Владимирский государственный университет

М. С. БЕСПАЛОВ В. Д. БУРКОВ

ЭЛЕМЕНТЫ ТЕОРИИ КОДИРОВАНИЯ

Учебное пособие

Владимир 2025

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

М. С. БЕСПАЛОВ В. Д. БУРКОВ

ЭЛЕМЕНТЫ ТЕОРИИ КОДИРОВАНИЯ

Учебное пособие

Электронное издание

Владимир 2025

ISBN 978-5-9984-2009-2

УДК 51
ББК 22.1

Авторы: М. С. Беспалов (гл. 1, 3, 4), В. Д. Бурков (гл. 2)

Рецензенты:
Кандидат физико-математических наук
доцент кафедры высшей математики
Московского физико-технического института
(национального исследовательского университета)
И. В. Каржеманов

Доктор технических наук, профессор
зав. кафедрой информатики и защиты информации
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
М. Ю. Монахов

Издаётся по решению редакционно-издательского совета ВлГУ

Беспалов, М. С. Элементы теории кодирования [Электронный ресурс] : учеб. пособие / М. С. Беспалов, В. Д. Бурков ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2025. – 76 с. – ISBN 978-5-9984-2009-2. – Электрон. дан. (0,98 Мб). – 1 электрон. опт. диск (CD-ROM). – Систем. требования: Intel от 1,3 ГГц ; Windows XP/7/8/10 ; Adobe Reader ; дисковод CD-ROM. – Загл. с титул. экрана.

Приводится теоретический и практический материал по следующим разделам теории кодирования: арифметическое кодирование, алфавитное кодирование, коды с исправлением ошибок и элементы криптографии.

Предназначено для студентов вузов направлений подготовки 02.03.01 и 02.04.01 «Математика и компьютерные науки» всех форм обучения, а также для студентов и аспирантов математических, информационных и технических специальностей.

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС ВО.

Ил. 9. Табл. 7. Библиогр.: 11 назв.

ISBN 978-5-9984-2009-2

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Теория кодирования – предмет изучения современной дискретной математики и информатики. Большинство учебных пособий по теории кодирования авторы строят как введение в выбранное направление с подробным описанием самого направления. В данном учебном пособии был выбран другой подход. Авторы знакомят студентов с основными базовыми понятиями теории кодирования.

Теорию кодирования изучают на старших курсах бакалавриата или в магистратуре, так как для ее восприятия нужны знания дискретной математики и информатики.

В первой главе излагаются три отдельных способа арифметического кодирования, которые представляют самостоятельный интерес и в дальнейших главах не затрагиваются. Общие принципы кодирования встречаются и в этих конструкциях. Отметим, что в настоящее время в некоторых источниках фибоначчиева система счисления описана некорректно, с нарушением принципов декодирования.

Вторая глава – основная вводная глава для последующего материала. Для наглядности в главе используется графическое представление кода. Рассматриваются коды Фано, Шеннона и Хаффмана. Последний служит иллюстрацией принципов оптимального кодирования.

Третья глава служит продолжением предыдущей главы. Однако для кодов, обнаруживающих и исправляющих ошибки, характерна избыточность, что противоречит требованию оптимальности кода.

В четвертой главе рассмотрен один конкретный криптографический алгоритм. Для его описания потребовались начальные сведения из теории чисел.

Глава 1

Арифметическое кодирование

Кодирование – представление исходных сообщений, которые являются словами A_j в некотором алфавите $A = \{a_1, a_2 \dots a_n\}$, в виде слов B_k в другом алфавите $B = \{b_1, b_2 \dots b_m\}$ (в некоторых случаях в том же). Поскольку процесс кодирования проводится с привлечением компьютера, то остановимся только на варианте двоичного кодирования, когда берется выходной алфавит $B = \{0, 1\}$.

Рассмотрим *арифметическое кодирование*, заключающееся в представлении неотрицательных целых чисел в алфавите $\{0, 1\}$. Простейший вариант кодирования – запись чисел в двоичной системе счисления. Недостаток этого метода кодирования – необходимость переходить в трехсимвольный алфавит на выходе, добавляя символ разделительного знака между *кодовыми словами*. Это обусловлено тем, что схема кодирования должна быть *разделимой*, то есть при *декодировании* (восстановлении исходного сообщения) сначала отделяем кодовые слова, по которым однозначно восстанавливается исходное сообщение.

По аналогии с предложенными далее вариантами двоичного кодирования можно построить существенно более удобные, чем рассматриваемый, методы троичного кодирования. Существует способ избежать добавления разделительного знака (то есть перехода к троичному кодированию), применяя вариант равномерного двоичного кодирования. В этом случае все кодовые слова будут иметь одинаковую длину. Этот вариант влечет *избыточность кода*, состоящую в необходимости рассмотрения кодовых слов большой длины, начинающихся с длинной цепочки нулей, не несущей информации о исходном сообщении.

Поэтому будут предложены двоичные разделимые коды без явно выделенных разделительных знаков, позволяющие организовать процедуру однозначного декодирования (включающую расстановку разделительных знаков по некоторому правилу) и применимые для исходных слов сколь угодно большой длины.

1.1. Последовательность Фибоначчи и ее производящая функция

Последовательность чисел $\{f_n\}_{n=0}^{\infty}$, заданная рекуррентным соотношением

$$f_n = f_{n-1} + f_{n-2} \tag{1.1}$$

и начальным условием

$$f_0 = 0, \quad f_1 = 1,$$

называется *последовательностью Фибоначчи* (табл. 1).

Таблица 1

Таблица начальных чисел Фибоначчи

Номер	0	1	2	3	4	5	6	7	8	9	10	11	12
Число	0	1	1	2	3	5	8	13	21	34	55	89	144

Вычислим производящую функцию $\varphi(t) = \sum_{n=0}^{\infty} f_n t^n$ для этой последовательности, которая представляет собой формальный степенной ряд по новой переменной t

$$\varphi(t) = f_0 + f_1 t + \sum_{n=2}^{\infty} f_n t^n = f_0 + f_1 t + \sum_{n=2}^{\infty} f_{n-1} t^n + \sum_{n=2}^{\infty} f_{n-2} t^n =$$

$$= f_0 + f_1 t + t \sum_{n=1}^{\infty} f_n t^n + t^2 \sum_{n=0}^{\infty} f_n t^n = 0 + t + t\varphi(t) + t^2 \varphi(t).$$

Отсюда $\varphi(t) - t\varphi(t) - t^2\varphi(t) = t$ и

$$\varphi(t) = \frac{t}{1 - t - t^2}.$$

Корни знаменателя $t_{1,2} = \frac{-1 \pm \sqrt{5}}{2}$. Возьмем корни с противоположными знаками и обозначим $\alpha = \frac{1+\sqrt{5}}{2}$, $\beta = \frac{1-\sqrt{5}}{2}$.

Производящую функцию можно переписать в виде

$$\varphi(t) = \frac{1}{\sqrt{5}} \left(\frac{1}{1 - \alpha t} - \frac{1}{1 - \beta t} \right).$$

Отсюда получим ее представление в виде ряда

$$\varphi(t) = \frac{1}{\sqrt{5}} \sum_{k=0}^{\infty} (\alpha^k - \beta^k) t^k$$

и явную формулу для чисел Фибоначчи, известную как *формула Бине*

$$f_k = \frac{1}{\sqrt{5}} (\alpha^k - \beta^k).$$

Лемма 1.1. *Скорость роста чисел Фибоначчи экспоненциальная с основанием* $\alpha \approx 1,6$

$$\lim_{n \to \infty} \frac{f_n}{f_{n-1}} = \alpha.$$

Доказательство. Так как $\alpha > |\beta|$, то

$$\lim_{n \to \infty} \frac{f_n}{f_{n-1}} = \lim_{n \to \infty} \frac{\alpha^n - \beta^n}{\alpha^{n-1} - \beta^{n-1}} = \lim_{n \to \infty} \frac{\alpha - \beta(\frac{\beta}{\alpha})^{n-1}}{1 - (\frac{\beta}{\alpha})^{n-1}} = \alpha.$$

1.2. Фибоначчиева система счисления

Предложим разделимую систему кодирования натуральных чисел в двоичном алфавите. Для этого изучим отдельные свойства чисел Фибоначчи.

Лемма 1.2. *При* $k \geq 2$ *имеем*

$$f_{2k} = f_1 + f_3 + \ldots + f_{2k-3} + f_{2k-1}, \tag{1.2}$$

$$f_{2k+1} = 1 + f_0 + f_2 + \ldots + f_{2k-2} + f_{2k}. \tag{1.3}$$

Доказательство. Для формулы (1.2) база индукции $f_4 = f_1 + f_3$.
Индуктивное предположение для формулы (1.2) имеет вид

$$f_{2k-2} = f_1 + f_3 + \ldots + f_{2k-3}.$$

Отсюда по основной формуле (1.1) имеем

$$f_{2k} = f_{2k-2} + f_{2k-1} = f_1 + f_3 + \ldots + f_{2k-3} + f_{2k-1}.$$

Для формулы (1.3) база индукции $f_3 = 1 + f_0 + f_2$.
Индуктивное предположение для формулы (1.3) имеет вид

$$f_{2k-1} = 1 + f_0 + f_2 + \ldots + f_{2k-2}.$$

Отсюда по основной формуле (1.1) имеем

$$f_{2k+1} = f_{2k-1} + f_{2k} = 1 + f_0 + f_2 + \ldots + f_{2k-2} + f_{2k}.$$

Следствие 1.1. *Если натуральное число n представлено в виде*

$$n = f_{i_0} + f_{i_1} + \ldots + f_{i_s}, \text{ где } i_0 = 0, i_k - i_{k-1} > 1, \qquad (1.4)$$

то $n < f_{i_s+1}$.

