0, 1, 0) \\ \alpha^2 & = & \alpha^2 & \longleftrightarrow & (0, 0, 1) \\ \alpha^3 & = & \alpha + 1 & \longleftrightarrow & (1, 1, 0) \\ \alpha^4 & = & \alpha^2 + \alpha & \longleftrightarrow & (0, 1, 1) \\ \alpha^5 & = & \alpha^2 + \alpha + 1 & \longleftrightarrow & (1, 1, 1) \\ \alpha^6 & = & \alpha^2 + 1 & \longleftrightarrow & (1, 0, 1). \end{array}$$

После того, как мы запишем проверочную матрицу кода Хэмминга по степеням α , получим новую проверочную матрицу

$$H_1 = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \end{pmatrix}$$

Пусть R — принятое декодером слово.

Синдром ошибки $H_1 R^t = S$ получается как сумма степеней α :

$$S(\alpha) = H_1 R^t = \sum_{i=0}^{n-1} R_i \alpha^i, \quad R_i \in \mathbb{Z}_2 \quad (i = 0, 1, \dots, n-1).$$

Если синдром равен нулю, то это означает, что неприводимый многочлен $g(x)$ делит многочлен $R(x)$, ибо α — корень неприводимого многочлена $g(x)$.

В общем случае $S(\alpha) = R(\alpha)$ есть значение многочлена $R(x)$ в точке α , а многочлен $C(x)$ является кодовым тогда и только тогда, когда $g(x)$ делит $C(x)$. Итак, справедлива

Теорема 38. *Код Хэмминга (при соответствующем переупорядочивании его проверочной матрицы) является полиномиальным кодом с кодирующим полиномом $g(x)$, делящим $x^n - 1$ ($n = 2^r - 1$).*

Пусть $h(x) = (x^n - 1)/g(x)$, где $g(x)$ — кодирующий многочлен для полиномиального кода Хэмминга. Допустим, что $C(x)$ — любой кодовый многочлен. Тогда $f(x)C(x) = (x^n - 1)g(x) + r(x)$, где $r, f, g \in \mathbb{Z}_2[x]$, а $r(x)$ — остаток от деления $f(x)C(x)$ на $x^n - 1$. Понятно, что $\deg r(x) < n$ и что $g(x)$ делит как $f(x)C(x)$, так и $x^n - 1$. Отсюда $r(x)$ также делится на $g(x)$, т. е. является кодовым многочленом.

Если $C(x) = C_0 + C_1x + \dots + C_{n-1}x^{n-1}$, то $xC(x) = C_0x + C_1x^2 + \dots + C_{n-1}x^n = C_{n-1} + C_0x + \dots + C_{n-2}x^{n-1} + C_{n-1}(x^n - 1)$. Поэтому циклический сдвиг любого кодового слова снова является кодовым словом.

Определение 23. *Код \mathcal{K} называется циклическим, если циклический сдвиг любого кодового слова также является кодовым словом. Из сказанного понятна справедливость следующей теоремы.*

Теорема 39. *Код Хэмминга является циклическим кодом.*

Связь циклических и полиномиальных кодов видна из следующей теоремы.

Теорема 40. *Любой циклический код является полиномиальным.*

Циклические коды являются одним из основных достижений теории кодирования и получают широкое распространение на практике.

7.3. Техническая реализация циклических и полиномиальных кодов

Арифметику полей Галуа можно реализовать с помощью логических цепей. Для этого требуются элементы цепи для запоминания элементов поля, называемые *разрядами регистра сдвига*. Регистр сдвига представляет собой последовательность элементов памяти, называемых разрядами (в двоичном случае — триггерами). Каждый разряд содержит ровно один элемент поля $GF(q)$ (для поля \mathbb{Z}_2 — 0 и 1). Содержащийся в каждом разряде символ, покидая этот разряд, появляется на выходящей из него линии. Каждый разряд снабжен входом, по которому в него поступает элемент поля. В дискретные моменты времени, называемые *тактами*, элементы, содержащиеся в устройствах памяти, замещаются элементами поля, находящимися на входах.

Кроме этого, необходимо иметь устройство, осуществляющее сложение в $GF(q)$ (сумматор), умножение на элемент поля $GF(q)$ (умножитель), а также элементы, реализующие логические операции «и», «или» и «не». Устройства, составленные из указанных элементов, позволяют перемножать, складывать и делить многочлены, а потому и выполнять действия в поле Галуа.

Триггер на самом деле не является простейшим элементом и может быть собран из элементов «и», «или», «не», но в логических схемах его удобно считать самостоятельным элементом.

7.3.1.

Умножение на фиксированный многочлен

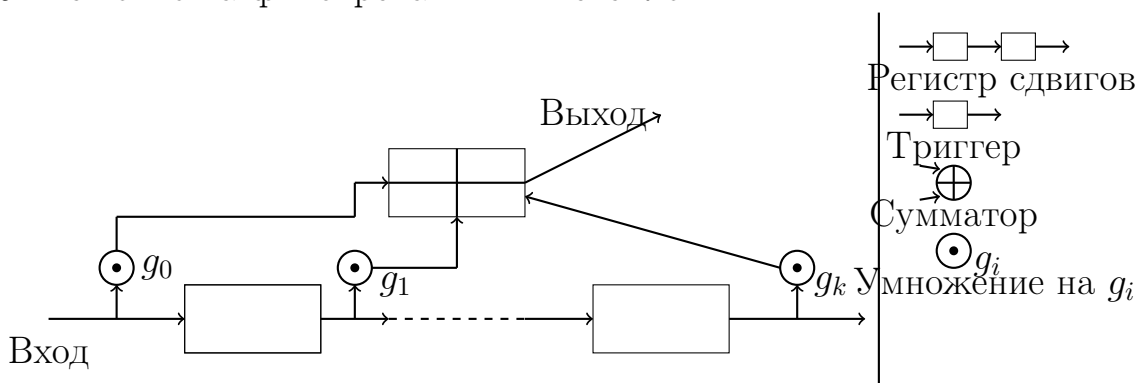
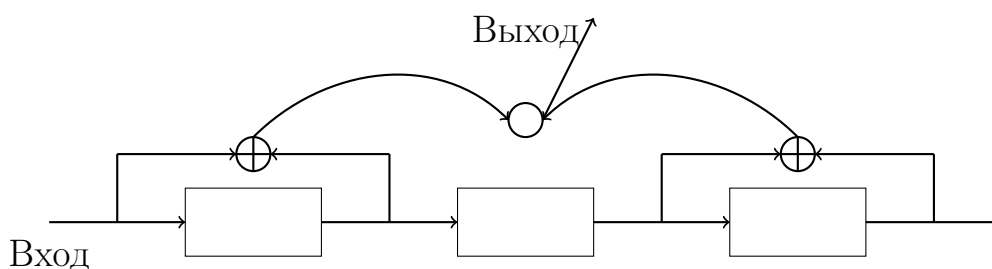


Рис. 6.1. Логическая схема, реализующая умножение на фиксированный многочлен $g(x)$

На вход подаются коэффициенты многочлена $f(x) = \sum_{i=0}^m f_i x^i$ в порядке возрастания степени. На выходе появляются коэффициенты произведения в порядке возрастания. В случае двоичных кодов умножение на g_i осуществляется очень просто: если $g_i = 1$, то проводник соединяет соответствующий элемент с сумматором на выходе. В противном случае проводник отсутствует. Триггеры задерживают вход на один такт.

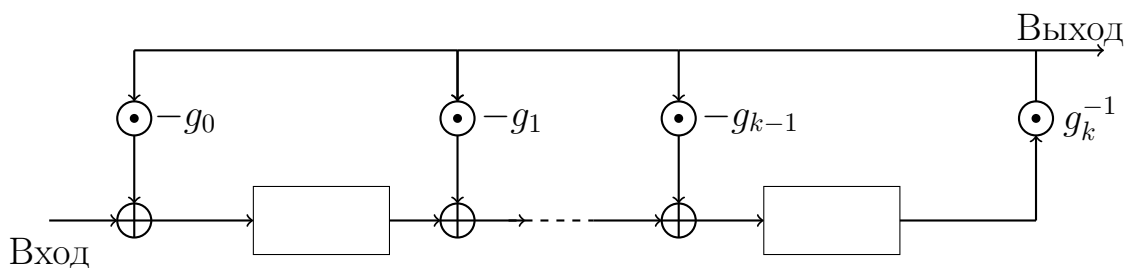
7.3.2.

Пример. Изобразим логическую схему, умножающую произвольный многочлен на многочлен $g(x) = 1 + x + x^2 + x^3$:



7.3.3.

Схема деления с остатком на фиксированный многочлен $g(x) = g_0 + g_1x + \dots + g_kx^k$ такова:



Коэффициенты делимого подаются в порядке убывания степени, на выходе появляются коэффициенты частного, а в триггерах оказываются записанными коэффициенты остатка.

7.3.4.

Пример. Схема деления на многочлен

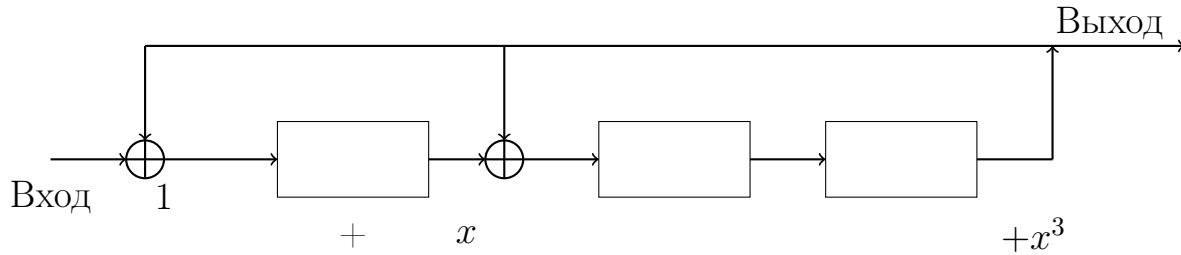


Рис. 6.4.

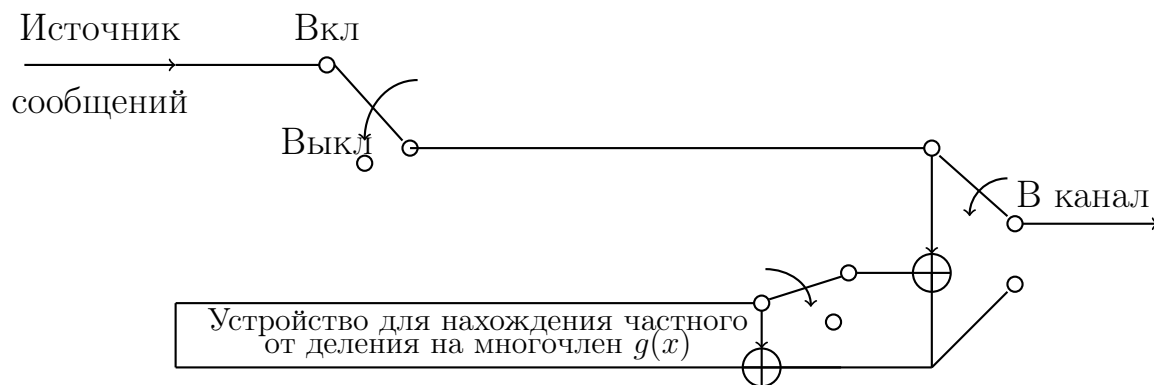
Правильность работы схемы можно проследить по тактам [3].

7.4. Кодер для циклического кода

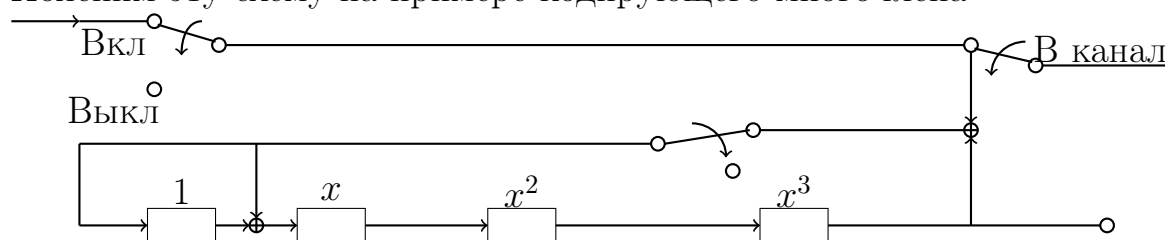
В соответствии с теоремой 41 можно рассматривать любой циклический код как множество всех многочленов, кратных данному и имеющих блоковую длину n . Однако при перемножении информационного многочлена $u(x) = \sum_{i=0}^{k-1} u_i x^i$ на кодирующий многочлен $g(x) = \sum_{i=0}^m g_i x^i$ происходит перемешивание символов. В результате, даже если мы будем уверены в правильности полученного сообщения, не так-то легко его будет извлечь из кодового слова. В связи с этим полезным является следующее соображение. Вместо слова $u(x)$ рассматривается слово $u(x)x^m = f(x)$. Слово $f(x)$ делится с остатком на многочлен $g(x)$: $f(x) = g(x)q(x) + r(x)$, где $\deg r(x) < m = \deg g(x)$. Очевидно, что тогда слово $f(x) - r(x)$ будет делиться на $g(x)$. Его и берут в качестве кодового слова.

Физически это реализуется следующим образом. Сначала блок, соответствующий информационному слову $u(x)$, дополняется $m = n - k$ нулями. Затем находится остаток от деления полученного многочлена на $g(x)$. Наконец, пользуясь тем, что в двоичной арифметике сложение и вычитание одно и то же, на место последних m нулей в слове $f(x)$ записываются коэффициенты остатка. Тогда $C(x) = f(x) + r(x)$ и есть кодовое слово, первые k символов которого информационные, а последние $n - k$ символов проверочные.

Общая схема кодера для такого кода выглядит так:



Поясним эту схему на примере кодирующего многочлена



При кодировании очередного блока на вход устройства деления и в канал подаются символы информационного слова. При этом все ключи занимают верхнее положение. После того как информационные позиции переданы в канал, все ключи занимают нижнее положение. С регистра сдвигов, на котором получается остаток, считываются коэффициенты остатка. После этого устройство подготавливается к приему нового блока.

В общем, почти все кодеры для различных циклических кодов выглядят аналогичным образом. В строении декодеров имеется существенно большее разнообразие.

7.5. Декодер для кода Хэмминга

В соответствии со схемой, предложенной в 7.2, столбы проверочной матрицы (n, k) -кода Хэмминга $(n = 2^r - 1)$ —упорядочены по степеням α , примитивного элемента поля $GF(2^r)$.

В этом случае для принятого слова $R(x) = R_0 + R_1x + \dots + R_{n-1}x^{n-1}$ синдром ошибок есть значение $R(x)$ в точке α : $S = R(\alpha)$. Так как каждой позиции отвечает степень α , и, если ошибка произошла в позиции, отвечающей столбцу α^i (говорят еще "в позиции с локатором α^i "), то $S = \alpha^i$. Переписав это равенство в виде $S\alpha^{-i} = 1$, убедимся, что $S\alpha^{n-1} = 1$ ($\alpha^n = 1$).

Сначала найдем синдром ошибок S (он вычисляется как остаток

от деления $R(x)$ на $g(x)$, где $g(x)$ — неприводимый многочлен над \mathbb{Z}_2 с корнем α), а затем будем поочередно сравнивать S с каждым из локаторов ошибок. Проще всего это сделать полным просмотром всех локаторов. При этом надо учесть, что сначала рассматривается позиция с локатором α^{n-1} , затем α^{n-2} и т. д. до $\alpha^0 = 1$. Но для $i = n - 1$ получим $S\alpha = 1$, для $i = n - 2$ — $S\alpha^2 = 1$ и т. д. Таким образом, необходимо каждый раз умножать полученный результат на α и сравнивать с единицей. Как только получится равенство, соответствующую позицию надо исправить.

7.5.1.

Приведем пример реализации этой методики для кода Хэмминга, основанного на неприводимом многочлене $g(x) = x^5 + x^2 + 1$.

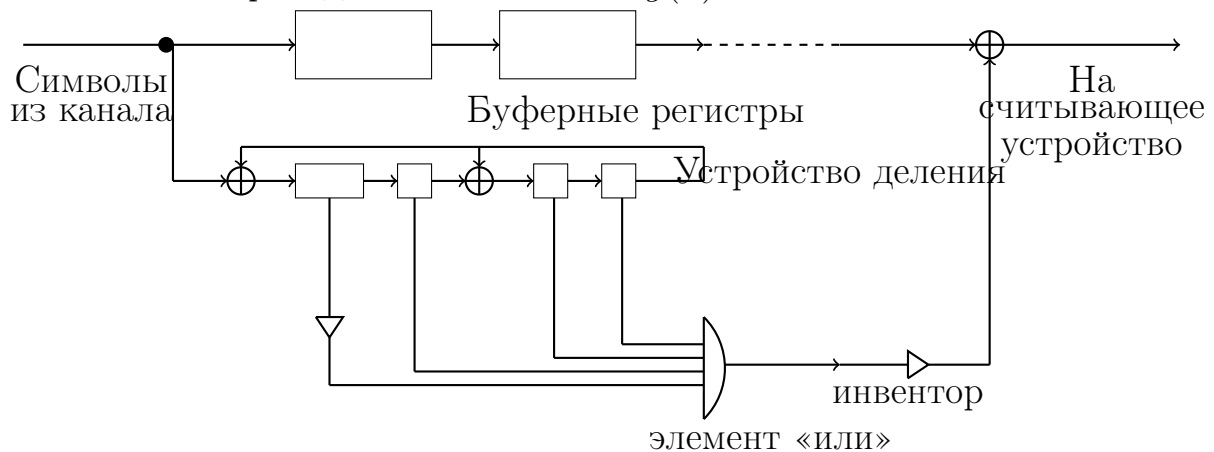


Рис. 6.7.

Предварительно в буферном регистре устанавливаются нули. Затем символы из канала посылаются одновременно в буфер и на устройство деления. После приема всего блока в буфер, в устройстве деления накопится остаток, равный синдрому S . После этого из буфера будут выводиться символы принятого слова, сложенные с нулем, если $S\alpha^i \neq 1$, и с единицей, если $S\alpha^i = 1$ для соответствующей позиции.

Здесь используется то обстоятельство, что то же устройство деления, которое находило синдром, может применяться и для вычисления произведения $S\alpha$ в поле $GF(2^r)$.

7.6. Упражнения

1. Доказать, что код, дуальный к циклическому, будет циклическим.
2. Какова экспонента неприводимого над $GF(2)$ многочлена, отличного от x ?
3. Построить регистр сдвигов, реализующий обращение элемента в поле вычетов по модулю неприводимого над \mathbb{Z}_2 многочлена $x^5 + x^2 + 1$.
4. Построить кодеры для двоичных циклических кодов длины 15 с кодирующими многочленами: а) $x^4 + x + 1$; б) $x^8 + x^7 + x^6 + x^4 + 1$.
5. Как выглядит кодирующая матрица для циклического кода Хэмминга?
6. Для решения некоторых задач теории кодирования необходимо уметь определять минимальный многочлен для степени примитивного элемента поля. Придумать алгоритм определения минимального многочлена для α^3 , если минимальный для α известен.

8. Двоичные БЧХ-коды

И вскоре Все–Все–Все пришли
(Не просто, а на помощь к нам),
И выход тут же мы нашли
(Вернее, он нашелся сам).

А. А. Милн. Винни-Пух
и все-все-все.

8.1. БЧХ-коды, исправляющие 2 ошибки

БЧХ-коды, исправляющие 2 ошибки, открыты в 1959–1960 годах независимо Р. Боузом, и Рой Чоудхури, и А. Хоквингемом. При этом сразу же было получено обобщение на случай числа ошибок.

Общая идея состояла в следующем. Если для исправления 1 ошибки коду с блоковой длиной $n = 2^r - 1$ требуется r проверок, то код с $2r$

проверками, по-видимому, сможет исправлять 2 ошибки. При этом использовались соображения предыдущего раздела о соответствии между многочленами и столбцами проверочной матрицы.

Мы построим сейчас проверочную матрицу БЧХ-кода, исправляющего 2 ошибки. Будем считать, что H состоит из двух подматриц H_1 и H_3 , так что

$$H = \begin{pmatrix} H_1 \\ H_3 \end{pmatrix},$$

где H_1 — проверочная матрица циклического кода Хэмминга (т. е. столбцы H_1 упорядочены по степеням α примитивного элемента поля $GF(2^r)$). Таким образом, $H_1 = (1|\alpha|\alpha^2|\dots|\alpha^{n-1})$. Столбцы матрицы H_3 упорядочиваются по степеням элемента α^3 , т. е.

$$H_3 = (1|\alpha^3|\alpha^6|\dots|\alpha^{3n-3}) \quad (n = 2^r - 1).$$

Для принятого вектора R также удобно разбить синдром ошибок на две части в соответствии с разбиением H , обозначив $S_1 = H_1 R^t, S_3 = H_3 R^t$.

Будем интерпретировать столбцы матриц H_1 и H_3 , а также S_1 и S_3 как элементы поля $GF(2^r)$. Допустим, что ошибки произошли в позициях, отвечающих столбцам $\beta_1, \beta_2 \in GF(2^r)$ матрицы H_1 . Учитывая, что β_1, β_2 — элементы $GF(2^r)$, $S_1 = \xi_1, S_3 = \xi_3 \in GF(2^r)$, получаем следующую систему уравнений с неизвестными β_1, β_2 :

$$\begin{cases} \beta_1 + \beta_2 = \xi_1 \\ \beta_1^3 + \beta_2^3 = \xi_3 \end{cases}.$$

Учитывая, что $\beta_1^3 + \beta_2^3 = (\beta_1 + \beta_2)(\beta_1^2 + \beta_2^2 + \beta_1\beta_2) = (\beta_1 + \beta_2)((\beta_1 + \beta_2)^2 + \beta_1\beta_2) = (\beta_1 + \beta_2)^3 + (\beta_1 + \beta_2)\beta_1\beta_2$, получим

$$\begin{cases} \beta_1 + \beta_2 = \xi_1 \\ \beta_1^3 + \xi_1\beta_1\beta_2 = \xi_3 \end{cases}.$$

При $\xi_1 \neq 0$ получим

$$\begin{cases} \beta_1 + \beta_2 = \xi_1 \\ \beta_1\beta_2 = \xi_3/\xi_1 + \xi_1^2 \end{cases}.$$

Воспользовавшись теоремой Виета, получим квадратное уравнение, которому удовлетворяют элементы β_1 и β_2 :

$$\beta^2 + \xi_1\beta + (\xi_1^2 + \xi_3/\xi_1) = 0.$$

Удобнее, однако, рассматривать взаимное к нему уравнение

$$\sigma(z) = 1 + \sigma_1 z + \sigma_2 z^2 = 1 + \xi_1 z + (\xi_1^2 + \xi_3/\xi_1) z^2 = 0. \quad (*)$$

Если произошла 1 ошибка, то $\xi_1 = \beta_1$, $\xi_3 = \beta_1^3$ и коэффициент при z^2 равен нулю. Если ошибок не произошло, то $S_1 = S_2 = 0$, а тогда $\sigma(z) = 1$ и $\xi_1 = \xi_3 = 0$. Многочлен $\sigma(z)$ часто называют многочленом *локаторов ошибок*.

Если имеется не более 2 ошибок, то декодер по уравнению $\sigma(z) = 0$ сможет найти локаторы этих ошибок. Если же исказится большее число символов, то либо произойдет ошибка, либо отказ от декодирования.

Обратимся к получению кодовых слов для БЧХ-кодов, исправляющих 2 ошибки. Понятно, что слово $C(x) = (C_0, C_1, C_2, \dots, C_{n-1})$ будет кодовым тогда и только тогда, когда для него $S_1 = S_3 = 0$. Так как столбцы матриц H_1 и H_3 закодированы с помощью степеней элементов α и α^3 соответственно, то $C(x)$ — кодовое слово, тогда и только тогда $C(\alpha) = C(\alpha^3) = 0$. Поэтому $C(x)$ делится на минимальные многочлены как для элемента α , так и для α^3 . Пусть $g_1(x)$ и $g_3(x)$ — эти многочлены. Тогда $C(x)$ должен делиться на $\varphi(x) = \text{НОК}(g_1(x), g_3(x))$. Нетрудно убедиться, что полученный код, исправляющий 2 ошибки, будет циклическим с кодирующим многочленом $\varphi(x)$. Кодер для этого кода мало чем отличается от кодеров для других циклических кодов.

Чуть сложнее обстоит дело с декодером. Для того чтобы построить декодер БЧХ-кода, исправляющего 2 ошибки, необходимо уметь составлять уравнение (*), приведенное выше. В частности, нужно иметь некоторые средства, позволяющие находить коэффициенты этого уравнения, затем в соответствии с общей идеей перебираются все элементы поля $GF(2^r)$ и предьявляются в качестве возможных корней этого уравнения. Если выясняется, что β^{-1} — корень уравнения (*), то соответствующая позиция исправляется.

Отметим, что БЧХ-коды, исправляющие 2 ошибки, также являются высокоскоростными кодами: $R = k/n \rightarrow 1$ при $n \rightarrow \infty$.

8.2. Двоичные БЧХ-коды, исправляющие t ошибок

Идея, лежащая в основе БЧХ-кода, исправляющего 2 ошибки, оказалась универсальной и используется для построения циклических кодов, исправляющих произвольное число t ошибок.

Кодирующий многочлен строится следующим образом. Выберем r так, чтобы $2^r \geq 2^t + 1 = d$. Возьмем α примитивным элементом поля $GF(2^r)$. Через $m_1(x)$ обозначим минимальный многочлен элемента α , $m_2(x)$ — минимальный многочлен элемента α^2 , и т. д. $m_i(x)$ — минимальный многочлен элемента α^i . Пусть $g(x) = \text{НОК}(m_1(x), m_2(x), \dots, m_{d-1}(x))$. Тогда справедлива следующая

Теорема 41. Пусть \mathcal{K} — циклический код с блоковой длиной $n = 2^r - 1$ и кодирующим многочленом $g(x)$. Тогда минимальное расстояние кода \mathcal{K} не меньше чем d .

Доказательство. Прежде всего $g(x)$ — многочлен над $GF(2)$ наименьшей степени с корнями $\alpha, \alpha^2, \dots, \alpha^{d-1}$. Поэтому если $c(x)$ — кодовый многочлен, то из $c(\alpha) = 0 = c(\alpha^2) = \dots = c(\alpha^{d-1})$ следует, что $c(x)$ делится на $g(x)$. Покажем, что $c(x) \neq 0$ имеет не менее d ненулевых коэффициентов. Допустим противное, т. е. существует такой кодовый многочлен

$$c(x) = c_1 x^{n_1} + c_2 x^{n_2} + \dots + c_{d-1} x^{n_{d-1}},$$

что $0 \leq n_1 < n_2 < \dots < n_{d-1} \leq n - 1$ и хотя бы один из коэффициентов отличен от нуля. Тогда коэффициенты должны удовлетворять следующей системе уравнений

$$\sum_{i=1}^{d-1} c_i d^{kn_i} = 0, \quad k = 1, 2, \dots, d-1,$$

Ее можно рассматривать как однородную систему уравнений относительно c_1, c_2, \dots, c_{d-1} с определителем

$$\begin{vmatrix} \alpha^{n_1} & \alpha^{n_2} & \dots & \alpha^{n_{d-1}} \\ \alpha^{2n_1} & \alpha^{2n_2} & \dots & \alpha^{2n_{d-1}} \\ \dots & \dots & \dots & \dots \\ \alpha^{(d-1)n_1} & \alpha^{(d-1)n_2} & \dots & \alpha^{(d-1)n_{d-1}} \end{vmatrix} = \prod_{i>j} (\alpha^{n_j} - \alpha^{n_i}),$$

где $n_i < n_j \leq 2^r - 1 = n$. Так как $\alpha^{n_i} \neq \alpha^{n_j}$ в силу выбора α ($\alpha^n = 1$ и $\alpha^m \neq 1$ при $0 < m < n$), то эта система имеет только нулевое решение, противоречие. \square

Теорема 42. Существует двоичный БЧХ-код с блоковой длиной $n = 2^r - 1$, минимальным расстоянием $d < 2n/r$ и числом проверок не более чем $r(d-1)/2$.