Доказательство. Сравним формулу (1.4) с формулами (1.2) или (1.3), в которых номера чисел Фибоначчи (слагаемых) идут с шагом 2, а в формуле (1.4) этот шаг не меньше, чем 2. Если старший номер в (1.4), равный i_s, нечетный, то полагаем $i_s = 2k - 1$ при сравнении с (1.2). Если старший номер в (1.4) четный, то полагаем $i_s = 2k$ при сравнении с (1.3).

Сравниваем соответствующие слагаемые по убыванию (справа налево). Как только в формуле (1.4) шаг окажется больше 2, то соответствующее слагаемое в (1.4) и все предшествующие (с меньшими номерами) будут меньше, чем в формуле (1.2) или (1.3). Число слагаемых ($f_{i_0} = 0$ опускаем) в (1.4) не может быть больше, чем в (1.2) или (1.3). Поэтому $n < f_{i_s+1}$. □

Рекомендуется отдельно рассмотреть крайний случай (1.4), когда шаг всегда равен 2.

Следствие 1.2. *При всех $n \geq 0$ имеем*

$$f_{n+2} = 1 + \sum_{k=0}^{n} f_n.$$

Доказательство. Сложить формулы (1.2) и (1.3) для f_{2k} и f_{2k+1}, а также отдельно формулы для f_{2k-1} и f_{2k}.

Теорема 1.1 (Цекендорфа). *Любое натуральное число n однозначно представимо в виде (1.4).*

Доказательство. Сначала докажем существование представления (1.4) от противного. Если не для всех чисел есть такое представление, то выберем в качестве n минимальное из натуральных чисел, для которого представление (1.4) невозможно. Обозначим $j(n)$ максимальный номер числа Фибоначчи, не превосходящего n,

$$f_{j(n)} < n < f_{j(n)+1}. \tag{1.5}$$

При этом частный случай $n = f_{j(n)}$ неприменим, так как существует требуемое представление $n = f_0 + f_{j(n)}$.

Рассмотрим $s = n - f_{j(n)}$. Если вычтем $f_{j(n)}$ из всех частей неравенства (1.5), то получим $0 < s < f_{j(n)-1}$. Поскольку $s < n$, то для s представление типа (1.4) есть, в котором согласно следствию 1.1 старшее слагаемое этого представления не больше $f_{j(n)-2}$. Если добавим к этому представлению $f_{j(n)}$, то получим представление (1.4) для n. Получили противоречие с предположением.

Докажем единственность представления (1.4). Пусть n – минимальное из чисел, для которого возможны два различных представления:

$$n = f_{i_0} + f_{i_1} + \ldots + f_{i_s}, \quad n = f_{j_0} + f_{j_1} + \ldots + f_{j_s}.$$

Если старшие слагаемые различны, например $j_s > i_s$, то согласно следствию 1.1 они представляют собой разные числа

$$n = f_{j_0} + f_{j_1} + \ldots + f_{j_s} \geq f_{j_s} > f_{i_0} + f_{i_1} + \ldots + f_{i_s} = n.$$

Остался случай $j_s = i_s$, который приводит к противоречию с утверждением о минимальности n при рассмотрении $m = n - f_{j_s}$. $\square$

Формулу (1.4) можно переписать в виде линейной комбинации с коэффициентами 0 и 1 всех возможных чисел Фибоначчи в порядке убывания индексов

$$n = n_{j_s} \cdot f_{j_s} + n_{j_s-1} \cdot f_{j_s-1} + n_{j_s-2} \cdot f_{j_s-2} + \ldots + n_1 \cdot f_1 + n_0 \cdot f_0. \tag{1.6}$$

Набор $n_{j_s} n_{j_s-1} n_{j_s-2} \ldots n_1 n_0$ будет фибоначчиевым кодом числа s.

Определение 1.1. Представлением любого целого неотрицательного числа в *фибоначчиевой системе счисления* называется его запись в алфавите $\{0, 1\}$ в виде набора, начинающегося и заканчивающегося единицей и не имеющего в записи двух рядом стоящих единиц, в соответствии с разложением (1.4), где единица на j-м разряде означает наличие слагаемого f_j в представлении (1.4), а нуль — его отсутствие.

В табл. 2 приведем представления начальных чисел множества $\mathbb{N}_0$ в фибоначчиевой системе (где младшие разряды справа).

Таблица 2

Числа в фибоначчиевой системе счисления

0	1	2	3	4	5	6	7	8
1	101	1001	10001	10101	100001	100101	101001	1000001

Фибоначчиеву систему счисления удобно использовать для записи нескольких целых неотрицательных чисел без каких-либо ограничений их значения. В этом случае мы заранее не знаем величину разрядной сетки для представления всех чисел в двоичной системе счисления. Возможен и другой случай, когда величина разрядной сетки большая, а большинство чисел в рассматриваемом наборе маленькие.

В электронных источниках в целях сокращения числа бит для представления одного числа опускают слагаемое $f_0 = 0$ и тем самым отказываются от требования **разделимости** схемы кодирования. Такой подход не позволяет расставлять разделительные знаки между числами и не подходит для кодирования набора из нескольких чисел, а также исключает представление нуля в схеме кодирования.

Для оценки длины кодового сообщения заметим, что разрядная сетка для представления чисел увеличивается на единицу на очередном числе Фибоначчи. Числа от f_k до $f_{k+1} - 1$ представляются $k + 1$ битом. Скорость роста разрядной сетки по лемме 1.1 равна $\alpha \approx 1,6$.

Для разложения произвольного числа n в фибоначчиевой системе счисления используется «жадный» алгоритм выделения на каждом шаге ближайшего снизу числа Фибоначчи, действие которого проиллюстрируем примером.

Пример 1.1. *Разложить число 105 в фибоначчиевой системе.*

Найдем числа Фибоначчи, между которыми лежит наше число,

$$f_{11} = 89 < 105 < f_{12}.$$

Значит, $105 = 89 + 16 = f_{11} + 16$. Повторим вычисления для числа 16

$$f_7 = 13 < 16 < f_8.$$

Значит, $105 = 89 + 13 + 3 = f_{11} + f_7 + 3 = f_{11} + f_7 + f_4 + f_0$.

Итак, получили для числа 105 код 100010010001.

Представление чисел в фибоначчиевой системе счисления позволяет их однозначно декодировать, что проиллюстрируем примером. Разделительный знак между числами ставим там, где расположены две или более единицы подряд.

Пример 1.2. *Декодировать сообщение в виде набора чисел в фибоначчиевой системе* $11011001100011100000100101$.

Сначала алгоритм выясняет число чисел набора расстановкой разделительных знаков между двумя единицами

$$1, 101, 1001, 10001, 1, 10000100101.$$

Для каждого числа проводим декодированием по табл. 2, а для больших чисел – по табл. 1: так, для последнего числа получаем $10000100101_{fib} = f_{10} + f_5 + f_2 + f_0 = 55 + 5 + 1 = 61$.
Ответ. $0, 1, 2, 3, 0, 61$.

1.3. Линейная форма Фибоначчи

Другой способ разделимого кодирования больших чисел – *линейная форма Фибоначчи*, которая базируется на следующей лемме.

Лемма 1.3. *Любые два соседних числа Фибоначчи взаимно просты.*

Доказательство. Если предположить, что числа f_{n+1} и f_n делятся на c, то и числа f_{n-1} и f_n делятся на c

$$f_{n+1} = ac, \quad f_n = bc \quad \Rightarrow f_{n-1} = (a - b)c.$$

Поскольку f_1 и f_2 взаимно просты, то и f_{n+1}, f_n взаимно просты.

Определение 1.2. Представление натурального числа n в виде

$$n = af_j + bf_{j+1}, \quad \text{где } a, b \in \mathbb{N}_0, \tag{1.7}$$

называют линейной формой Фибоначчи. Форма (1.7) называется *канонической*, если $0 \le b < a < f_{j+1}$.

Теорема 1.2. *Для каждого натурального n каноническая форма (1.7) единственна.*

Доказательство. Предположим, что для n существуют две канонические формы: форма (1.7) и $n = cf_k + df_{k+1}$, где $0 \le d < c < f_{k+1}$.