Доказательство. Прежде всего убедимся, что минимальный многочлен элемента $\beta \neq 0$ совпадает с минимальным многочленом элемента β_s^2 (для любого $\beta \in GF(2^r)$). Действительно, пусть $f(x) = \sum_{i=0}^s f_i x^i$ — минимальный многочлен над $GF(2)$ для элемента β . Тогда $f(\beta) = \sum_{i=0}^s f_i \beta^i = 0$. Так как поле $GF(2^r)$ имеет характеристику 2, то по теореме 24 имеем

$$0 = f(\beta)^2 = \sum_{i=0}^s f_i^2 \beta^{2i}.$$

Так как $f_i \in GF(2)$, то $f_i^2 = f_i$. Поэтому $f(\beta)^2 = f(\beta^2)$. Таким образом, β^2 является корнем многочлена $f(x)$. С другой стороны, минимальный многочлен для любого элемента поля $GF(2^r)$, очевидно, неприводим над $GF(2)$. Отсюда минимальные многочлены для β и β^2 совпадают.

Таким образом, достаточно рассматривать наименьшее общее кратное многочленов, являющихся минимальными многочленами для нечетных степеней примитивного элемента α .

Покажем теперь, что степень минимального многочлена над $GF(2)$ для любого элемента β из $GF(2^r)$ не превосходит r . Действительно, $GF(2^r)$ является r -мерным векторным пространством над $GF(2)$. Поэтому любые $r + 1$ элементов поля $GF(2^r)$ линейно зависимы. В частности, линейно зависимыми являются элементы $1, \beta, \beta^2, \dots, \beta^{r-1}, \beta^r$. Но тогда существуют такие коэффициенты $\lambda_0, \lambda_1, \dots, \lambda_{r-1}, \lambda_r$, что $\sum_{i=0}^r \lambda_i \beta^i = 0$. Отсюда следует, что найдется многочлен $\varphi(x)$ степени $\leq r$ с коэффициентами из $GF(2)$, корнем которого является β . Степень же минимального многочлена для β , конечно, не превосходит $\deg \varphi(x) \leq r$.

Итак, степень $g(x) = \text{НОК}(m_1(x), \dots, m_{d-1}(x))$ не превосходит $r(d - 1)/2$. Это доказывает теорему 42. \square

БЧХ-коды широко распространены и работают во многих системах связи. Их надежность достаточно велика. В то же время то расстояние, которое гарантируется теоремами 41 и 42 (говорят, что это — конструктивное расстояние БЧХ-кода), достаточно далеко от настоящего их расстояния, которое в большинстве случаев неизвестно. Кроме того, при заданной корректирующей способности d/n и $n \rightarrow \infty$ оказывается, что БЧХ-коды имеют большую избыточность. Поэтому не прекращаются попытки построения кодов с меньшей избыточностью. Строящиеся коды, как правило, конструируются при помощи БЧХ-кодов, но лишены некоторых недостатков последних.

8.3. Общая схема декодера для БЧХ-кода

Если БЧХ-коды используются лишь для обнаружения ошибок, то соответствующий декодер лишь вычисляет синдром ошибок, что реализуется очень просто уже известными средствами. Если же требуется построить декодер, который исправляет ошибки, то появляются некоторые дополнительные затруднения.

Прежде всего, для решения этой задачи необходимо построить (по образцу п. 8.1) ключевое уравнение, корнями которого будут элементы поля $GF(2^r)$, взаимные к локаторам ошибок. Затем необходимо сконструировать соответствующую схему, которая проверяет, является ли элемент поля корнем ключевого уравнения. Наконец, необходимо так синхронизировать работу устройств, чтобы при выходе из декодера ошибочные позиции сразу же исправлялись. Общая схема декодера для произвольного двоичного циклического кода заключается в следующем (см. рис. 17). Декодер состоит из четырех основных частей: буферного регистра, содержащего $2n$ ячеек, регистров сдвига, осуществляющих деление каждого из принимаемых многочленов на неприводимые множители кодирующего многочлена, центрального устройства обработки в поле Галуа и схемы, реализующей проверку. Обращается ли в нуль многочлен локаторов ошибок (называемый процедурой Ченя)?

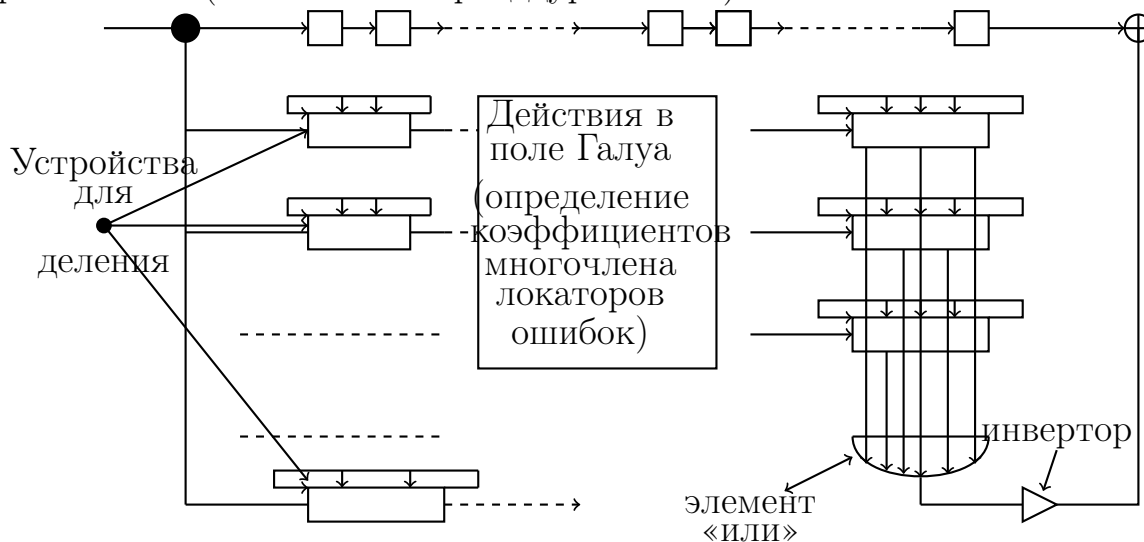


Рис. 7.1.

В случае кодов Хэмминга центральное устройство может быть вообще исключено, а объем буфера уменьшен до n триггеров.

В случае БЧХ-кодов, исправляющих 2 ошибки, центральное устройство обработки должно по заданным синдромам S_1 и S_2 вычислить коэффициенты $\sigma_1 = (S_1^3 + S_3)/S_1$ и $\sigma_2 = S_2$ для многочлена локатора ошибок.

8.4. Упражнения

1. Большая часть изложенной здесь теории переносится на случай, когда алфавитом является произвольное поле. Дать доказательство теорем 41 и 42 с соответствующими модификациями.
2. Построить кодер и декодер для двоичного БЧХ-кода с блоковой длиной 15, исправляющего 2 ошибки.
3. Многочлен $g(x) = x^8 + x^7 + x^6 + x^4 + 1$ порождает над $GF(2)$ циклический код длины 15. Найти проверочную матрицу этого кода. Сколько ошибок может исправлять этот код?
4. Найти минимальный многочлен для каждого элемента поля $GF(16)$.

9. Заключительные замечания

«Собака кусается»...
 Что ж, не беда.
 Загадочно то, что собака,
 Хотя и кусается, но никогда
 Себя не кусает однако...

А. А. Милн. Винни-Пух
 и все-все-все.

9.1. БЧХ-коды над полями нечетной характеристики и коды Рида-Соломона

Безусловно, построенная выше теория линейных, циклических и БЧХ-кодов почти дословно переносится и на случай кодов над конечными полями нечетной характеристики с метрикой Хэмминга в качестве основной руководящей идеи. Интерпретация кодов БЧХ в качестве матричных кодов и использование строения конечных полей дают ключ к означенному, более общему, взгляду на вещи.

Пусть $F = GF(q)$, $q = p^m$ — поле характеристики $p > 2$. Для построения БЧХ-кода, исправляющего t ошибок, необходимо, чтобы q было не меньше $2t + 2$. На самом деле, чтобы $q > 2t + k + 2$, где k — число информационных символов (коды строятся для защиты информации!). Пусть α — примитивный элемент поля F , а g_i — неприводимый над $GF(p)$ многочлен, у которого корнем является элемент α^i . Тогда $\text{НОК}(g_1, g_2, \dots, g_{2t}) = g(x)$ — кодирующий многочлен кода \mathcal{K} с минимальным расстоянием $d \geq 2t + 1$, т. е. кода, способного исправить любые $\leq t$ ошибок.

Легко видеть, что степень каждого из многочленов g_i не превосходит m . Поэтому можно рассмотреть матрицу H , составленную из блоков H_i , отвечающих многочлену g_i , так что в результате имеем $2t$ синдромов $S_i = H_i R^t$, где R — это принятое сообщение. Фактически $S_i = R(\alpha^i)$.

$$H = \begin{pmatrix} \frac{H_1}{H_2} \\ \dots \\ \frac{H_{2t}}{H_{2t}} \end{pmatrix}.$$

Эта информация затем используется для обнаружения и исправления ошибок. В отличие от случая поля $GF(2)$ требуется не только диагностировать места появления ошибок, но и их величины. Как и ранее, мы считаем, что принятое сообщение имеет вид: $R = C + E$, где $E = (\gamma_0, \gamma_1, \dots, \gamma_{n-1})$ — вектор ошибок, где компоненты γ_i принимают значения в поле $GF(p)$. После определения мест появления (локаторов) ошибок величины γ_i коэффициенты вектора ошибок можно определить, решая уравнения $R(\alpha^i) = S_i$. Детали можно найти в [4]. Пусть \mathcal{K} — циклический (n, k) -код с кодирующим многочленом $g(x)$. По определению, кодовое слово $C(x)$ этого кода является произведением $U(x)g(x) \pmod{x^n - 1}$, тогда как $g(x)$ делит $x^n - 1$. Пусть, $h(x) = \frac{x^n - 1}{g(x)}$. Тогда $C(x)h(x) \equiv 0 \pmod{x^n - 1}$. Таким образом, многочлен $h(x)$ будет проверочным многочленом для кода \mathcal{K} .

Важным и часто используемым является частный случай БЧХ-кодов, код Рида - Соломона. Это такие БЧХ-коды, у которых мультипликативный порядок алфавита символов кодового слова делится на длину кода. Например, если поле символов $GF(p)$ совпадает с полем локаторов ошибок $GF(p)$, т. е. $n = p - 1$. Коды Рида-Соломона оптимальны в смысле границы Синглтона. Действительно, в этом случае минимальный многочлен для любого элемента поля является линейным. Поэтому

число проверок равно $m = n - k = 2t$. Отсюда минимальное расстояние $d = n - k + 1$ для таких кодов достигается.

Хороший и нетривиальный пример для такого кода — поле $GF(2^m)$. К сожалению, алгоритмы декодирования для БЧХ-кодов большой длины и числа t исправляемых ошибок нетривиальны.

9.2. Латинские квадраты и коды

Латинские квадраты получили широкую известность благодаря задаче Л. Эйлера о 36 офицерах, построенных в каре (1782 год). Имеется по 6 офицеров из 6 различных полков 6 различных званий. Можно ли их построить в каре так, чтобы в каждой колонне и в каждой шеренге встретились офицеры всех званий и всех полков?

Лишь в 1910 году была доказана неразрешимость этой задачи. Но латинские квадраты не потеряли интереса для прикладников, так как имеют разнообразные практические применения; в конце 60-х годов нашего века латинские квадраты были использованы для построения кодов с простым (мажоритарным) декодированием и большим кодовым расстоянием [5].

9.3. Коды Рида – Маллера

Коды Рида–Маллера были применены в 1972 году при передаче фотографий Марса космическим кораблем "Маринер". Пусть $r, m \in \mathbb{N}$, где $r < m$. Код Рида–Маллера r -го порядка длины $n = 2^m$ обозначается как $RM(r, m)$. Он определяется как линейный код с порождающей матрицей

$$G = \begin{pmatrix} G_0 \\ G_1 \\ \dots \\ G_r \end{pmatrix},$$

где G_0 — строка длины 2^m , состоящая из одних единиц. G_1 — матрица кода Хэмминга, к которой добавлен столбец из нулей высоты m . Таким образом, код $RM(1, m)$ есть дуальный код к расширенному коду Хэмминга (см. §4.5).

При $2 \leq s \leq r$ матрица G_s состоит из всевозможных покомпонентных произведений s строк матрицы G_1 , так что число строк матрицы

G_s равно C_m^s . Минимальное расстояние кода $RM(r, m)$ не меньше 2^{m-r} . Например, космические корабли "Маринер" снабжались для связи кодом $RM(1, 5)$ длины 32 с минимальным расстоянием 16, т. е. могли обнаруживать 15 ошибок и исправлять 7.

В настоящее время коды Рида–Маллера не считаются самыми важными среди линейных кодов несмотря на исторические заслуги. Однако, кажется, получают новую судьбу в связи с использованием квантовой криптографии.

9.4. Матрицы Адамара

Матрицы Адамара были введены в математику в связи с решением задачи о максимизации определителя. Матрица H размера $n \times n$ называется матрицей Адамара, если ее элементами являются числа ± 1 и $HH^t = nI$ (I — единичная матрица).

Справедлива следующая

Теорема 43. Пусть $n > 2$. Необходимым условием для существования матрицы Адамара является условие $n \equiv 0 \pmod{4}$.

С матрицами Адамара связано много прикладных комбинаторных задач. В частности, неизвестно, для любого ли кратного 4 существует матрица Адамара (наименьшее число n , для которого это не удалось проверить, — 668).

В 1960 году обнаружилось, что при помощи матрицы Адамара порядка n можно построить код, состоящий из $2n$ слов с минимальным расстоянием $n/2$.

Очень легко построить матрицу Адамара размера $2^m \times 2^m$, поскольку кронекеровское произведение двух матриц Адамара тоже является матрицей Адамара. Напомним, что если $A = (a_{ij})$ и $B = (b_{ij})$ — две матрицы размеров $m \times m$ и $n \times n$, то их кронекеровское произведение есть матрица $A \otimes B = (a_{ij}B)$ — размера $mn \times mn$.

9.4.1.

Пример. Матрица Адамара размера 2×2 :

$$H_1 = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}.$$

Матрица размера 4×4 :

$$H_2 = \begin{pmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Очевидно, $H_2 = H_1 \otimes H_1$. Для построения кодовых слов кода Адамара возьмем 4 строки матрицы H_2 , заменив в них -1 на 0 : $(1, 0, 0, 1)$; $(1, 1, 0, 0)$; $(1, 0, 1, 0)$; $(1, 1, 1, 1)$. Получены первые 4 кодовых слова. Следующие 4 кодовых слова получаются аналогичным образом из матрицы H_2 с помощью замен 1 на 0 и -1 на 1 : $(0, 1, 1, 0)$; $(0, 0, 1, 1)$; $(0, 1, 0, 1)$; $(0, 0, 0, 0)$. Всего получено 8 кодовых слов с минимальным расстоянием 2. Код не с очень большим минимальным расстоянием, но это из-за очень маленького числа $n = 4$. Отметим, что код нелинейный!

9.5. Метрика Хэмминга и метрика Ли

Пусть алфавит кода представляет собой символы некоторого поля $GF(q)$.

Метрику Хэмминга легко определить, пользуясь весовой функцией w_n . Для любого $\alpha \in GF(q)$

$$w_n(\alpha) = \begin{cases} 0, & \text{если } \alpha = 0 \\ 1, & \text{если } \alpha \neq 0 \end{cases}.$$

Тогда для слова $a = (\alpha_1, \alpha_2, \dots, \alpha_n)$ ($\alpha_i \in GF(q)$) полагаем $w_H(a) = \sum_{i=1}^n w_H(\alpha_i)$. Наконец, расстояние Хэмминга между словами a и b определяется как

$$w_H(a - b) = \rho_H(a, b).$$

Таким образом, расстояние Хэмминга для слов a и b — число несовпадающих позиций, что вполне согласуется с двоичным случаем.

Метрика Хэмминга хорошо соответствует схемам с амплитудной модуляцией. Для других способов модуляции она является весьма грубой. В случае схем с фазовой модуляцией более удобно пользоваться другой — метрикой Ли. Пусть $GF(q) = \mathbb{Z}_p$ — алфавит кода \mathcal{K} ($p > 2$). Пусть $C = (\alpha_1, \alpha_2, \dots, \alpha_n)$ — вектор, коэффициенты которого — элементы \mathbb{Z}_p . Определим вес Ли $w_L(\alpha_i)$ элемента α_i как $|\alpha_i|$, где $|\alpha_i| \equiv \pm \alpha_i \pmod{p}$, причем $0 \leq \alpha_i \leq p/2$. Тогда

$$w_L(C) = \sum_{i=1}^n w_L(\alpha_i).$$

Расстояние Ли $\rho_L(a, b)$ между векторами a и b определяется как вес Ли разности $w_L(a - b)$.

9.6. Границы возможного и невозможного в кодировании

Важным для теории кодирования является вопрос о взаимоотношениях между блоковой длиной, числом информационных символов, числом кодовых слов и минимальным расстоянием кода. Один пример такой границы (граница Синглтона) рассмотрен в 5.2. Обзор результатов, относящихся к различным границам, можно найти в [5], [7], [4], [11].

9.6.1.

Граница Хэмминга

Мы обозначаем через $M_q(n, d)$ максимальное число элементов q -значного кода $\mathcal{K} \subseteq X^n$, $|X| = q$, длины n с кодовым расстоянием d .

Обозначим, следуя [11], шар $V_{t,n}(x)$ радиуса t с центром в точке x в пространстве Хэмминга X^n , множество точек кода \mathcal{K} , отстоящих от x на расстояние не больше чем t . Легко вычислить (упражнение), что

$$|V_{t,n}(x)| = \sum_{s=0}^t (q-1)C_n^s.$$

Теорема 44. Для нечетного $d = 2t + 1$ справедлива оценка

$$|M_q(n, d)| \leq \frac{q^n}{|V_{t,n}(x)|} = \frac{q^n}{\sum_{s=0}^t (q-1)C_n^s}.$$

Граница Плоткина

Рассматриваемая оценка справедлива для кодов с «большим» кодовым расстоянием $d \geq n/2$.

Теорема 45. Если $d > n/2$, то

$$|M_2(n, d)| \leq \frac{2d}{2d - n}.$$

Если $n = 2d$, то $|M_2(n, d)| \leq 4d$.

Граница Синглтона

Теорема 46. Для линейного кода $\mathcal{K} \subseteq \mathbb{F}_q^n$ длины n , размерности k и с кодовым расстоянием d справедлива оценка

$$d \leq n - k + 1.$$

Коды, для которых достигается равенство, называются MDR-кодами (разделимыми кодами с максимальным расстоянием). Например, коды Рида–Соломона являются такими.

Граница Сидельникова

Двоичный код \mathcal{K} называется *антиподальным*, если для всякого $x \in \mathcal{K}$ вектор $\bar{x} = x + (1, 1, \dots, 1) \in \mathcal{K}$. Расстояние между различными векторами $x, y \in \mathcal{K}$ принимает значения из интервала $[d, n - d]$. Оценка максимального числа $M_A(n, d)$ элементов двоичного антиподального кода длины n с кодовым расстоянием d найдена В. М. Сидельниковым в 1971 году.

Теорема 47. (Сидельников) Справедлива оценка

$$|M_A(n, d)| \leq \frac{2n^3 - 2n(n - 2d)^2}{3n - (n - 2d)^2 - 2},$$

если $3n - (n - 2d)^2 - 2 > 0$ и $n \geq 4$.

9.7. Упражнения

1. Доказать, что код, дуальный к циклическому, будет циклическим.
2. Код Рида–Маллера первого порядка – это дуальный код к расширенному коду Хэмминга. Нарисовать кодирующую матрицу кода $RM(1, 3)$.
3. Для кода Рида–Маллера первого порядка $RM(1, 3)$ исправить или обнаружить ошибки в следующих словах: (11010001) , (01011111) .
4. Найти матрицу Адамара размера 8×8 .
5. Как выглядит кодирующая матрица для циклического кода Хэмминга с тремя проверками над $GF(3)$?
6. Найти антиподальный двоичный код длины 8.

7. Построить стандартное расположение для кода над $GF(3)$, заданного кодирующей матрицей размера 2×4 (выбрать матрицу самостоятельно). Определить параметры кода.

Часть II.

Элементы теории сжатия и восстановления информации

10. Методы сжатия информации

Сжатие данных является одной из форм обработки информации. Цель использования алгоритмов сжатия – сокращение объема памяти, необходимого для хранения данных или передачи информации по сетям связи.

Блоком будем называть конечную последовательность цифровой информации. Блок – последовательность с произвольным доступом.

Поток – последовательность цифровой информации с неизвестными границами.

Обрабатываемое сообщение порождается дискретным источником данных. Приведем формальное определение дискретного источника данных.

Пусть $X = \{x_1, x_2, \dots, x_m\}$ – множество, состоящее из m элементов. Элементы множества будем называть сообщениями.

Определение 24. *Дискретный источник – источник, выдающий в каждый определенный момент времени одно из сообщений дискретного множества X .*

Для полного задания источника необходимо дать вероятностное описание процесса появления сообщений на выходе источника, т. е. определить вероятности появления любых сочетаний сообщений. Если известно распределение вероятностей частот (то есть известно, сколько

раз встретился каждый символ и подсчитана его вероятность), то можно построить оптимальное кодирование. Задача усложняется в случае, если распределение частот символов неизвестно. В этом случае существуют два различных подхода к кодированию сообщений.

Первый: просмотреть входной поток и построить кодирование на основе собранной статистики (при этом потребуются два прохода по файлу), что ограничивает сферу применения таких алгоритмов. В выходной поток должна быть записана схема использованного кодирования, то есть модель, которая затем будет использоваться декодером (при восстановлении сжатой информации). Примером алгоритмов этой группы как раз и является алгоритм Хаффмана, который мы рассмотрим позднее.

Второй: использует так называемый адаптивный кодер. Идея таких алгоритмов состоит в том, чтобы менять схему кодирования в зависимости от исходных данных. Такой алгоритм использует один проход и не требует передачи информации в явном виде. Вместо этого декодер, считывая кодированный поток данных, синхронно с кодером изменяет схему кодирования, начиная с некоторой predetermined точки. При использовании такого подхода необходимо постоянно корректировать коды в соответствии с изменяющейся статистикой входного потока.

Алгоритмы сжатия данных применяются в следующих случаях: в программах-архиваторах, при передаче данных по сетям связи, в программах резервного копирования, в форматах файлов с мультимедиа данными (JPEG, GIF, MP3, AVI), в системах безопасности и видеонаблюдения, в системах телеконференций.

Методы сжатия данных разделяют на сжатие без потерь информации и сжатие с потерей информации.

Сжатие информации без потерь

На текущий момент существует большое количество алгоритмов сжатия без потерь, которое условно можно разделить на две большие группы:

1. Алгоритмы статистического (энтропийного) сжатия. Эта группа алгоритмов сжимает информацию, используя неравномерность частот, с которыми различные символы встречаются в сообщении. К алгоритмам этой группы относятся алгоритмы арифметического и префиксного кодирования.

2. Поточные и словарные алгоритмы. Особенностью всех алгоритмов этой группы является то, что при кодировании используется не инфор-

мация о частотах символов в сообщении, а информация о последовательностях, встречавшихся ранее.

В отдельную группу можно выделить алгоритмы преобразования информации. Алгоритмы этой группы не производят непосредственного сжатия информации, но их применение значительно упрощает дальнейшее сжатие с использованием поточных, словарных и энтропийных алгоритмов. Алгоритм сжатия RLE является самым простым в применении. Он сопоставляет серии битов значение счетчика – количества повторений. Во многих методах и алгоритмах сжатия мультимедиа данных после выполнения соответствующих преобразований формируется последовательность чисел; элементы этой последовательности содержат большое количество нулей. Это и создает возможности для сжатия.

Сжатие информации с потерями

При сжатии с потерями мы имеем дело с новой концепцией, которая предполагает сжатие путем удаления несущественной информации. Сжатие с потерями обычно используют при обработке изображений, звука, видео.

Изображение можно сжать с потерей, удалив несущественную информацию, даже если в нем нет избыточности. Имеются два вида избыточности в цифровых изображениях. Первый вид похож на избыточность в текстовом файле. В каждом случайном изображении некоторые цвета могут преобладать, другие же встречаться редко. Такая избыточность может быть удалена с помощью кодов переменной длины, присваиваемых разным пикселям, точно так же как и при сжатии текстовых файлов (сжатие без потерь). Другой вид избыточности гораздо более важен, он является результатом корреляции пикселей. Когда наш взгляд перемещается по картинке, он обнаруживает в большинстве случаев, что соседние пиксели окрашены в близкие цвета. Представим себе фотографию, на которой изображено голубое небо, белые облака, коричневые горы и зеленые деревья. Пока мы смотрим на горы, близкие пиксели имеют похожий цвет; все или почти все из них имеют разные оттенки коричневого цвета. Про близкие пиксели неба можно сказать, что они носят различные оттенки голубого цвета. И только на горизонте, там, где горы встречаются с небом, соседние пиксели могут иметь совершенно разные цвета. Таким образом, отдельные пиксели не являются совершенно независимыми. Можно сказать, что ближайшие пиксели изображения коррелируют между собой. Описанная избыточность называется *пространственной*.

Любое изображение можно оцифровать, разбив его на пикселы, а каждому пикселу приписать некоторое число. Точно так же звук можно оцифровать, разбив его на фрагменты и присвоив им некоторые числовые значения. Если записывать звук через микрофон, то он переводится в электрический сигнал, напряжение которого непрерывно зависит от времени. Это напряжение называется аналоговым представлением звука. Оцифровка звука делается с помощью измерения напряжения сигнала во многих точках оси времени, перевода каждого измерения в числовую форму и записи полученных чисел в файл. Этот процесс называется сэмплированием или отбором фрагментов. Для сжатия звука могут быть использованы алгоритмы сжатия без потерь (статистические или словарные), но результат существенно зависит от конкретных данных. Можно добиться лучших результатов при сжатии звука с потерей части аудиоинформации, развивая методы компрессии, которые учитывают особенности восприятия звука. Они удаляют ту часть данных, которая остается неслышимой для органов слуха. Идея *подавления пауз* заключается в рассмотрении малых сэмплов, как если бы их не было (то есть они равны нулю). Такое обнуление будет порождать серии нулей, поэтому метод подавления пауз на самом деле является вариантом RLE, приспособленным к сжатию звука. Этот метод основан на особенности звукового восприятия, которое состоит в терпимости уха человека к отбрасыванию еле слышных звуков. *Уплотнение* основано на том свойстве, что ухо лучше различает изменения амплитуды тихих звуков, чем громких. Метод сжатия на основе уплотнения сначала анализирует каждый сэмпл звукового файла и применяет к нему нелинейную функцию для сокращения числа бит, назначенных этому сэмплу.

Сжатие видео основано на двух важных принципах. Первый – это пространственная избыточность, присущая каждому кадру видеоряда. А второй принцип основан на том факте, что большую часть времени каждый кадр похож на своего предшественника. Это называется временной избыточностью. Таким образом, типичный метод сжатия видео начинается с кодирования первого кадра с помощью некоторого алгоритма компрессии изображения. Затем следует кодировать каждый последующий кадр, находя расхождение или разность между этим кадром и его предшественником и кодируя эту разность. Сжатие видео обычно допускает частичную потерю информации.

Важным вопросом в теории сжатия данных является оценки эффективности работы алгоритма. Для алгоритмов сжатия данных без потерь

можно выделить несколько важных критериев: время компрессии, время, затраченное на восстановление исходных данных, размер сжатого сообщения, степень (коэффициент), фактор и качество сжатия. В случае с алгоритмами сжатия данных с потерями также необходимо оценивать степень влияния потерь информации (то есть оценивать отличие между исходными и восстановленными данными).

Степень (коэффициент сжатия) – отношение размера выходного файла к размеру входного файла. Если значения коэффициента больше единицы, то сжатия не произошло, иначе величина показывает, насколько сжатый файл меньше исходного.

Фактор сжатия – обратное отношение (размера входного файла к размеру выходного файла). Чем больше фактор сжатия, тем эффективнее работает алгоритм.

Качество сжатия $k = 100 \cdot (1 - \text{коэффициент})$ показывает, на сколько процентов сжатый файл меньше исходного.

Определение 25. Пусть $E = \{0, 1\}$, $E^* = \bigcup_{i=1}^{\infty} E^i$ - множество всех наборов 0 и 1. Пусть $A = \{a_1, a_2, \dots\}$ - алфавит сообщения. Кодированием называется отображение $f : A \rightarrow E^*$. Образ $f(a_i)$ называется кодовым словом или кодом символа a_i . Длину кодового слова будем обозначать через $L(a_i) = |f(a_i)|$ - количество бит в слове.

11. Энтропия и информация

11.1. Энтропия

Рассмотрим дискретную случайную величину X , принимающую значения x_1, \dots, x_n с вероятностями p_1, \dots, p_n . Случайной величине X присуща степень неопределенности. Возникает вопрос: как измерить неопределенность?