Если $k > j$, то, применяя (1.1), приведем к форме разложения $n = cf_k + df_{k+1} = df_{k-1} + (c+d)f_k = \ \ldots = Af_j + Bf_{j+1}$, где $A < B$. Равенство $af_j + bf_{j+1} = Af_j + Bf_{j+1}$ перепишем в виде

$$(a - A)f_j = (B - b)f_{j+1}, \quad \text{где } a - A \geq 0, B - b \geq 0.$$

Если бы было $a - A < 0$, то неравенство $b < a < A < B$ привело бы к противоречию.

С одной стороны, $a - A$ делится на f_{j+1} по лемме 1.3, а с другой стороны, $a - A < f_{j+1}$. Отсюда $a - A = 0$.

Случай $k = j$ проверяется аналогично. $\square$

Замена натурального числа n на тройку чисел (j, a, b) из канонического разложения (1.7) соответствует кодированию *линейным кодом Фибоначчи.*

Алгоритм разложения n в каноническую линейную форму Фибоначчи состоит из итераций *основного шага*:

1) от представления $af_k + bf_{k+1}$ при $a \leq b$ по формуле (1.1) переходим к $(b - a)f_{k+1} + af_{k+2}$.

Если окажется $a > b$ для $n = af_k + bf_{k+1}$, то проверяем второе условие $a < f_{k+1}$, выполнение которого означает окончание алгоритма. При невыполнении второго условия делается *поправочный шаг*:

2) от $af_k + bf_{k+1}$ переходим к $a_1 f_k + b_1 f_{k+1}$ с новыми коэффициентами $a_1 = a - f_{k+1}$, $b_1 = b + f_k$.

Пример 1.3. *Разложить число 20 в канонической линейной форме Фибоначчи.*

Действие алгоритма стартует с начального представления, которое можно брать произвольно с условием $a < b$. Например, представим число 20 в виде $20 = 6 + 14 = 6f_1 + 14f_2$.

Применим основной шаг

$$6f_1 + 14f_2 \Rightarrow 8f_2 + 6f_3.$$

Поскольку $a > b$ $(8 > 6)$, то проверим второе условие $a < f_{k+1}$ $(8 <? 2)$, которое не выполняется. Значит, мы обязаны сделать поправочный шаг $(f_2 = 1,\ f_3 = 2)$

$$8f_2 + 6f_3 \Rightarrow (8 - 2)f_2 + (6 + 1)f_3 = 6f_2 + 7f_3.$$

Применим основной шаг

$$6f_2 + 7f_3 \Rightarrow 1f_3 + 6f_4 \Rightarrow 5f_4 + 1f_5.$$

Применим поправочный шаг

$$5f_4 + 1f_5 \Rightarrow (5-5)f_4 + (1+3)f_5 = 4f_5 \Rightarrow 4f_5 + 0f_6.$$

Всю цепочку переходов можно оформить так (разные значки для основного и поправочного шагов):

$$20 = 6f_1 + 14f_2 = 8f_2 + 6f_3 \Rightarrow 6f_2 + 7f_3 = 1f_3 + 6f_4 = 5f_4 + 1f_5 \Rightarrow 4f_5.$$

Дополним ее тривиальным переходом $4f_5 \to 4f_5 + 0f_6$.

Ответ. $20 = 4f_5 + 0f_6$.

Начальное разложение $20 = 6f_1 + 14f_2$ в приведенном примере не оптимальное. Если начать алгоритм с разложения $20 = 8f_1 + 12f_2$, то поправочные шаги не встретятся

$$20 = 8f_1 + 12f_2 = 4f_2 + 8f_3 = 4f_3 + 4f_4 = 4f_5 \to 4f_5 + 0f_6.$$

Оптимальное начальное разложение строится на основе следующей леммы.

Лемма 1.4 о золотом сечении. *Если весь отрезок поделен в отношении $\alpha : 1$, где $\alpha = \frac{1+\sqrt{5}}{2}$, то длина всего отрезка относится к длине большей части так же, как длина большей части относится к длине меньшей части. Это отношение сохраняется при подобных смещениях как в сторону увеличения, так и в сторону уменьшения длин.*

Доказательство. Будем считать $\alpha > 1$ неизвестным и составим требуемое соотношение

$$\frac{1+\alpha}{\alpha} = \frac{\alpha}{1},$$

приводящее к уравнению $\alpha^2 - \alpha - 1 = 0$, положительный корень которого $\alpha = \frac{1+\sqrt{5}}{2}$ совпадает с ранее полученным α.

Подобное смещение состоит в том, что теперь большую часть α примем за длину всего отрезка, а меньшую часть, равную 1, примем за новую большую часть. Тогда меньшая часть нового отрезка вычисляется как $\alpha - 1$. Аналогичное соотношение примет вид $\frac{\alpha}{1} = \frac{1}{\alpha-1}$, приводящий к тому же квадратному уравнению.

Проверьте, что подобное смещение можно снова повторять, уравнение при этом сохранится.

Следствие 1.3. *Если в разложении (1.7) убрать требование целочисленности, взять в качестве начального разложения золотое сечение*

$$n = \frac{n}{\alpha+1} f_1 + \frac{\alpha n}{\alpha+1} f_2,$$

то основной шаг алгоритма можно применять бесконечно долго (без перехода на поправочный шаг).

Если в разложении (1.7) целое a составляет примерно 38 процентов от $a+b$, то в алгоритме канонического разложения можно ожидать минимум поправочных шагов (возможно, что все шаги основные).

Дополнение к алгоритму, минимизирующее число поправочных шагов.

Метод поиска начального разложения через фибоначчиеву систему счисления (1.6)

$$n = n_j \cdot f_j + n_{j-1} \cdot f_{j-1} + n_{j-2} \cdot f_{j-2} + \ldots + n_1 \cdot f_1 + n_0 \cdot f_0,$$

где $n_k \in \{0, 1\}$ и двух рядом стоящих единиц нет. В качестве слагаемого b возьмем число, полученное сдвигом всех разрядов на один вправо $b = n_j \cdot f_{j-1} + n_{j-1} \cdot f_{j-2} + n_{j-2} \cdot f_{j-3} + \ldots + n_1 \cdot f_0$, а в качестве слагаемого a возьмем число, полученное еще одним сдвигом всех разрядов на один вправо $a = n_j \cdot f_{j-2} + n_{j-1} \cdot f_{j-3} + \ldots + n_2 \cdot f_0$. Первоначальное разложение $n = a + b = af_1 + bf_2$.

Линейная форма Фибоначчи превращается в разделимую схему кодирования при сочетании двух рассмотренных систем кодирования, когда числа этой тройки (j, a, b) из канонического разложения (1.7) могут быть представлены в фибоначчиевой системе счисления, что и продемонстрируем следующим примером.

Пример 1.4. *Разложение числа 20 в смешанной системе счисления (линейная форма Фибоначчи, а потом фибоначчиева система счисления).*

Три числа $j = 5$, $a = 4$ и $b = 0$ из канонического представления линейной формы Фибоначчи числа 20 закодируем в фибоначчиевой системе счисления (см. табл. 2) и запишем одним словом

100001101011.

1.4. Код Левенштейна

При записи натурального числа в двоичной системе счисления сначала определяют количество разрядов, тогда старший разряд заполняется единицей. Значит, если нам известно количество разрядов, то старшую единицу можно (для экономии места) не писать. Эта идея реализуется в коде Левенштейна, где сначала вычисляется количество разрядов, а потом на этих разрядах указывается число. При реализации алгоритма кодирования необходимо помнить о требовании *разделимости* схемы кодирования.

Поскольку количество разрядов числа может быть большим, то один и тот же прием для кодирования числа многократно повторяется для кодирования разрядов. Поэтому сначала должны закодировать количество повторов, которое назовем количеством уровней.

В коде Левенштейна количество уровней (или количество шагов) кодируется набором из единиц: 10 – один уровень, 110 – два уровня, 1110 – три уровня и т. д. Заканчивается этот код количества уровней нулем, который является также разделительным знаком. Отметим, что эта запись количества уровней тоже считается одним (первым) уровнем.

Основная идея разделимости кода состоит в том, что количество разрядов (бит) каждого уровня записывается в предыдущем уровне; это позволяет расставить разделительные знаки.

Кодирование осуществляется через запись числа в двоичной системе счисления. Поэтому введем обозначения: $L(M)$ – код Левенштейна числа M, представленного в десятичной системе; $l(M)$ – код Левенштейна числа M, представленного в двоичной системе; $+\!+$ используем как знак присоединения, играющий роль временного разделительного знака. При необходимости конкретизировать вид системы для представления числа будем использовать нижний индекс; например $21_{10} = 10101_2$. Поэтому очевидно, что $L(M_{10}) = l(M_2)$, то есть коды Левенштейна одного и того же числа, записанного в разных системах, совпадают. Поскольку индексы очевидны, то их опускаем, например $L(21) = l(10101)$.

Шаг алгоритма следующий: аргумент M для операции l разобьем на две составляющих $1{+}{+}M'$, отделив один знак (который всегда ра-

вен 1), например $L(21) = l(10101) = l(1{+}{+}0101)$. Обозначим десятичным числом N количество разрядов второго слагаемого M' данного присоединения $M = 1{+}{+}M'$ в двоичной системе счисления.