11.1.1.

Чтобы ответить на этот вопрос, сравним между собой две системы, каждой из которых присуща некоторая неопределенность. В качестве первой системы возьмем монету, которая в результате бросания может оказаться в одном из двух состояний: 1) выпал герб и 2) выпала цифра.

В качестве второй — игральную кость, у которой шесть возможных состояний: 1, 2, 3, 4, 5 и 6. Спрашивается: неопределенность какой системы больше? Очевидно, второй, так как у нее больше возможных состояний, в каждом из которых она может оказаться с одинаковой вероятностью.

11.1.2.

Рассмотрим второй пример. Пусть случайная величина X имеет два значения x_1 и x_2 с вероятностями $P(X = x_1) = 0,01$ и $P(X = x_2) = 0,99$. Интуитивно понятно, что неопределенность этой случайной величины меньше, чем в примере с монетой, так как почти наверняка в испытании X примет значение x_2 .

В качестве меры неопределенности в теории информации принимается *энтропия*.

Определение 26. *Энтропией случайной величины X называется сумма произведений вероятностей различных состояний системы на логарифмы этих вероятностей, взятая с обратным знаком:*

$$H(X) = - \sum p_i \log p_i.$$

Логарифм в определении (26) может быть взят при любом основании большим 1. Перемена основания равносильна простому умножению энтропии на постоянное число, а выбор основания равносильен выбору определенной единицы измерения энтропии. Если за основание выбрано число 10, то говорят о «десятичных единицах» энтропии, если 2 — о «двоичных единицах». На практике удобнее всего пользоваться логарифмами при основании 2 и измерять энтропию в двоичных единицах. В дальнейшем мы будем везде, если не оговорено противное, под символом \log понимать двоичный логарифм.

11.1.3.

Вычислим энтропию случайной величины X :

X	0	1
p_i	0,5	0,5

$H(X) = -0 \cdot \log_2(0,5) - 1 \cdot \log_2(0,5) = 1$. Полученная единица энтропии называется *битом*.

Энтропия $H(X)$ обладает рядом свойств, оправдывающих ее выбор в качестве характеристики степени неопределенности. Во-первых, она обращается в нуль, когда одно из состояний системы достоверно, а другие — невозможны. Во-вторых, при заданном числе состояний она обращается в максимум, когда эти состояния равновероятны, а при увеличении числа состояний увеличиваются. Наконец, и это самое главное, она обладает свойством аддитивности, т. е., когда несколько независимых систем объединяются в одну, их энтропии складываются.

Теорема 48. *Энтропия дискретной случайной величины с конечным множеством значений достигает максимума, когда все значения равновероятны.*

Доказательство см. [8]

11.2. Энтропия двумерной случайной величины

Рассмотрим двумерную случайную величину (X, Y) , которая принимает значения (x_i, y_j) с вероятностями $p_{ij} = P((X, Y) = (x_i, y_j))$. Энтропия (X, Y) определяется естественным образом:

$$H(X, Y) = - \sum p_{ij} \log p_{ij}.$$

Пусть X и Y — независимые случайные величины. Тогда верна следующая формула:

$$H(X, Y) = H(X) + H(Y).$$

Доказательство см. ([8]).

Эта формула может быть обобщена на случай n независимых случайных величин.

11.3. Условная энтропия

Пусть имеются две случайные величины X и Y . Обозначим $P(y_j|x_i)$ условную вероятность того, что случайная величина Y примет значение y_j при условии, что случайная величина $X = x_i$.

Определение 27. Условной энтропией случайной величины Y при условии, что случайная величина $X = x_i$ называется число

$$H(Y|x_i) = - \sum_{j=1}^n P(y_j|x_i) \cdot \log P(y_j|x_i).$$

Условная энтропия зависит от того, какое значение приняла случайная величина X . Для одних значений она будет больше, для других — меньше. Определим среднюю, или полную, энтропию случайной величины Y с учетом того, что система может принимать разные состояния. Определим полную условную энтропию $H(Y|X)$:

$$H(Y|X) = \sum p_i H(Y|x_i).$$

Величина $H(Y|X)$ характеризует степень неопределенности случайной величины Y , остающуюся после того, как состояние случайной величины X полностью определилось.

Теорема 49.

$$H(X, Y) = H(X) + H(Y|X).$$

Доказательство см. ([8]).

11.4. Энтропия и сжатие информации

11.4.1. Информация

Рассмотрим некоторую систему X , над которой производится наблюдение, и оценим информацию, получаемую в результате того, что состояние системы X становится полностью известным. До получения сведений энтропия системы была $H(X)$, после получения сведений состояние системы полностью определилось, т. е. энтропия стала равной нулю. Естественно определить информацию $I(X)$, получаемую в результате выяснения состояния системы X равную энтропии $H(X)$. Таким образом,

$$I(X) = - \sum p_i \log p_i. \quad (1)$$

То есть, информация $I(X)$ представится как средняя информация, получаемая от всех возможных отдельных сообщений с учетом их вероятностей. Формулу (1) можно переписать в виде:

$$I(X) = M(-\log P(X)).$$

Каждое отдельное слагаемое $-\log p_i$ в формуле (1) естественно рассматривать как частную информацию, получаемую от отдельного сообщения, состоящего в том, что система X находится в состоянии x_i . Частную информацию обычно обозначают I_{x_i} .

11.4.2. Сжатие данных

С помощью понятия энтропии теория информации показывает, как вычислять вероятности строк символов алфавита, и предсказывает ее наилучшее сжатие то есть наименьшее, в среднем, число бит, необходимое для представления этой строки символов.

Цель сжатия — уменьшение количества бит, необходимых для хранения или передачи заданной информации, что дает возможность передавать сообщения более быстро и хранить более экономно и оперативно (последнее означает, что операция извлечения данной информации с устройства ее хранения будет проходить быстрее, что возможно, если скорость распаковки данных выше скорости считывания данных с носителя информации). Сжатие данных не может быть больше некоторого теоретического предела.

Продemonстрируем это на простом примере. Для последовательности символов ((ABCDE» с вероятностями 0.5, 0.2, 0.1, 0.1 и 0.1, соответственно, вероятность строки «AAAAABBCDE» равна $P = 0.5^5 \times 0.2^2 \times 0.1^3 = 1.25 \times 10^{-6}$. Логарифм по основанию 2 этого числа $\log_2 P = -19.6096$. Тогда наименьшее в среднем число требуемых бит для кодирования этой строки равно $[-\log_2 P]$, то есть 20. Кодер, достигающий этого сжатия, называется *энтропийным кодером*.

Доказано, что среднее количество бит, приходящихся на одно кодируемое значение д. с. в., не может быть меньше, чем энтропия этой д. с. в., т. е. $M(L(X)) \geq H(X)$, где $L(X)$ — длина кода значения X , для любой д.с.в. X и любого ее кода.

Кроме того, доказано утверждение о том, что существует такое кодирование

(Шеннона-Фэно), что $H(X) \geq M(L(X)) - 1$.

Теорема 50. *С ростом длины n сообщения, при кодировании методом Шеннона-Фэно всего сообщения целиком среднее количество бит*

на единицу сообщения будет сколь угодно мало отличаться от энтропии единицы сообщения.

Подобное кодирование практически нереализуемо из-за того, что с ростом длины сообщения трудоемкость построения этого кода становится недопустимо большой. Кроме того, такое кодирование делает невозможным отправку сообщения по частям, что необходимо для непрерывных процессов передачи данных. Дополнительным недостатком этого способа кодирования является необходимость отправки или хранения собственно полученного кода вместе с его исходной длиной, что снижает эффект от сжатия. На практике для повышения степени сжатия используют метод блокирования.

Выбирая множество кодов переменной длины, необходимо соблюдать два принципа: (1) следует назначать более короткие коды чаще встречающимся символам и (2) коды должны удовлетворять свойству префикса: если некоторая последовательность битов выбрана в качестве кода какого-то символа, то ни один код другого символа не должен иметь в начале эту последовательность (не может быть префиксом то есть приставкой). Следуя этим принципам, можно построить короткие, однозначно декодируемые коды, но не обязательно наилучшие (то есть самые короткие) коды.

11.5. Упражнения

1. Вычислите энтропию случайной величины X .

X	1	2	3
p	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Придумайте для X три разных кода. Для каждого из них вычислите среднюю длину.

2. Вычислите энтропию случайной величины X .

X	1	2	3
p	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{98}{100}$

Придумайте для X три разных кода. Для каждого из них вычислите среднюю длину.

3. Вычислите энтропию случайной величины X .

X	1	2	3	4	5	6	7	8	9	10
p	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$

Придумайте для X три разных кода. Для каждого из них вычислите среднюю длину.

4. Вычислите энтропию случайной величины X .

X	1	2	3	4	5	6	7	8	9	10
p	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{1}{100}$	$\frac{91}{10}$

Придумайте для X три разных кода. Для каждого из них вычислите среднюю длину.

5. Вычислите энтропию равномерного распределения.
6. Монету подбрасывают 5 раз. Случайная величина X – количество выпавших «гербов». Вычислите энтропию с. в. X . Придумайте для X три разных кода. Для каждого из них вычислите среднюю длину.
7. Игральный кубик подбрасывают 5 раз. Случайная величина X – количество выпавших «шестерок». Вычислите энтропию с. в. X . Придумайте для X три разных кода. Для каждого из них вычислите среднюю длину.
8. Вычислите энтропию биномиального распределения с параметрами n и p .
9. Монету подбрасывают до тех пор, пока не выпадет «герб». Случайная величина X – количество подбрасываний. Вычислите энтропию с. в. X .
10. Вычислите энтропию геометрического распределения.

12. Алгоритм Шеннона–Фэно

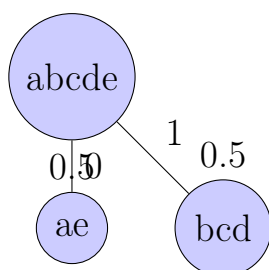
12.1.

Алгоритм Шеннона–Фэно состоит в следующем, значения д. с. в. располагают в порядке убывания их вероятностей, а затем последовательно

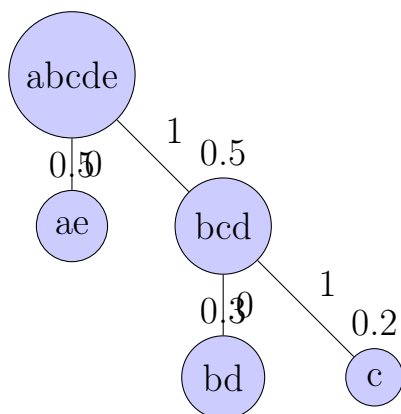
делят на две части с приблизительно равными вероятностями, к коду первой части добавляют 0, а к коду второй — 1.

Рассмотрим алгоритм Шеннона–Фэно более подробно на примере. Пусть задан алфавит, состоящий из пяти символов $A = \{a, b, c, d, e\}$ с вероятностями $P(a) = 0.4$, $P(b) = 0.2$, $P(c) = 0.2$, $P(d) = 0.1$, $P(e) = 0.1$. Для наглядности будем строить граф.

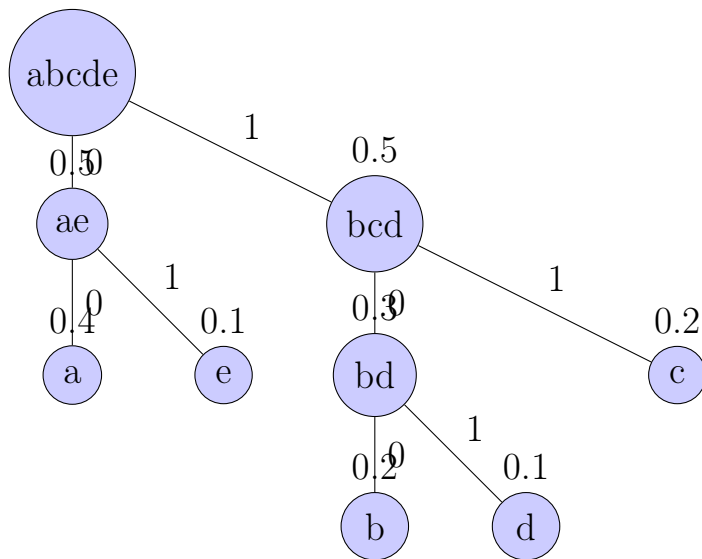
Первый шаг. Делим символы на две группы с примерно равными вероятностями: a и bcd .



Второй шаг. Делим множество bcd на две группы с примерно равными вероятностями. Получаем граф:



Третий и четвертый шаги. Делим каждое из множеств ae и bd на две группы с примерно равными вероятностями. Получаем граф:



Спускаясь от начальной вершины графа до конечной вершины, получаем код символа. В нашем примере это:

$a - 00,$
 $b - 100,$
 $c - 11,$
 $d - 101,$
 $e - 01.$

Наш код получился префиксным.

Подсчитаем среднюю длину получившегося кода:

$$L(C) = 0.4 \cdot 2 + 0.2 \cdot 3 + 0.2 \cdot 2 + 0.1 \cdot 3 + 0.1 \cdot 2 = 2.3 \text{ бит/символ.}$$

Вычислим энтропию случайной величины X :

X	a	b	c	d	e
p	0.4	0.2	0.2	0.1	0.1

$$H(X) = -(0.4 \log 0.4 + 0.2 \log 0.2 + 0.2 \log 0.2 + 0.1 \log 0.1 + 0.1 \log 0.1) = 2.1.$$

Средняя длина кода очень близка к теоретическому пределу!

Заметим, что были разные варианты построения графа. Например, на первом шаге мы могли бы сформировать две группы по-другому: ad и bce . Средняя длина кода при этом осталась бы без изменений. Предлагаем читателю убедиться в этом при решении упражнений.

12.1.1.

Рассмотрим строку *aaabbccde*. Код, соответствующий этой строке, – 00000001001001110101. Так как наш код префиксный, то декодирование осуществляется естественным образом. Первый символ кода 0. Поэтому первая буква или *a*, или *e*. Смотрим на второй символ. Это – 0. Поэтому первая буква – *a*. Далее аналогично.

Для определенности будем считать, что буквы нашего алфавита заданы в 8-битной кодировке. Поэтому наша строка *aaabbccde* занимает в памяти $8 \times 8 = 64$ бит. Наш код – 19 бит. Коэффициент сжатия $k = \frac{64}{19} = 3,37$.

12.2. Упражнения

1. Имеется алфавит $A = \{a, b, c, d, e\}$, $P(a) = 0.2$, $P(b) = 0.1$, $P(c) = 0.3$, $P(d) = 0.1$, $P(e) = 0.3$. Составить код Фэно–Шеннона. Получить код для строки «*abcdeabcdeabcdeabcde*». Вычислите среднюю длину и степень сжатия.
2. Имеется алфавит $A = \{a, b, c, d, e\}$, $P(a) = 0.1$, $P(b) = 0.1$, $P(c) = 0.1$, $P(d) = 0.1$, $P(e) = 0.6$. Составить код Фэно–Шеннона. Получить код для строки *abcdeabcdeabcdeabcde*. Вычислите среднюю длину и степень сжатия.
3. Имеется алфавит $A = \{a, b, c, d, e\}$, $P(a) = 0.2$, $P(b) = 0.2$, $P(c) = 0.2$, $P(d) = 0.2$, $P(e) = 0.2$. Составить код Фэно–Шеннона. Получить код для строки *abcdeabcdeabcdeabcde*. Вычислите среднюю длину и степень сжатия.
4. Имеется алфавит $A = \{a, b, c, d, e, f, g\}$, $P(a) = 0.01$, $P(b) = 0.01$, $P(c) = 0.01$, $P(d) = 0.01$, $P(e) = 0.01$, $P(f) = 0.05$, $P(g) = 0.9$. Получить код для строки *abcdefgabcdefgabcdefgabcdefg*. Составить код Фэно–Шеннона. Вычислите среднюю длину и степень сжатия.
5. Алфавит состоит из двух букв, *a* и *b*, встречающихся с вероятностями $P(a) = 0,8$ и $P(b) = 0,2$. Примените метод Фэно–Шэннона к кодированию всевозможных двухбуквенных и трехбуквенных комбинаций.
6. Составьте код Фэно–Шеннона для строки *abcdeabcde*. Вычислите среднюю длину и степень сжатия.

7. Составьте код Фэно–Шеннона для строки *aaaaabbbcc*. Вычислите среднюю длину и степень сжатия.
8. Составьте код Фэно–Шеннона для строки *aaaaaaaaabbbbcccd*. Вычислите среднюю длину и степень сжатия.
9. Составьте код Фэно–Шеннона для строки *aaaaaaaaaaaaabcd*. Вычислите среднюю длину и степень сжатия.
10. Придумайте фразу, получите для нее код Фэно–Шеннона и передайте соседу для декодирования. Сравните результат декодирования с оригиналом.

13. Алгоритм Хаффмана

13.1.

Кодирование Хаффмана является простым алгоритмом для построения кодов переменной длины, имеющих минимальную среднюю длину. Этот весьма популярный алгоритм служит основой многих компьютерных программ сжатия текстовой и графической информации. Некоторые из них используют непосредственно алгоритм Хаффмана, а другие берут его в качестве одной из ступеней многоуровневого процесса сжатия. Метод Хаффмана производит идеальное сжатие (то есть сжимает данные до их энтропии), если вероятности символов точно равны отрицательным степеням числа 2. Алгоритм начинает строить кодовое дерево снизу вверх, затем скользит вниз по дереву, чтобы построить каждый индивидуальный код справа налево (от самого младшего бита к самому старшему).

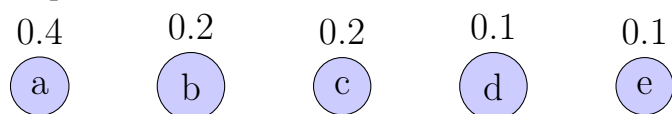
Алгоритм начинается составлением списка символов алфавита в порядке убывания их вероятностей. Затем от корня строится дерево, листьями которого служат эти символы. Это делается по шагам, причем на каждом шаге выбираются два символа с наименьшими вероятностями, добавляются наверх частичного дерева, удаляются из списка и заменяются вспомогательным символом, представляющим эти два символа. Вспомогательному символу приписывается вероятность, равная сумме вероятностей, выбранных на этом шаге символов. Когда список сокращается до одного вспомогательного символа, представляющего весь ал-

фавит, дерево объявляется построенным. Завершается алгоритм спуском по дереву и построением кодов всех символов.

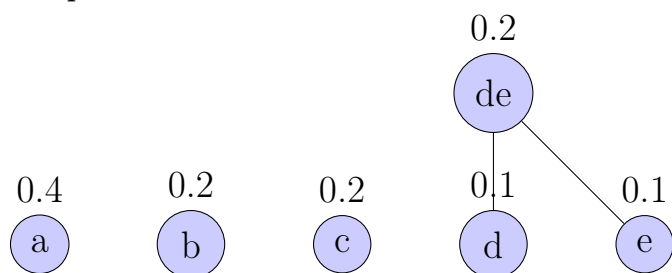
13.1.1.

Проиллюстрируем работу алгоритма на примере 12.1. Итак, имеется алфавит, состоящий из пяти символов $A = \{a, b, c, d, e\}$ с вероятностями $P(a) = 0.4$, $P(b) = 0.2$, $P(c) = 0.2$, $P(d) = 0.1$, $P(e) = 0.1$. Будем строить граф по шагам.

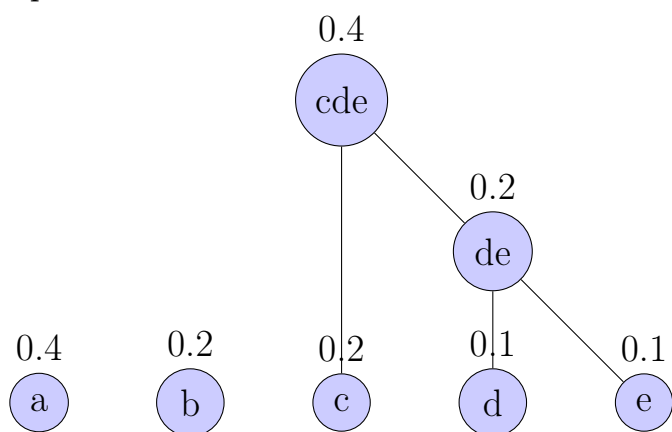
Первый шаг.



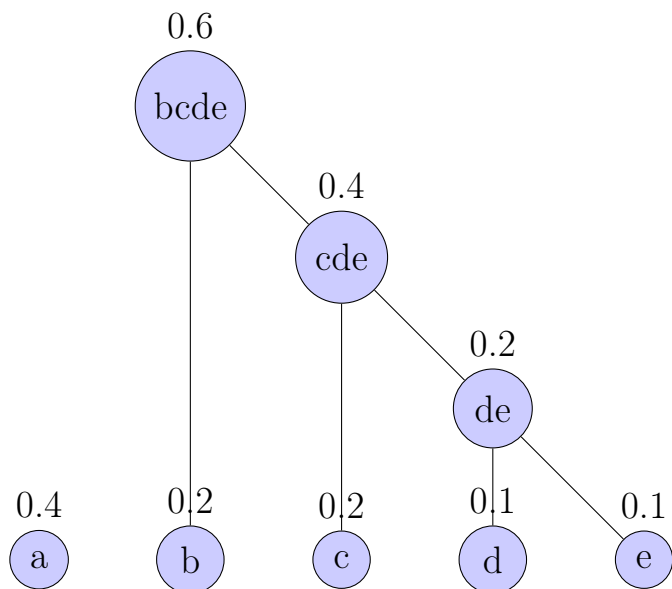
Второй шаг.



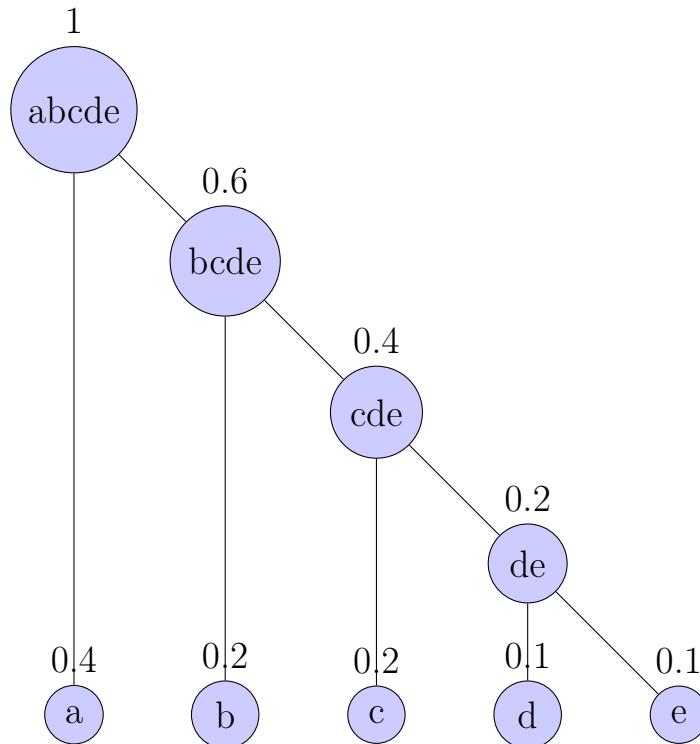
Третий шаг.



Четвертый шаг.



Пятый шаг.



Наш код:

a - 0,

b - 10,

c - 110,

d - 1110,

e - 1111.

Средняя длина равна

$$L(C) = 0,4 \cdot 1 + 0,2 \cdot 2 + 0,2 \cdot 3 + 0,1 \cdot 4 + 0,1 \cdot 4 = 2,2 \text{ бит/символ.}$$

Чуть лучше, чем алгоритм Шэннона-Фэно и еще ближе к теоретическому пределу $H(X) = 2,1$. Также как и в предыдущем параграфе, у нас есть некоторый произвол в построении графа.

13.1.2.

Рассмотрим строку $aaabbcde$. Код, соответствующий этой строке, — 000101011011101111. Декодирование осуществляется естественным образом.

Наша строка $aaabbcde$ занимает в памяти $8 \times 8 = 64$ бит. Наш код — 18 бит. Коэффициент сжатия $k = \frac{64}{18} = 3,56$.

13.2. Упражнения

В каждом упражнении сравните результат с полученным ответом в соответствующей задаче в пункте 10.5.

1. Имеется алфавит $A = \{a, b, c, d, e\}$, $P(a) = 0.2$, $P(b) = 0.1$, $P(c) = 0.3$, $P(d) = 0.1$, $P(e) = 0.3$. Составить код Хаффмана. Получить код для строки $abcdeabcdeabcdeabcde$. Вычислите среднюю длину и степень сжатия.
2. Имеется алфавит $A = \{a, b, c, d, e\}$, $P(a) = 0.1$, $P(b) = 0.1$, $P(c) = 0.1$, $P(d) = 0.1$, $P(e) = 0.6$. Составить код Хаффмана. Получить код для строки $abcdeabcdeabcdeabcde$. Вычислите среднюю длину и степень сжатия.
3. Имеется алфавит $A = \{a, b, c, d, e\}$, $P(a) = 0.2$, $P(b) = 0.2$, $P(c) = 0.2$, $P(d) = 0.2$, $P(e) = 0.2$. Составить код Хаффмана. Получить код для строки $abcdeabcdeabcdeabcde$. Вычислите среднюю длину и степень сжатия.
4. Имеется алфавит $A = \{a, b, c, d, e, f, g\}$, $P(a) = 0.01$, $P(b) = 0.01$, $P(c) = 0.01$, $P(d) = 0.01$, $P(e) = 0.01$, $P(f) = 0.05$, $P(g) = 0.9$. Получить код для строки $abcdefgabcdefgabcdefgabcdefg$. Составить код Хаффмана. Вычислите среднюю длину и степень сжатия.
5. Алфавит состоит из двух букв, a и b , встречающихся с вероятностями $P(a) = 0,8$ и $P(b) = 0,2$. Примените метод Хаффмана к кодированию всевозможных двухбуквенных и трехбуквенных комбинаций.
6. Составьте код Хаффмана для строки $abcdeabcde$. Вычислите среднюю длину и степень сжатия.
7. Составьте код Хаффмана для строки $aaaaabbbcc$. Вычислите среднюю длину и степень сжатия.
8. Составьте код Хаффмана для строки $aaaaaaaaabbbbcccd$. Вычислите среднюю длину и степень сжатия.
9. Составьте код Хаффмана для строки $aaaaaaaaaaaaaaaaabcd$. Вычислите среднюю длину и степень сжатия.

10. Придумайте фразу, получите для нее код Хаффмана и передайте соседу для декодирования. Сравните результат декодирования с оригиналом.

14. Арифметическое кодирование

14.1.

Метод Хаффмана является простым и эффективным, однако он порождает наилучшие коды переменной длины (коды, у которых средняя длина равна энтропии алфавита), только когда вероятности символов алфавита являются степенями числа 2, то есть равны $1/2$, $1/4$, $1/8$ и т. п. Это связано с тем, что метод Хаффмана присваивает каждому символу алфавита код с целым числом битов. Теория информации предсказывает, что при вероятности символа, скажем, 0.4, ему в идеале следует присвоить код длины 1.32 бита, поскольку $-\log_2 0.4 = 1.32$. А метод Хаффмана присвоит этому символу код длины 1 или 2 бита.

Арифметическое кодирование решает эту проблему путем присвоения кода всему обычно большому передаваемому файлу вместо кодирования отдельных символов. (Входным файлом может быть текст, изображение или данные любого вида.) Алгоритм читает входной файл символ за символом и добавляет биты к сжатому файлу.

Сначала требуется составить вероятностную модель алфавита, то есть вычислить или оценить вероятности появления каждого символа. Далее, для наглядности, рассмотрим интервал $[0; 1)$. Разделим этот интервал на части пропорционально вероятностям символов алфавита. Выбираем ту часть, которая соответствует текущему символу. Назначаем выбранный интервал новым и повторяем указанные действия для следующего символа. Кодом файла будет считаться любая точка последнего интервала.

14.1.1.

Рассмотрим строку *aabcb*. Пусть вероятностная модель имеет вид: $P(a) = 0.4$, $P(b) = 0.4$, $P(c) = 0.2$.

Первый шаг. Разбиваем интервал $[0; 1)$ соответственно вероятностной модели. Получаем интервалы: $[0; 0.4)$ – символ *a*, $[0.4; 0.8)$ – символ *b*,

$[0.8; 1)$ – символ c . Первый символ строки – a , поэтому выбираем интервал $[0; 0.4)$. Далее работаем с интервалом $[0; 0.4)$.