Основная формула алгоритма кодирования

$$L(M) = l(M) = 1{+}{+}L(N){+}{+}M'.$$

Проводится многократная итерация этой формулы.

Разберем простейшие частные случаи:

$$L(0) = l(0) = 0,$$

поскольку нет первого и второго слагаемых присоединения;

$$L(1) = l(1) = l(1{+}{+}) = 1{+}{+}L(0) = 1{+}{+}0 = 10,$$

поскольку количество разрядов второго слагаемого равно нулю.

Пример 1.5. *Разложение числа 21 в код Левенштейна.*

$$L(21) = l(10101) = 1{+}{+}L(4){+}{+}0101 = 1{+}{+}l(100){+}{+}0101 =$$

$$= 1{+}{+}1{+}{+}L(2){+}{+}00{+}{+}0101 = 1{+}{+}1{+}{+}l(10){+}{+}00{+}{+}0101 =$$

$$= 1{+}{+}1{+}{+}1{+}{+}L(1){+}{+}0{+}{+}00{+}{+}0101 =$$

$$= 1{+}{+}1{+}{+}1{+}{+}10{+}{+}0{+}{+}00{+}{+}0101 = 111100000101.$$

Как только процесс вычисления оборвался, мы убрали все знаки $+{+}$.

В табл. 3 указаны коды отдельных чисел и выделены уровни. Код записывается сплошным числом без разделения на уровни.

Поскольку указания на разрядность второго уровня у нас нет, то для него отводится один разряд, а число предполагается двухразрядное. На каждом уровне (начиная со второго) указано количество разрядов следующего уровня (с опущенной начальной единицей).

Пример 1.6. *Закодировать набор из трех чисел 32, 0, 5.*

Кодируется словом, составленным из слов табл. 3

11110001000000111000 1

Декодирование слова из примера 1.6 проводится по табл. 3. Код Левенштейна служит примером *префиксного* кода, при котором ни одно кодовое слово не является началом другого.

Таблица 3

Таблица кодов Левенштейна отдельных чисел

Число	Код			Число	Код			
0	0			15	1110	1	111	
1	10			16	11110	0	00	0000
2	110	0		17	11110	0	00	0001
3	110	1		20	11110	0	00	0100
4	1110	0	00	24	11110	0	00	1000
5	1110	0	01	31	11110	0	00	1111
6	1110	0	10	32	11110	0	01	00000
7	1110	0	11	63	11110	0	01	11111
8	1110	1	000	64	11110	0	10	000000
9	1110	1	001	100	11110	0	10	100110
10	1110	1	010	127	11110	0	10	111111
11	1110	1	011	128	11110	0	11	0000000
12	1110	1	100	200	11110	0	11	1001100
13	1110	1	101	256	11110	1	000	00000000
14	1110	1	110	512	11110	1	001	000000000

Алгоритм декодирования кода Левенштейна.

1. Считываем и отделяем часть кода слева направо до первого нуля, вычисляя количество шагов C. Если сразу увидели 0, то это одновременно количество шагов и код первого числа $M = 0$. Отделяем это число $M = 0$ от кода и возвращаемся в начало алгоритма. Другой особый случай – когда закончилась считываемая информация, то происходит остановка и печать ответа. Иначе C равно количеству единиц до первого нуля.

2. Полагаем $M := 1$ (показатель количества разрядов), $C := C - 1$ (так как считывание количества шагов считаем за выполненный шаг).

3. Если $C > 0$, то: а) формируем число M', помещая в ячейку M' из кода очередные M бит информации;

б) присоединяем в начало двоичного числа M' единицу, формируя новое число $M_2 := 1{+}{+}M'$;

в) переводим двоичное число M_2 в десятичное M;

г) $C := C - 1$ и повторяем пункт 3.

Если $C = 0$, то M выводим как ответ и возвращаемся к пункту 1.

Пример 1.7. *Декодировать сообщение* 111100010000001110001.

Шаг А.

1. Формируем счетчик шагов $C = 4$, отделяя от кода первые пять символов (до первого нуля) 11110.

2. $M := 1$ и $C = 3$.

3. От оставшейся части кода $K = 0010000001110001$ отделяем один (так как $M = 1$) разряд в виде числа 0 и добавляем к нему 1 в начало (счетчик шагов при этом уменьшаем на единицу $C = 2$). Сформированное двоичное число 10 переводим в десятичную систему $M = 2$ – указатель количества разрядов для следующего шага. В символьных обозначениях этот шаг опишем так:

$$C = 3, M = 1 \Rightarrow M' = 0 \Rightarrow M = 10 \to 2,\ C = 2,\ \ K = 010000001110001.$$

Повтор пункта 3

$$C = 2, M = 2 \Rightarrow M' = 01 \Rightarrow M = 101 \to 5,\ C = 1,\ \ K = 0000001110001.$$

Повтор пункта 3

$$C = 1, M = 5 \Rightarrow M' = 00000 \Rightarrow M = 100000 \to 32,\ C = 0,\ K = 01110001.$$

Поскольку $C = 0$, то число 32 пересылаем в ответ в следующем виде (*Ответ.* 32) и возвращаемся к пункту 1 для оставшейся части кода $K = 01110001$.

Шаг Б.

1. Отделяем от кода один символ $M = 0$, который добавляем в ответ (*Ответ.* 32, 0) и возвращаемся к пункту 1 для $K = 1110001$.

Шаг В.

1. Формируем счетчик шагов $C = 3$, отделяя от кода первые четыре символа (до первого нуля) 1110.

2. $M := 1$ и $C = 2$.

3. От оставшейся части кода $K = 001$ отделяем один разряд (так как $M = 1$) в виде числа 0 и добавляем к нему 1 в начало, вычисляя $M = 2$. Меняем $C := 1$. Поскольку от кода остались тоже два символа, то они и дают последнее число 101, которое переводим в десятичную систему $M = 5$ и формируем окончательный ответ.

Ответ. 32, 0, 5.

Пример 1.8. *Закодировать 2025 кодом Левенштейна.*
Поскольку $2025 = 1024 + 512 + 256 + 128 + 64 + 32 + 8 + 1$, то

$$L(2025) = l(11111101001) = 1{+}{+}L(10){+}{+}1111101001,$$

$$\text{где} \quad L(10) = l(1010) = 1{+}{+}L(3){+}{+}010,$$

$$L(3) = l(11) = 1{+}{+}L(1){+}{+}1 = 1{+}{+}10{+}{+}1.$$

Итак, $L(2025) = 1{+}{+}1{+}{+}1{+}{+}10{+}{+}1{+}{+}010{+}{+}1111101001$

$$\text{или} \quad L(2025) = 1111010101111101001.$$

Из приведенных примеров видно, как реализуется основная формула алгоритма кодирования. Единица слева в формуле нужна для вычисления количества шагов, а числа справа указывают количество отделяемых разрядов на каждом шаге. Основная формула применяется до тех пор, пока не придем к $L(1)$ в центре формулы.

Вопросы для самопроверки

1. Какая кодовая схема называется разделимой?
2. Являются ли схемы фибоначчиевой системы и кода Левенштейна разделимыми?
3. Можно ли кодирование линейной формой Фибоначчи назвать разделимым?
4. При каком допущении равномерное двоичное кодирование разделимое?
5. По какому принципу разделяются числа в фибоначчиевой системе счисления?
6. По какому принципу ставят разделительные знаки в коде Левенштейна?
7. С чего начинается декодирование в коде Левенштейна?
8. Что записывают на последнем уровне в коде Левенштейна?
9. Какой метод кодирования базируется на теореме Цекендорфа?
10. Для кодирования каким методом применяют “жадный” алгоритм и нужно ли знать значения чисел Фибоначчи?
11. При кодировании каким методом одно число заменяют на три числа меньшей длины?
12. Каким способом код линейной формы Фибоначчи можно превратить в раделимое кодирование?

Упражнения

1.1. Запишите числа от 10 до 20 в фибоначчиевой системе.

1.2. Представьте число 50 в фибоначчиевой системе счисления.

1.3. Дан набор нескольких чисел в фибоначчиевой системе счисления: 100101001100011101. Что это за числа и сколько их? Представьте сумму этих чисел в фибоначчиевой системе счисления.

1.4. Разработайте алгоритм и предложите программу сложения чисел в фибоначчиевой системе счисления.

1.5. Разработайте алгоритм и предложите программу представления натурального числа в канонической линейной форме Фибоначчи.

1.6. Тройка чисел (j, a, b) для представления (1.7) канонической линейной формой Фибоначчи, записанная в фибоначчиевой системе счисления, имеет вид

$$10010110100001100001.$$

Какое число таким образом закодировано? Как записать это число в фибоначчиевой системе счисления и кодом Левенштейна?

1.7. Для числа 2024 найдите каноническую линейную форму Фибоначчи и запишите, закодировав каждое число этой тройки в фибоначчиевой системе счисления.