Второй шаг. Разбиваем интервал $[0; 0.4)$ соответственно вероятностной модели. Получаем интервалы: $[0; 0.16]$ – символ a , $[0.16; 0.32]$ – символ b , $[0.32; 0.4)$ – символ c . Читаем следующий символ в строке – a . Выбираем соответствующий интервал – $[0; 0.16)$.

Третий шаг. Разбиваем интервал $[0; 0.16)$ соответственно вероятностной модели. Получаем интервалы: $[0; 0.064]$ – символ a , $[0.064; 0.128]$ – символ b , $[0.128; 0.16)$ – символ c . Читаем следующий символ в строке – b . Выбираем соответствующий интервал – $[0.064; 0.128)$.

Четвертый шаг. Разбиваем интервал $[0.064; 0.128)$ соответственно вероятностной модели. Получаем интервалы: $[0.064; 0.0896]$ – символ a , $[0.0896; 0.1152]$ – символ b , $[0.1152; 0.128)$ – символ c . Читаем следующий символ в строке – c . Выбираем соответствующий интервал – $[0.1152; 0.128)$.

Пятый шаг. Разбиваем интервал $[0.1152; 0.128)$ соответственно вероятностной модели. Получаем интервалы: $[0.1152; 0.12032]$ – символ a , $[0.12032; 0.12544]$ – символ b , $[0.12544; 0.128)$ – символ c . Читаем следующий символ в строке – b . Выбираем соответствующий интервал – $[0.12032; 0.12544)$. Любая точка из этого интервала будет кодом нашей строки. Естественно выбрать число с наименьшим количеством знаков. Выберем число 0.121. Ясно, что кодом можно считать число 121.

Итак, исходная строка занимала $5 \times 8 = 40$ бит. Число $121 = 1111001_2$ – 7 бит. Коэффициент сжатия $k = \frac{40}{7} = 5.7$.

Упражнение. Пусть алфавит содержит символы a_1, a_2, \dots, a_n с вероятностями p_1, p_2, \dots, p_n соответственно. Дан интервал $[x; y)$. Выпишите формулы для нового интервала, который соответствует символу a_i .

Замечание. Описанный процесс кодирования невозможно реализовать на практике, так как мы оперируем с очень большими действительными числами, что требует огромных вычислительных ресурсов. Любое практическое применение арифметического кодирования должно основываться на оперировании с целыми числами, которые не могут быть слишком длинными. В процессе кодирования границы текущего интервала сходятся все ближе друг к другу. Поэтому некоторое количество первых знаков после запятой не будут меняться далее. Эти знаки можно запомнить (записать в файл) и не использовать в дальнейших вычислениях.

Процесс декодирования по сути повторяет процедуру кодирования: на каждом шаге мы выбираем тот интервал, которому принадлежит код,

параллельно восстанавливая символы. Заметим, что для декодирования необходимо знать вероятностную модель, а также понимать, когда остановить декодирования. Для последнего можно запомнить длину исходной строки или использовать специальный символ конца строки.

Рассмотрим процесс декодирования на нашем примере. Для декодирования нам необходимо знать вероятностную модель, сам код и, например, длину исходного сообщения (можно использовать специальный символ окончания строки). Итак, имеем код 0.121. Это число принадлежит интервалу, который соответствует символу a (см. первый шаг примера выше).

14.2. Адаптивное арифметическое кодирование

В случае когда вероятностная модель неизвестна, при арифметическом кодировании ее можно строить по мере чтения исходной строки. Декодер, соответственно, должен восстанавливать вероятностную модель. Алгоритм кодирования состоит из двух частей: вероятностной модели и арифметического кодера. Вероятностная модель читает следующий символ из входного файла и вызывает кодер, сообщая ему символ и две текущие накопленные частоты. Затем модель увеличивает на единицу счетчик символов и изменяет накопленные частоты. Главным здесь является то, что вероятности символов определяются моделью по старым значениям счетчиков, которые увеличиваются только после кодирования данного символа. Это позволяет декодеру делать зеркальные действия. Кодер знает символ, который ему предстоит закодировать, а декодер должен его распознать по сжатому коду, поэтому декодер знает только старые значения счетчиков, но может увеличивать и изменять их значения точно так же, как и кодер.

Рассмотрим пример. Закодируем строку $aabcb$. Будем считать, что нам известен алфавит $A = \{a, b, c\}$. Изначально каждому символу алфавита присваиваем частоту 1. Тогда вероятностная модель будет иметь вид: $P(a) = \frac{1}{3}$, $P(b) = \frac{1}{3}$, $P(c) = \frac{1}{3}$.

Первый шаг. Читаем символ – a . Соответствующий интервал – $[0; \frac{1}{3})$. Увеличиваем на единицу частоту символа a . Меняем вероятностную модель: $P(a) = \frac{2}{4}$, $P(b) = \frac{1}{4}$, $P(c) = \frac{1}{4}$.

Второй шаг. Читаем следующий символ – a . Интервал – $[0; \frac{1}{6})$. Увеличиваем частоту символа a на единицу. Вероятностная модель имеет вид: $P(a) = \frac{3}{5}$, $P(b) = \frac{1}{5}$, $P(c) = \frac{1}{5}$.

Третий шаг. Читаем символ – b . Интервал – $[\frac{3}{30}; \frac{4}{30})$. Увеличиваем частоту символа b на единицу. Вероятностная модель: $P(a) = \frac{3}{6}$, $P(b) = \frac{2}{6}$, $P(c) = \frac{1}{6}$.

Четвертый шаг. Читаем символ – c . Интервал – $[\frac{23}{180}; \frac{24}{180})$. Увеличиваем частоту символа c на единицу. Вероятностная модель принимает вид: $P(a) = \frac{3}{7}$, $P(b) = \frac{2}{7}$, $P(c) = \frac{2}{7}$.

Пятый шаг. Читаем символ – b . Интервал – $[\frac{166}{1260}; \frac{168}{1260})$. В качестве кода нашей строки можно взять число 0.132.

14.3. Упражнения

1. Составьте код для строки $ababc$ с помощью арифметического кодирования. Вычислите среднюю длину и степень сжатия.
2. Составьте код для строки $abcdeabcde$ с помощью арифметического кодирования. Вычислите среднюю длину и степень сжатия.
3. Составьте код для строки $aaaaabbbcc$ с помощью арифметического кодирования. Вычислите среднюю длину и степень сжатия.
4. Составьте код для строки $aaaaaaaaabbbbcccd$ с помощью арифметического кодирования. Вычислите среднюю длину и степень сжатия.
5. Составьте код для строки $aaaaaaaaaaaaaaaaabcd$ с помощью арифметического кодирования. Вычислите среднюю длину и степень сжатия.
6. Придумайте фразу, получите для нее код с помощью арифметического кодирования и передайте соседу для декодирования. Сравните результат декодирования с оригиналом.

15. Словарные алгоритмы сжатия информации

Методы, основанные на словарном подходе, не рассматривают статистические модели; они также не используют коды переменной длины. Вместо этого они выбирают некоторые последовательности символов, сохраняют их в словаре, а все последовательности кодируются в виде меток, которые ссылаются на словарь. Словарь может быть статическим или динамическим (адаптивным). Первый является постоянным; иногда

в него добавляют новые последовательности, но никогда не удаляют. Динамический словарь содержит последовательности, ранее поступившие из входного файла, при этом разрешается и добавление, и удаление данных из словаря по мере чтения входного файла. Простейшим примером статического словаря может служить словарь английского языка, используемый для сжатия английских текстов. Представьте себе словарь, в котором содержится полмиллиона слов (без их описания или перевода). Слово (последовательность символов, заканчивающаяся пробелом или знаком препинания) читается из входного файла и пишется в словаре. Если оно найдено в словаре, его индекс, или словарная метка, записывается в выходной файл. В противном случае записывается само это слово без сжатия. В результате выходной файл состоит из индексов и рядов слов, поэтому необходимо уметь делать различие между ними. Далее мы рассмотрим два классических словарных алгоритма сжатия данных - LZ77 и LZ78.

15.1. LZ77

Основная идея этого метода состоит в использовании ранее прочитанной части входного файла в качестве словаря. Кодер создает окно для входного файла и двигает его справа налево в виде строки символов, требующих сжатие. Таким образом, метод основан на скользящем окне. Окно разбивается на две части. Часть слева называется буфером поиска. Она будет служить текущим словарем, и в ней всегда содержатся символы, которые недавно поступили и были закодированы. Правая часть окна называется упреждающим буфером, содержащим текст, который будет сейчас закодирован. На практике буфер поиска состоит из нескольких тысяч байт, а длина упреждающего буфера равна нескольким десяткам символов. Рассмотрим работу алгоритма на примере.

15.1.1.

Дана строка *abracadabraabracadabra*. Кодер читает левый символ в упреждающем буфере и ищет совпадения в буфере поиска, просматривая его справа налево. Далее кодер определяет, сколько совпадающих символов следует за найденным совпадением. Потом кодер продолжает искать совпадения в буфере поиска. В результате выбирается самая длинная строка совпадения. При равенстве длин выбирается самая удаленная(находящаяся правее). В выходной файл вместо найденной строки

записывается метка, состоящая из трех полей: смещение, длина, следующий символ в упреждающем буфере. Если совпадений нет, то двигается окно. Для наглядности представим работу кодера в виде таблицы, в которой каждая строка соответствует одному шагу.

	Буфер поиска	Упреждающий буфер	Метка
1		<i>abracadabraabracadabra</i>	(0,0, <i>a</i>)
2	<i>a</i>	<i>bracadabraabracadabra</i>	(0,0, <i>b</i>)
3	<i>ab</i>	<i>racadabraabracadabra</i>	(0,0, <i>r</i>)
4	<i>abr</i>	<i>acadabraabracadabra</i>	(3,1, <i>c</i>)
5	<i>abrac</i>	<i>adabraabracadabra</i>	(5,1, <i>d</i>)
6	<i>abracad</i>	<i>abraabracadabr</i>	(7,4, <i>a</i>)
7	<i>abracadabraa</i>	<i>bracadabr</i>	(11,9,eof)

Итак, наш код имеет вид: (0,0,*a*)(0,0,*b*)(0,0,*r*)(3,1,*c*)(5,1,*d*)(7,4,*a*)(11,9,eof). Оценим степень сжатия. Можем считать, что каждая метка занимает $4 + 4 + 8 = 16$ бит. В результате получаем 112 бит. Исходная строка занимала 176 бит. Коэффициент сжатия $k = \frac{176}{112} = 1.57$. Смещение и длину можно закодировать с помощью кода переменной длины, тогда степень сжатия увеличится. Ясно, что словарные алгоритмы не работают эффективно на строках малой длины. Декодирование производится естественным образом.

15.1.2.

Перед тем, как обсуждать алгоритм LZ78, остановимся на недостатках метода LZ77 и его вариантов. Было уже отмечено, что этот алгоритм основывается на предположении, что похожие образцы сжимаемых данных находятся близко друг от друга. Если содержимое файла не удовлетворяет этому условию, то он будет сжиматься плохо.

Другой недостаток метода – это ограниченные размеры упреждающего буфера. Размер совпадающей строки лимитирован длиной L упреждающего буфера, которую приходится держать небольшой, так как процесс сравнения строк основан на сравнении индивидуальных символов. Если удвоить L , то степень сжатия могла бы возрасти, но одновременно с этим произойдет замедление процесса поиска совпадений. Размер S буфера поиска тоже ограничен. Большой буфер поиска мог бы тоже улучшить компрессию, но опять возрастет сложность поиска. Увеличение буферов также означает удлинение меток, что ухудшает сжатие.

15.2. LZ78

Метод LZ78 не использует буфер поиска, упреждающий буфер и скользящее окно. Вместо этого имеется словарь встретившихся ранее строк. В начале этот словарь пуст (или почти пуст), и размер этого словаря ограничен только объемом доступной памяти. На выход кодера поступает последовательность меток, состоящих из двух полей. Первое поле – это указатель на строку в словаре, а второе – код символа. Метка не содержит длины строки, поскольку строка берется из словаря. Каждая метка соответствует последовательности во входном файле, и эта последовательность добавляется в словарь после того, как метка записана в выходной сжатый файл. Ничего из словаря не удаляется, что является одновременно и преимуществом над LZ77 (поскольку будущие строки могут совпадать даже с очень давними последовательностями) и недостатком (так как быстро растет объем словаря).

Словарь начинает строиться из пустой строки в позиции нуль. По мере поступления и кодирования символов новые строки добавляются в позиции 1, 2 и т. д. Когда следующий символ x читается из входного файла, в словаре ищется строка из одного символа x . Если такой строки нет, то x добавляется в словарь, а на выход подается метка $(0, x)$. Если вхождение символа x обнаружено (скажем, в позиции 37), то читается следующий символ y и в словаре ищется вхождение двухсимвольной строки xy . Если такое не найдено, то в словарь записывается строка xy , а на выход подается метка $(37, y)$. Такая метка означает строку xy , так как позицию 37 в словаре занимает символ x . Процесс продолжается до конца входного файла.

В общем случае текущий символ читается и становится однобуквенной строкой. Затем кодер пытается найти ее в словаре. Если строка найдена, читается следующий символ и присоединяется к текущей строке, образуя двухбуквенную строку, которую кодер опять пытается найти в словаре. До тех пор пока такие строки находятся в словаре, происходит чтение новых символов и их присоединение к текущей строке. В некоторый момент такой строки в словаре не оказывается. Тогда кодер добавляет ее в словарь и строит метку, в первом поле которой стоит указатель на последнюю найденную в словаре строку, а во втором поле записан последний символ строки. Рассмотрим пример.

15.2.1.

Дана строка *abracadabraabracadabra*. Проиллюстрируем работу алгоритма с помощью таблицы.