1.8. Для числа 2024 найдите каноническую линейную форму Фибоначчи и запишите, закодировав каждое число этой тройки в коде Левенштейна.

1.9. Представьте число 2024 кодом Левенштейна.

Глава 2

Алфавитное кодирование

Двоичным кодированием алфавита $A = \{a_1, a_2 \dots a_n\}$ называется инъективное отображение $\varphi : A \to B^*$, где B^* – это множество слов в алфавите $B = \{0, 1\}$. Инъективность означает, что для различных букв a_i, a_j *кодовые слова* $\varphi(a_i)$ и $\varphi(a_j)$ различны.

Отображение букв исходного алфавита $\varphi : A \to B^*$ определяет отображение слов $\varphi : A^* \to B^*$ по правилу: для *кодируемого сообщения* $C = a_{i_1} a_{i_2} \dots a_{i_k}$ *кодовое сообщение (код)* $\varphi(C) = B_{i_1} B_{i_2} \dots B_{i_k}$, где $B_j = \varphi(a_j)$ – кодовые слова.

Алфавитное кодирование (код) называется взаимно однозначным (*декодируемым*), если для любых различных слов $C, D \in A^*$ их коды различны $\varphi(C) \neq \varphi(D)$. Приведем пример не декодируемого кода.

Пример 2.1. *Пусть схема кодирования задана табл. 4. Закодируйте слово "муха" и предложите разные варианты декодирования.*

Таблица 4

Буква	а	л	м	н	о	у	с	х
Кодовое слово	101	111	0	1	10	01	00	11

Исходное сообщение "муха" переводим в кодовое сообщение 00111101.

Есть вариант декодирования сообщения следующей расстановкой разделительных знаков: 00, 111, 10, 1. Какое слово прочтет получатель кодового сообщения при данной декодировке?

Более простым вариантом алфавитного кодирования, аналогичным данному, считается кодирование выделенных слов в алфавите A, а не букв, то есть отображение $\varphi : A^* \to B^*$. Этот метод применяется в случае небольшого числа возможных исходных сообщений $S \subset A^*$ и $\varphi : S \to B^*$.

2.1. Основные классы алфавитных кодов

Дополнительным требованием к однозначному декодированию, которое предъявляется к методу кодирования, служит экономичность

(оптимальность по длине кодовых слов) кода. Для выполнения этих двух условий берут схемы кодирования следующих классов.

Определение 2.1. Способ кодирования (код) называется *равномерным*, если длины всех кодовых слов $\varphi(a_j)$ равны.

Код называется *префиксным*, если никакое кодовое слово не является началом другого кодового слова.

Код называется *суффиксным* (постфиксным), если никакое кодовое слово не является концом другого кодового слова.

Все утверждения будем рассматривать только для префиксного кода, поскольку их легко переформулировать для суффиксного кода.

Теорема 2.1. *Код любого из трех классов – равномерный, префиксный или суффиксный – является разделимым, а следовательно, взаимно однозначным.*

Доказательство. Разбивку кодового сообщения на кодовые слова для равномерного кода проводят отсчетом заданного числа бит.

Разбивку кодового сообщения на кодовые слова для префиксного кода проводят сравнением с кодовыми словами в направлении слева направо.

Разбивку кодового сообщения на кодовые слова для суффиксного кода проводят сравнением с кодовыми словами в направлении справа налево.

Пример 2.2 равномерного кода из четырех кодовых слов:

$$A_1 \sim 00, A_2 \sim 10, A_3 \sim 01, A_4 \sim 11;$$

префиксного кода из четырех кодовых слов:

$$B_1 \sim 0, B_2 \sim 10, B_3 \sim 110, B_4 \sim 111;$$

суффиксного кода из четырех кодовых слов:

$$C_1 \sim 0, C_2 \sim 01, C_3 \sim 011, C_4 \sim 111.$$

Декодируйте кодовое сообщение 11000111.

В случае равномерного кода исходное сообщение $A_4A_1A_3A_4$.

В случае префиксного кода исходное сообщение $B_3B_1B_1B_4$.

В случае суффиксного кода декодирование невозможно, так как нет нужной разбивки (11 не кодовое слово): $11 \leftarrow 0 \leftarrow 0 \leftarrow 0 \leftarrow 111$.

Теорема 2.2. *При равномерном кодировании m сообщений (или m букв алфавита) длина кодового слова $r \geq \log m$.*

Доказательство. В случае двоичного кодирования число возможных (кодовых) слов длины r равно 2^r, что влечет $m \leq 2^r$ – ограничение на число сообщений (или букв исходного алфавита). □

Если $m = 2^r$, то равномерный код будет *полным*. В примере 2.2 приведен полный равномерный код. Примером неполного равномерного служит код из трех кодовых слов: $A_1 \sim 00, A_2 \sim 10, A_3 \sim 01$.

Префиксный код предпочтительнее равномерного кода при алфавитном кодировании таких сообщений, где отдельные буквы встречаются чаще других. Для чаще встречающихся букв берут более короткие кодовые слова.

Для оценки эффективности кодирования вычисляют показатели: *математическое ожидание кода K*

$$M(K) = \sum_{k=1}^{n} p_k l_k,$$

если заданы вероятности p_k появления всех кодовых слов B_k;
или *цену кодирования* $c(\varphi) = \sum_{k=1}^{n} w_k l_k$, где w_k – частота появления, а l_k – длина кодового слова B_k. Эти формулы идентичны, так как вероятность и частота связаны формулой $w_k = N \cdot p_k$, где N – объем выборки.

Например, для алфавита с заданными в табл. 5 вероятностями

Таблица 5

Буква	о	к	л	м
Вероятность	1/2	1/4	1/8	1/8

предложим вариант равномерного кодирования K

Буква	о	к	л	м
Кодовое слово	00	01	10	11

и вариант префиксного кодирования K'

Буква	о	к	л	м
Кодовое слово	0	10	110	111

Вычислим $M(K) = 2 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 2 \cdot \frac{1}{8} + 2 \cdot \frac{1}{8} = 2$ для равномерного кода K и $M(K') = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + 3 \cdot \frac{1}{8} = 1,75$ для префиксного. кода K'. Значит, построенный префиксный код K' в этом случае более экономичный.

2.2. Графическое представление кода

Алфавитное кодирование допускает простую интерпретацию в виде графа, являющегося деревом. В двоичном случае это двоичное дерево. Правило построения полного двоичного дерева следующее.

Берем одну вершину в качестве корня дерева, которая составляет множество вершин нулевого этажа. Из этой вершины на первый этаж проводим два ребра, помечая их символами 0 (левое) и 1 (правое). Из каждой вершины рассматриваемого этажа проводим по два ребра, помечая их символами 0 и 1, в вершины следующего этажа и т. д.

Таким образом строится граф, называемый *кодовым деревом* полного равномерного кода с 2^r кодовыми словами, где r – число этажей. Все вершины последнего r-го этажа объявляются *концевыми вершинами*, а маршрут из корня в концевую вершину называется *кодовой ветвью*. Поскольку ребра помечены символом 0 или 1, то каждый маршрут имеет свой код, который и присваиваем соответствующей вершине. При этом длина каждого кодового слова r.

Например, для $r = 3$ это слова $000, 001, 010, 011, 100, 101, 110, 111$, графически представленные на рис. 1. Набор этих слов является также записью начальных целых неотрицательных чисел в двоичной системе в порядке возрастания.

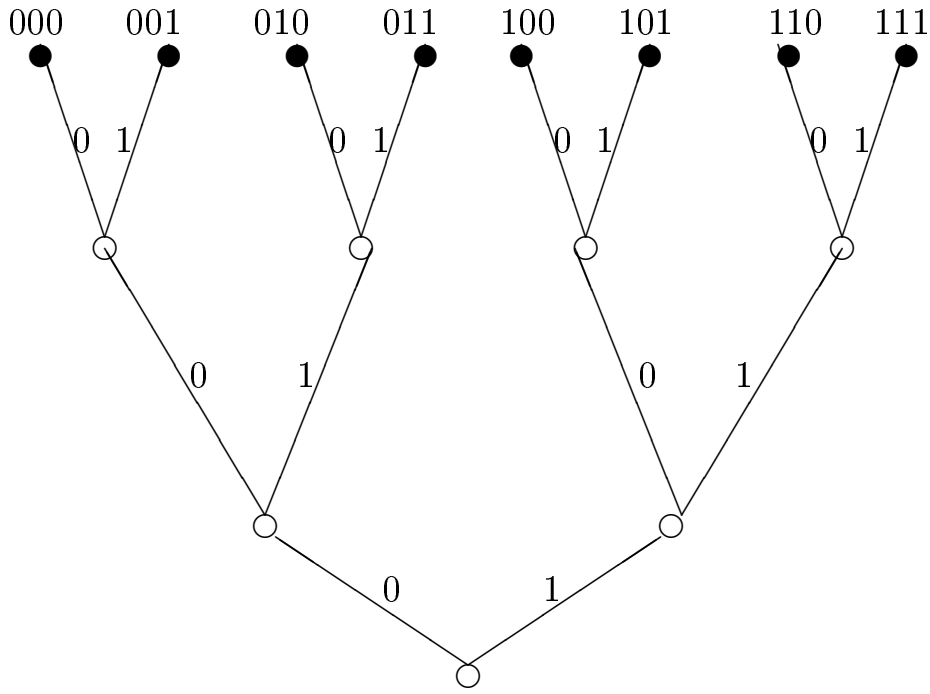

Рис. 1. Дерево для полного равномерного кода уровня 3

Если число m кодовых слов не представлено в виде степени двойки, то $2^{r-1} < m < 2^r$ и равномерный код будет неполным. Кодовое дерево для него получается удалением $2^r - m$ ребер последнего этажа. Вместо удаления ребер проще немного недостроить описанный выше граф. А именно на последнем шаге из каждой вершины предпоследнего $r-1$-го этажа провести или одно или два ребра так, чтоб их общее число равнялось m. Здесь также вершины r-го этажа объявляются концевыми с кодовыми словами, соответствующими маршруту по кодовой ветви. Кодовые вершины рисуем черными точками, а некодовые – кружочками (см. рис. 1 и 2). Для равномерного кода очевидно, что кодовые вершины являются концевыми.

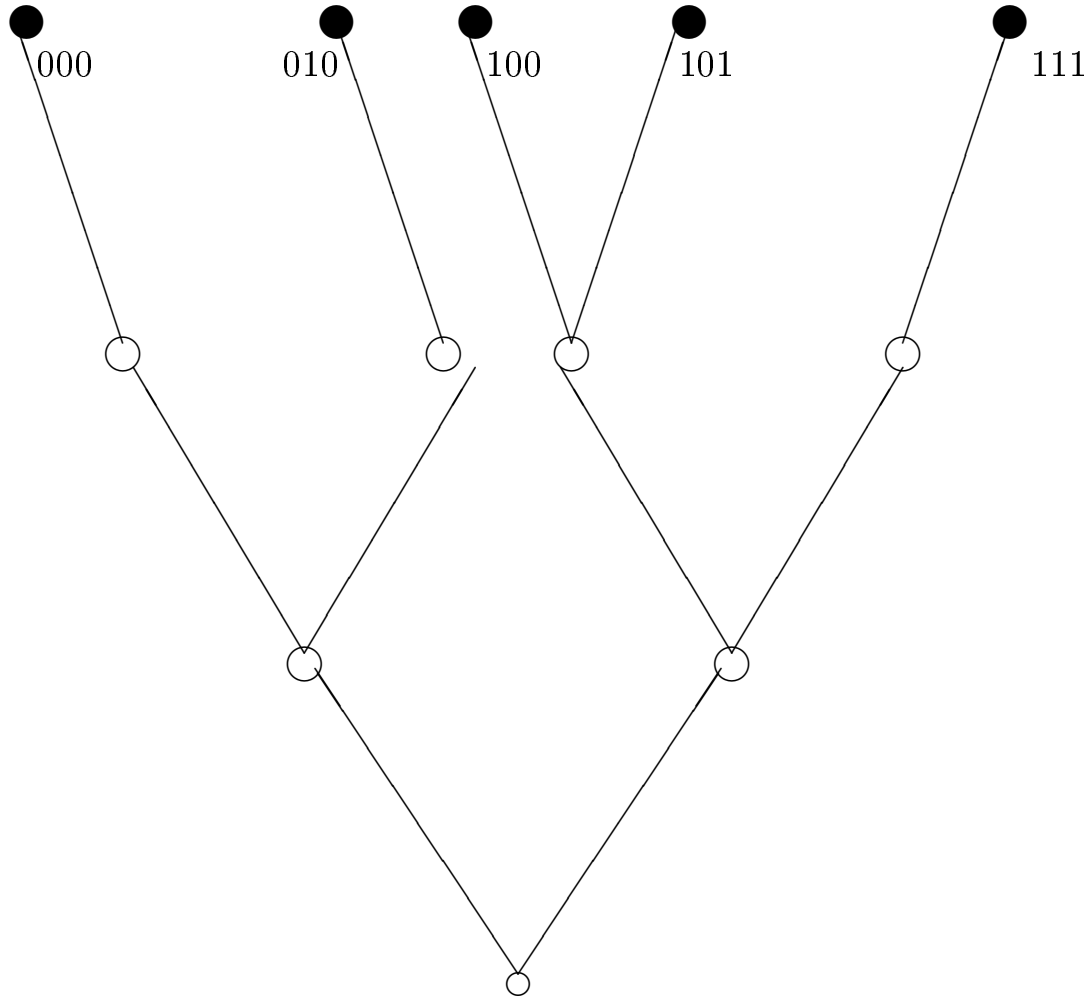

Рис. 2. Дерево равномерного кода с пятью кодовыми словами

Равномерный неполный код на рис. 2 не будет оптимальным, так как легко построить соответствующий префиксный код удалением ребер. Характерным свойством некодовых вершин графа на рис. 1 служит то, что из них на следующий этаж строили два ребра (ветвление). Если же ветвления нет (как на рис. 2), то соответствующую ветвь можно укоротить с переносом концевой вершины в смежную (на этаж ниже), что сделано на рис. 3.

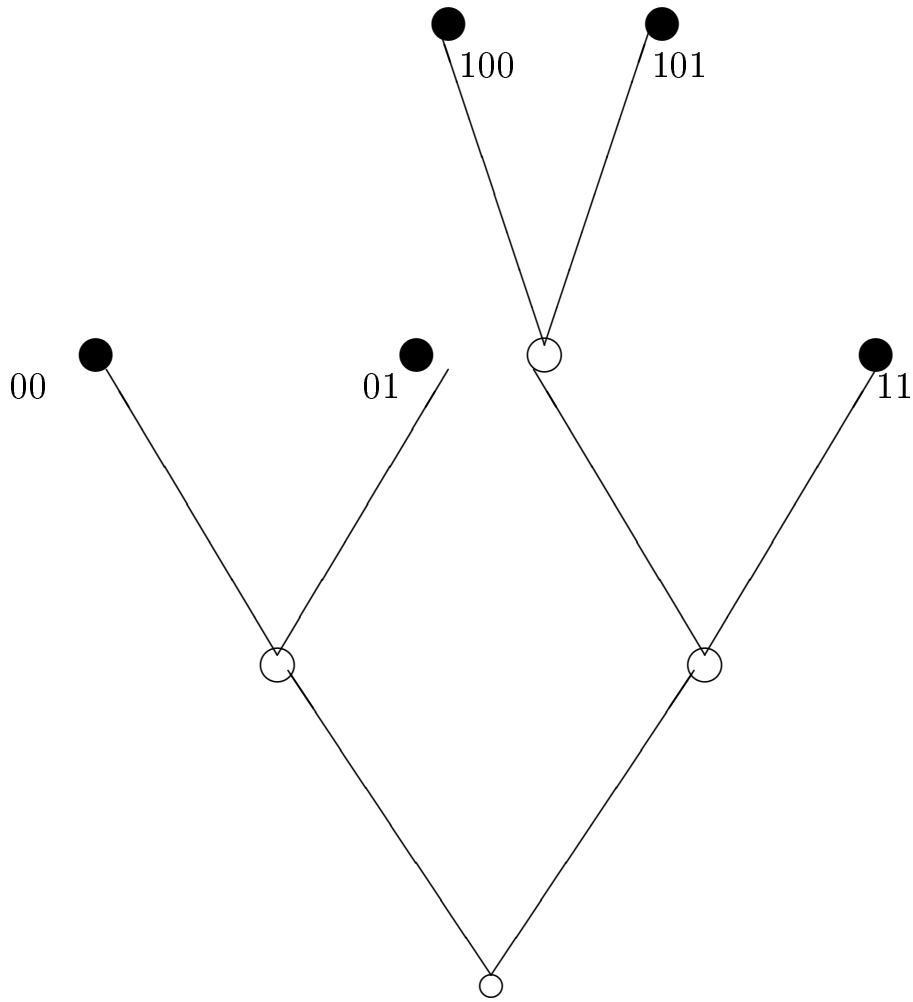

Рис. 3. Дерево префиксного кода с пятью кодовыми словами

Теорема 2.3. *Кодирование является префиксным, если все кодовые вершины кодового дерева являются концевыми.*

Доказательство. Все возможные начала произвольно выбранного кодового слова соответствуют вершинам (отмеченным кружочком) кодовой ветви в выбранную кодовую вершину.

2.3. Код Фано

Описанная ранее концепция эффективных кодов реализуется при построении *кода Фано*. Построение кода проиллюстрируем построением для него кодового дерева.

Алгоритм построения кода Фано

0. Начинаем с корневой вершины нулевого этажа, которая помечена как пустое множество $u =$.

1. Разбиваем множество букв алфавита $A = A_u = \{a_1, a_2 \dots a_m\}$ на два непересекающихся подмножества A_0 и A_1 (на первом шаге) или на A_{u0} и A_{u1} (в общем случае) так, чтобы суммарные вероятности (или суммарные частоты) для этих подмножеств были бы по возможности равны.