	Читаемый символ	Метка	Словарь
1	<i>a</i>	$(0,a)$	1 - <i>a</i>
2	<i>b</i>	$(0,b)$	1 - <i>a</i> 2 - <i>b</i>
3	<i>r</i>	$(0,r)$	1 - <i>a</i> 2 - <i>b</i> 3 - <i>r</i>
4	<i>a</i>	$(1,c)$	1 - <i>a</i> 2 - <i>b</i> 3 - <i>r</i> 4 - <i>ac</i>
5	<i>a</i>	$(1,d)$	1 - <i>a</i> 2 - <i>b</i> 3 - <i>r</i> 4 - <i>ac</i> 5 - <i>ad</i>
6	<i>a</i>	$(1,b)$	1 - <i>a</i> 2 - <i>b</i> 3 - <i>r</i> 4 - <i>ac</i> 5 - <i>ad</i> 6 - <i>ab</i>
7	<i>r</i>	$(3,a)$	1 - <i>a</i> 2 - <i>b</i> 3 - <i>r</i> 4 - <i>ac</i> 5 - <i>ad</i> 6 - <i>ab</i> 7 - <i>ra</i>

	Читаемый символ	Метка	Словарь
8	a	$(6,r)$	1 - a 2 - b 3 - r 4 - ac 5 - ad 6 - ab 7 - ra 8 - abr
9	a	$(4,a)$	1 - a 2 - b 3 - r 4 - ac 5 - ad 6 - ab 7 - ra 8 - abr 9 - aca
10	d	$(0,d)$	1 - a 2 - b 3 - r 4 - ac 5 - ad 6 - ab 7 - ra 8 - abr 9 - aca 10 - d
11	a	$(8,a)$	1 - a 2 - b 3 - r 4 - ac 5 - ad 6 - ab 7 - ra 8 - abr 9 - aca 10 - d 11- $abra$

Получили код $(0,a)(0,b)(0,r)(1,c)(1,d)(1,b)(3,a)(6,r)(4,a)(0,d)(8,a)$. Каждая метка занимает $3+8 = 11$ бит. Наш код — $11 \times 11 = 121$ бит. Исходная строка — 176 бит. Коэффициент сжатия $k = \frac{176}{121} = 1.45$.

15.3. Упражнения

1. С помощью алгоритмов *LZ77* и *LZ78* закодируйте фразу «Интервьюер интервента интервьюировал». Вычислите степень сжатия.
2. С помощью алгоритмов *LZ77* и *LZ78* закодируйте фразу «Жутко жуку жить на суку» Вычислите степень сжатия.
3. С помощью алгоритмов *LZ77* и *LZ78* закодируйте фразу «Рыла свинья белорыла, тупорыла; полдвора рылом изрыла, вырыла, подрыла». Вычислите степень сжатия.
4. С помощью алгоритмов *LZ77* и *LZ78* закодируйте фразу «Скороговорун скороговорил скоровыговаривал, Что всех скороговорок не перескороговоришь не перескоровыговариваешь, Но, заскороговорившись, vysкороговорил, Что все скороговорки перескороговоришь, да не перескоровыговариваешь». Вычислите степень сжатия.
5. С помощью алгоритмов *LZ77* и *LZ78* закодируйте фразу «Жутко жуку жить на суку». Вычислите степень сжатия.
6. С помощью алгоритмов *LZ77* и *LZ78* закодируйте фразу «Жужжит жужелица, жужжит, да не кружится». Вычислите степень сжатия.
7. С помощью алгоритмов *LZ77* и *LZ78* закодируйте фразу «Ехал Грека через реку, видит Грека — в реке рак. Сунул Грека руку в реку, рак за руку Греку — цап!» Вычислите степень сжатия.
8. С помощью алгоритмов *LZ77* и *LZ78* закодируйте фразу «Турка курит трубку, курка клюет крупку. Не кури, турка, трубку; не клюй, курка, крупку» Вычислите степень сжатия.
9. С помощью алгоритмов *LZ77* и *LZ78* закодируйте фразу «Жили-были три китайца: Як, Як-цедрак, Як-цедрак-цедрак-цедрони. Жили-были три китайки: Цыпа, Цыпа-дрыпа, Цыпа-дрыпа-дрымпампони. Все они переженились: Як на Цыпе, Як-цедрак на Цыпе-дрыпе,

Як-цедрак-цедрак-цедрони на Цыпе-дрыпе-дрымпампони. И у них родились дети. У Яка с Цыпой — Шах, у Яка-цедрака с Цыпой-дрыпой — Шах-шарах, у Яка-цедрака-цедрака-цедрони с Цыпой-дрыпой-дрымпампони — Шах-шарах-шарах-широни». Вычислите степень сжатия.

10. Придумайте фразу, получите для нее код с помощью алгоритмов *LZ77* и *LZ78* и передайте соседу для декодирования. Сравните результат декодирования с оригиналом.

16. Преобразование Барроуза–Уиллера и RLE

Преобразование Барроуза – Уилера применяется в алгоритмах сжатия качественных данных. Как и многие другие применяемые в алгоритмах сжатия преобразования, преобразование Барроуза – Уилера предназначено для того, чтобы сделать сжатие данных входного блока более эффективным. Посредством перестановки элементов данное преобразование превращает входной блок данных со сложными зависимостями в блок, структуру которого моделировать гораздо легче, причем отображение происходит без потерь информации.

Этот метод появился сравнительно недавно. Впервые он был опубликован 10 мая 1994 г. Авторами метода являются Дэвид Уилер и Майк Барроуз. Причем первый из них придумал этот метод еще в 1983 г., когда работал в компании AT&T Bell Laboratories. Но, видимо, либо тогда не придавал значения своему открытию, либо посчитал чрезмерными требования к вычислительным ресурсам.

По имени авторов алгоритм был назван преобразованием Барроуза – Уилера (Burrows - Wheeler Transform, далее – BWT). Метод был объявлен компромиссным между быстрыми словарными алгоритмами, с одной стороны, и статистическими алгоритмами, дающими более сильное сжатие, но, малоприменимыми в то время на практике, с другой стороны. Благодаря таким свойствам описываемое преобразование стало довольно популярным и среди разработчиков архиваторов, и среди научных работников. Уже опубликовано более сотни статей на разных языках, посвященных этому методу, и написано столько же программ, его реализующих.

16.0.1.

Дана строка *abrakadabra*. Составим матрицу циклических перестановок нашей строки.

$$\begin{pmatrix} abrakadabra \\ brakadabaa \\ rakadabraab \\ akadabraabr \\ kadabraabra \\ adabraabrak \\ dabraabraka \\ abraabrakad \\ braabrakada \\ raabrakadab \\ aabrakadabr \end{pmatrix}$$

Отсортируем все строки в лексикографическом порядке.

$$\begin{pmatrix} 1 & aabrakadabr \\ 2 & abraabrakad \\ \mathbf{3} & \mathbf{abracadabra} \\ 4 & adabraabrak \\ 5 & akadabraabr \\ 6 & braabrakada \\ 7 & brakadabaa \\ 8 & dabraabraka \\ 9 & kadabraabra \\ 10 & raabrakadab \\ 11 & rakadabraab \end{pmatrix}$$

Запомним в этой матрице позицию исходной строки (позиция – 3).

Выписываем символы последнего столбца *rdakraaaaabb*. Результат преобразования Барроуза–Уилера $BWT(abrakadabra) = (rdakraaaaabb, 3)$. Полученную последовательность будем сжимать с помощью алгоритма RLE(Run Length Encoding). Суть алгоритма проста: последовательности одинаковых символов заменяются на метку, состоящую из самого символа и длины последовательности одинаковых подряд идущих символов.

Для нашей строки *rdakraaaaabb*, 3 код будет таким: *r1d1a1k1r1a4b2*, 3. Полученный код будет занимать 7×8 бит на символы, 7×2 на длины, 8 бит на разделитель и 2 на позицию исходной строки в матрице. Получа-

ем 70 бит. Исходная строка *abracadabra* занимает 176 бит. Коэффициент сжатия $k = \frac{176}{70} = 2.5$. Сжатие произошло за счет того, что у преобразованной последовательности были ряды подряд идущих одинаковых символов.

Осталось обсудить вопрос восстановления исходной строки по результату *BWT*, то есть понять, как действует обратное преобразование *BWT*. Как и прежде, рассмотрим пример. Пусть известен результат преобразования $BWT(abrakadabra) = (rdakraaaabb, 3)$. Будем восстанавливать исходную матрицу. Первый столбик восстанавливается просто, так как строки матрицы были лексикографически упорядочены, то есть первый столбик это упорядоченный лексикографически последний, известный нам.

$$\begin{pmatrix} 1 & a \dots r \\ 2 & a \dots d \\ 4 & a \dots k \\ 5 & a \dots r \\ 6 & b \dots a \\ 7 & b \dots a \\ 8 & d \dots a \\ 9 & k \dots a \\ 10 & r \dots b \\ 11 & r \dots b \end{pmatrix}$$

Строки матрицы были получены в результате циклического сдвига исходной строки, то есть, символы последнего и первого столбцов образуют друг с другом пары. И нам ничто не может помешать отсортировать эти пары, поскольку обязательно существуют такие строки в матрице, которые начинаются с этих пар. Таким образом, получим второй столбец.

$$\begin{pmatrix} 1 & aa \dots r \\ 2 & ab \dots d \\ 4 & ad \dots k \\ 5 & ak \dots r \\ 6 & br \dots a \\ 7 & br \dots a \\ 8 & da \dots a \\ 9 & ka \dots a \\ 10 & ra \dots b \\ 11 & ra \dots b \end{pmatrix}$$

Отсортированные пары вместе с символами последнего столбца составляют тройки.

$$\begin{pmatrix} 1 & aab \dots r \\ 2 & abr \dots d \\ \mathbf{3} & \mathbf{abr} \dots a \\ 4 & ada \dots k \\ 5 & aka \dots r \\ 6 & bra \dots a \\ 7 & bra \dots a \\ 8 & dab \dots a \\ 9 & kad \dots a \\ 10 & raa \dots b \\ 11 & rak \dots b \end{pmatrix}$$

Аналогично восстанавливается вся матрица и, соответственно, исходная строка.

16.1. Упражнения

1. Преобразуйте строку *ababab* с помощью *BWT* и получите код с помощью RLE. Вычислите коэффициент сжатия.
2. Преобразуйте строку *ababababab* с помощью *BWT* и закодируйте с помощью алгоритма Хаффмана. Вычислите коэффициент сжатия.
3. Преобразуйте строку *abcabcabc* с помощью *BWT* и получите код с помощью арифметического кодирования. Вычислите коэффициент сжатия.
4. Преобразуйте строку *abcdaaaaaa* с помощью *BWT* и получите код с помощью RLE. Вычислите коэффициент сжатия.
5. Преобразуйте строку *aaaaabbbbb* с помощью *BWT* и получите код с помощью RLE. Вычислите коэффициент сжатия.

17. Приложения

17.1. Лабораторные работы

Кодирование Хаффмана

1. Написать программу, которая на входе получает текстовый файл и сжимает его с помощью кодов Хаффмана.
2. Вычислить степень сжатия.
3. Написать декодер для кодирования Хаффмана.

Коды Фэно–Шеннона

1. Написать программу, которая на входе получает текстовый файл и сжимает его с помощью кодов Фэно–Шеннона.
2. Вычислить степень сжатия.
3. Написать декодер для кодирования Фэно–Шеннона.

Арифметическое кодирование

1. Написать программу, которая на входе получает текстовый файл и сжимает его с помощью арифметического кодирования.
2. Вычислить степень сжатия.
3. Написать декодер для арифметического кодирования.

Алгоритм RLE и преобразование Барроуза–Уилера

1. Написать программу, которая на входе получает текстовый файл, преобразует его с помощью преобразования Барроуза–Уилера и сжимает с помощью алгоритма RLE.
2. Вычислить степень сжатия.
3. Написать декодер для RLE.

Словарный метод сжатия LZ77

1. Написать программу, которая на входе получает текстовый файл и сжимает его с помощью LZ77.
2. Вычислить степень сжатия.
3. Написать декодер для LZ77.

Словарный метод сжатия LZ78

1. Написать программу, которая на входе получает текстовый файл и сжимает его с помощью LZ78.
2. Вычислить степень сжатия.
3. Написать декодер для LZ78.

Код Хэмминга

1. Написать программу, которая шифрует двоичное сообщение с помощью кода Хэмминга, далее случайно делается ошибка в коде, а программа исправляет эту ошибку.

Нахождение примитивного элемента в конечном поле

1. Написать программу, которая выводит список примитивных элементов конечного поля ($p = 2, p = 3, n = 3, n = 4$).

Линейный код (5,2)

1. Написать программу, которая реализует линейный код типа (5,2).

Список литературы

- [1] **Белоногов, В. А.** Теория кодирования : учебное пособие / В. А. Белоногов. – Екатеринбург : УГТУ – УПИ, 2009. – 161 с.
- [2] **Биркгоф, Г.** Современная прикладная алгебра / Г. Биркгоф, Т. Бартти. – М. : Мир, 1976. – 400 с.
- [3] **Берлекэмп, Э.** Алгебраическая теория кодирования / Э. Берклекэмп. – М. : Мир, 1971. – 480 с.
- [4] **Блейхут, Р.** Теория и практика кодов, контролирующих ошибки / Р. Блейхут. – М. : Мир, 1986. – 576 с.
- [5] **Аршинов, М. Н.** Коды и математика / М. Н. Аршинов, Л. Е. Садовский. – М. : Наука, 1983. – 144 с.
- [6] **Краснов, М. В.** Теория кодирования : метод. указания / сост. М. В. Краснов ; Яросл. гос. ун-т. им. П. Г. Демидова. – Ярославль : ЯрГУ, 2006. – 50 с.
- [7] **Мак-Вильямс, Ф.** Теория кодов, исправляющих ошибки / Ф. Мак-Вильямс, Н. Дж Слоэн. – М. : Связь, 1979. – 746 с.
- [8] **Вентцель, Е. С.** Теория вероятностей / Е. С. Вентцель. – М. : Наука, 1969. – 576 с.
- [9] **Лидовский, В. В.** Теория информации : учебное пособие / В. В. Лидовский. – М. : Компания Спутник+, 2004. – 111 с.
- [10] **Сэломон, Д.** Сжатие данных, изображений и звука / Д. Сэломон. – М. : Техносфера, 2004. – 368 с.
- [11] **Сидельников, В. М.** Теория кодирования / В. М. Сидельников. – М. : Физматлит, 2008. – 322 с.
- [12] **Ватолин, Д.** Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. – М. : ДИАЛОГ-МИФИ, 2003. – 384 с.

Учебное издание

Казарин Лев Сергеевич
Заводчиков Михаил Александрович

**Элементы теории кодирования,
сжатия и восстановления информации**
Учебно-методическое пособие

Редактор, корректор Л. Н. Селиванова
Компьютерный набор, верстка М. А. Заводчиков

Подписано в печать 25.03.2020. Формат 60 × 84 1/16
Усл.-печ. л. 6,4. Уч.-изд. л. 5,5. Тираж 4 экз.

Оригинал-макет подготовлен
в редакционно-издательском отделе ЯрГУ

Ярославский государственный университет им. П. Г. Демидова.
1500003, Ярославль, ул. Советская, 14.