2. Из корня дерева на первом шаге (или из каждой вершины u рассматриваемого этажа на последующих шагах) проводим два ребра, которые помечаем символами 0 и 1: одно ребро будет инцидентно новой вершине $u0$ (вершине 0 при первом проходе), соответствующей подмножеству A_{u0} (A_0 при первом проходе), а другое ребро будет инцидентно новой вершине $u1$, соответствующей подмножеству A_{u1} (вершине 1 и подмножеству A_1 при первом проходе).

3. Каждое из этих подмножеств A_{u0}, A_{u1}, содержащее только одну букву, соответствует кодовой вершине, и на ней заканчивается кодовая ветвь.

4. Для каждого из остальных подмножеств A_{uj} рассматриваемого этажа, которое содержит более одной буквы, повторяем все шаги этого алгоритма с пункта 1 для множества A_{uj} вместо A, продолжая с вершины uj вместо корневой ($j = 0$ или $j = 1$).

Поскольку алфавит A конечен, то за конечное число шагов все подмножества перейдут в однобуквенные и все кодовые ветви будут построены.

Код Фано строится неоднозначно, так как (в общем случае) возможны различные разбиения на подмножества и разные способы их нумерации. Код Фано является префиксным, то есть декодируемым.

Алгоритм поясним следующим примером построения рис. 4.

Пример 2.3. *Для алфавита из четырех букв с вероятностями, заданными табл. 5 построить код Фано.*

1. Множество $A = \{\text{о, к, л, м}\}$ разобьем на $A_0 \cup A_1$, где $A_0 = \{\text{о}\}$, $A_1 = \{\text{к, л, м}\}$ с равными вероятностями.

2. Вершина 0 соответствует A_0, вершина 1 соответствует A_1.

3. Вершину 0 объявляем концевой $\text{о} \sim 0$.

4. Поскольку $|A_1| = 3$, то для множества $A_1 = \{\text{к, л, м}\}$ и соответствующей вершины 1 переходим к началу алгоритма.

1. Множество $A_1 = \{\text{к, л, м}\}$ разобьем на $A_{10} \cup A_{11}$, где $A_{10} = \{\text{к}\}$, $A_{11} = \{\text{л, м}\}$ с равными вероятностями.

2. Вершина 10 соответствует A_{10}, вершина 11 соответствует A_{11}.

3. Вершину 10 объявляем концевой $\text{к} \sim 10$.

4. Поскольку $|A_{11}| = 2$, то для множества $A_{11} = \{\text{л, м}\}$ и соответствующей вершины 11 переходим к началу алгоритма.

1 – 2 – 3. $A_{11} = A_{110} \cup A_{111}$; $\text{л} \sim 110$, $\text{м} \sim 111$.

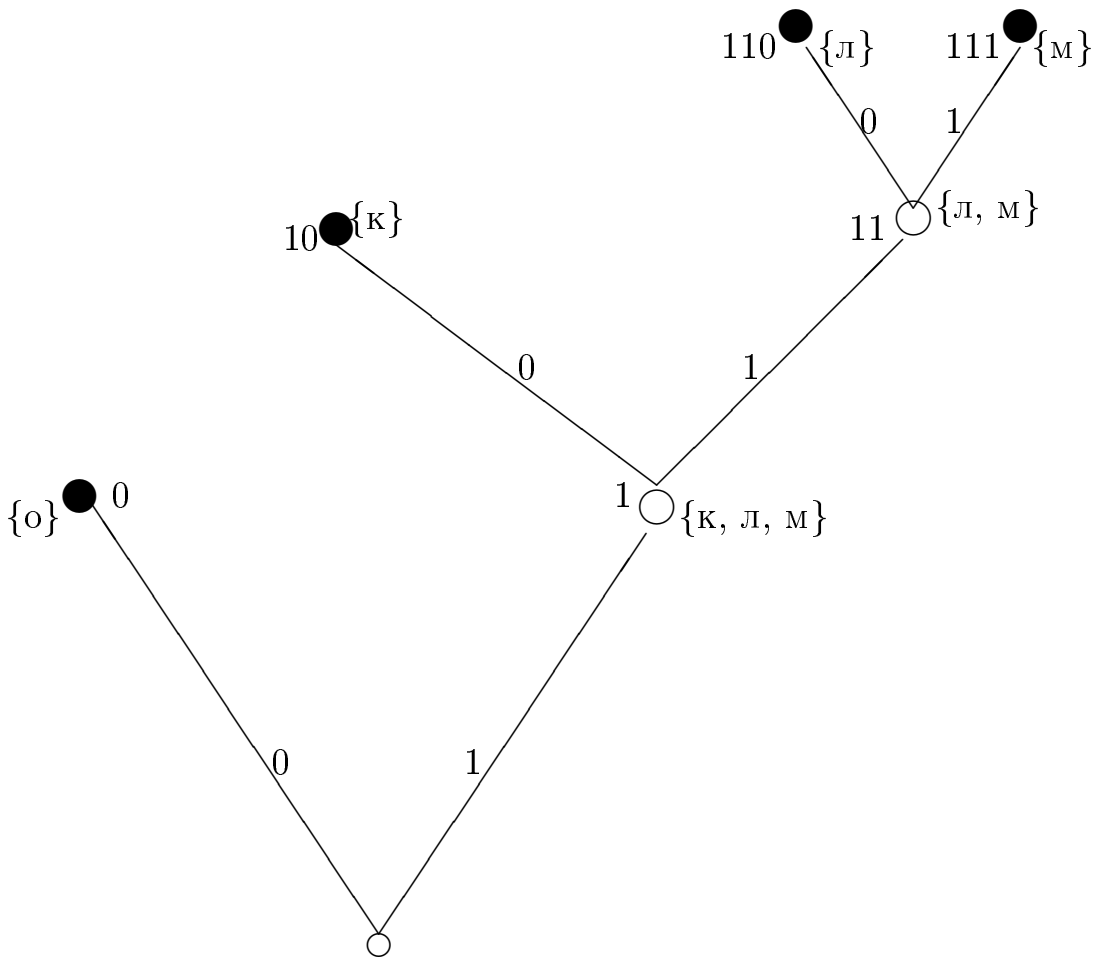

Рис. 4. Построение дерева префиксного кода Фано

Пример 2.4. *Для следующих частот появления букв:* $w(a) = 20$, $w(b) = 15$, $w(c) = 8$, $w(d) = 4$, $w(e) = 3$ *построить код Фано.*

Все построения будем производить прямо на строящемся кодовом дереве (рис. 5), делая отметки о виде множества и частоте для него. Поскольку коды вершин легко восстанавливаются по виду дерева, то не будем загромождать рисунок их указанием.

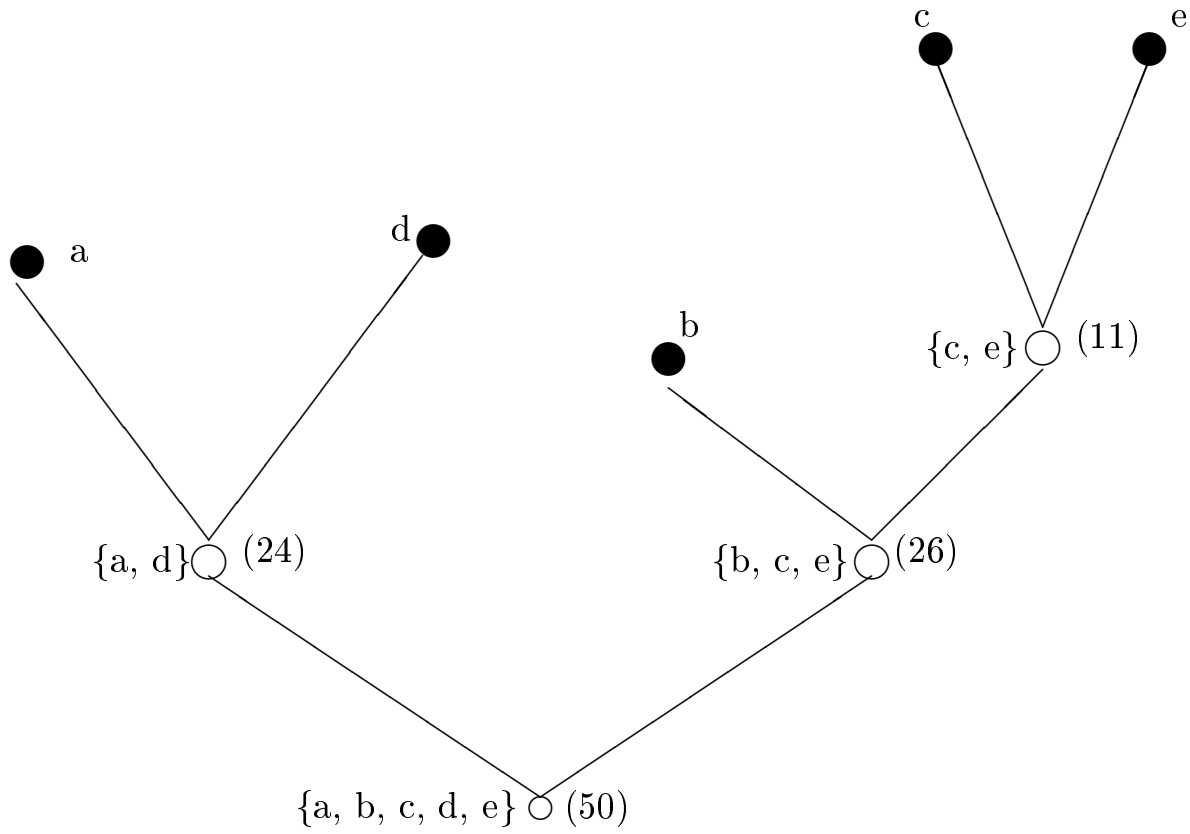

Рис. 5. Построение кода Фано на кодовом дереве

Ответ. $a \sim 00$, $b \sim 10$, $c \sim 110$, $d \sim 01$, $e \sim 111$.

2.4. Коды с заданным спектром. Код Шеннона

Рассмотрим двоичное кодирование $\varphi : A \to B^*$ и множество кодовых слов $\{B_1, B_2 \dots B_m\}$ таких, что $B_j = \varphi(a_j)$.

Определение 2.2. Набор чисел $S = \{l_1, l_2 \dots l_m\}$, где l_j – длина кодового слова B_j, называется *спектром* этого кодирования.

Для равномерного кодирования спектр постоянный $l_j \equiv r$ с условием $r \geq \log m$ согласно теореме 2.2. Для произвольного кода, представленного в виде кодового дерева, число l_j есть длина кодовой ветви в концевую вершину с кодовым словом B_j.

На спектр накладываются естественные ограничения, сформулированные в следующей теореме.

Теорема 2.4 (неравенство Макмиллана). *Если код префиксный со спектром* $S = \{l_1, l_2 \dots l_m\}$, *то выполняется неравенство*

$$\sum_{k=1}^{m} \frac{1}{2^{l_k}} \leq 1.$$

Доказательство. Рассмотрим кодовое дерево префиксного кода и самую длинную кодовую ветвь этого дерева. Пусть длина (число ребер) этой ветви равна r. Все кодовые вершины этого дерева, согласно теореме 2.3, являются концевыми. Достроим это дерево (другим цветом или пунктиром) до этажа r следующим образом.

Для каждой из этих достроек кодовая вершина B_j исходного дерева будет корнем, а вся достройка будет полным деревом с числом этажей $r - l_j$, поскольку всего r этажей, а вершина B_j на этаже l_j. Обозначим D_j множество вершин r-го этажа, достижимых во вновь построенном дереве из кодовой вершины B_j. Поскольку каждое вновь достроенное дерево (от корня B_j во множество D_j) полное с $r - l_j$ этажами, то число вершин этого множества $|D_j| = 2^{r-l_j}$. При этом учтены и кодовые вершины B_s на r-м этаже, для которых $|D_s| = 2^{r-r} = 1$, поскольку множество $D_s = B_s$ и есть эта вершина. Просуммируем число вершин всех этих непересекающихся множеств D_j и оценим эту сумму общим числом возможных вершин r-го этажа

$$2^{r-l_1} + 2^{r-l_2} + \dots + 2^{r-l_m} \leq 2^r.$$

Поделив обе части неравенства на 2^r, придем к неравенству Макмиллана

$$\frac{1}{2^{l_1}} + \frac{1}{2^{l_2}} + \dots + \frac{1}{2^{l_m}} \leq 1. \qquad \square$$

Аналогичными рассуждениями показывается, что неравенство Макмиллана является также и достаточным условием существования префиксного (суффиксного) кода с заданным спектром $S = (l_1, l_2 \dots l_m\}$.

Алгоритм построения префиксного кода по заданному спектру, удовлетворяющему нераверству Макмиллана, покажем на примере.

Пример 2.5. *Предложим префиксный код со следующим спектром длин кодовых слов* $S = \{2, 3, 3, 4, 4, 4, 4\}$.

Сначала убедимся, что такой префиксный код существует, составив для него неравенство Макмиллана

$$\frac{1}{2^2}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^4}+\frac{1}{2^4}+\frac{1}{2^4}+\frac{1}{2^4}=\frac{3}{4}<1.$$

Числа спектра S упорядочены по возрастанию. Поскольку $l_1=2$, то сначала строим дерево до второго этажа, где могут располагаться $4=2^2$ вершины. Из-за того, что в левой части неравенства Макмиллана сумма меньше или равна $\frac{3}{4}$ (у нас равна), на втором уровне отмечаем три вершины: одну черным цветом, так как одна двойка в спектре, а другие две – светлым. Кодовое слово B_1 длиной 2 отмечено.

Из каждой светлой вершины второго этажа проводим инцидентные ребра, в нашем случае в количестве четырех. В спектре две цифры 3, поэтому две новые вершины третьего этажа делаем черными, а две остальные – светлыми (рис. 6).

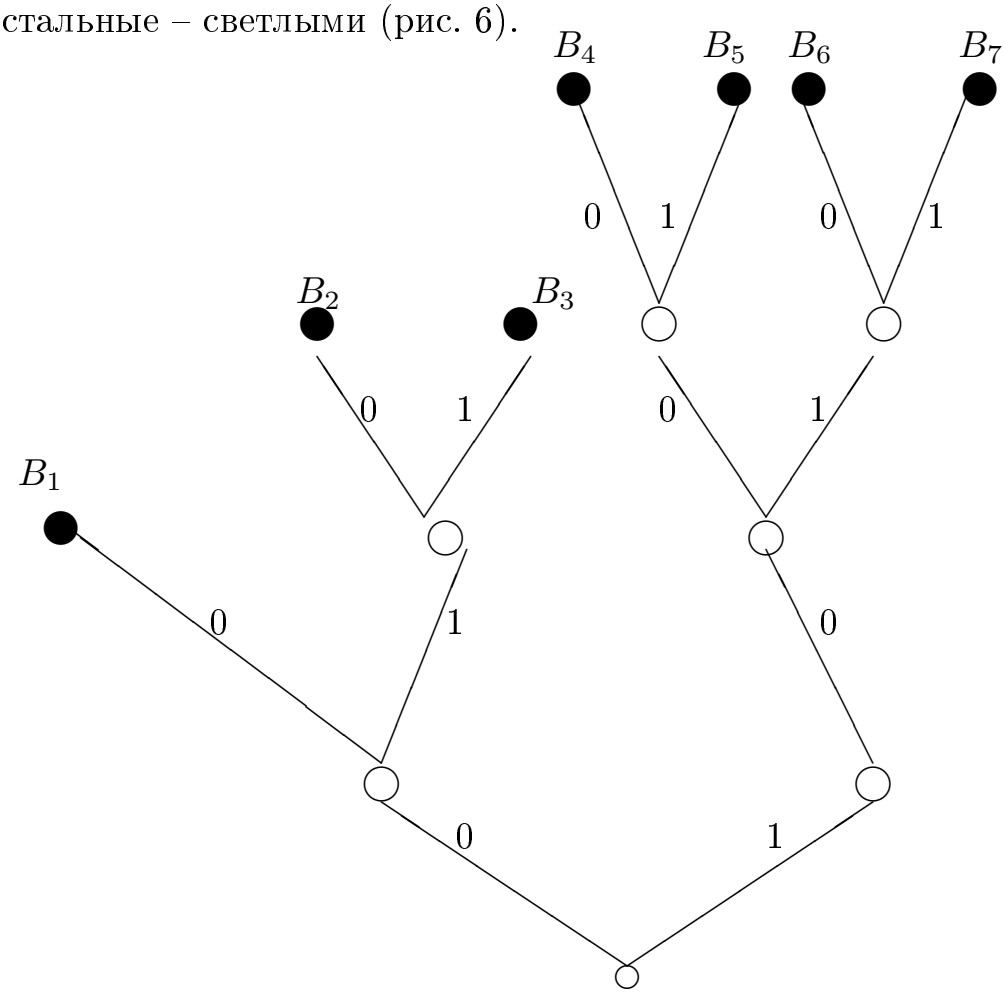

Рис. 6. Дерево префиксного кода с заданным спектром

Повторяем предыдущий шаг: из каждой светлой вершины третьего

этажа на четвертый проводим инцидентные ребра в количестве четырех. Поскольку в спектре четыре цифры 4, то все эти четыре вершины делаем черными. Дерево кода построено.

Получили следующие кодовые слова заданных длин: $B_1 = 00$, $B_2 = 010, B_3 = 011, B_4 = 1000, B_5 = 1001, B_6 = 1010, B_7 = 1011$.

Определение 2.3. Код называется *полным префиксным*, если добавление к нему кодового слова с любым кодом нарушает свойство префиксности.

Утверждение 2. *Код является полным префиксным тогда и только тогда, когда неравенство Макмиллана выполняется в виде равенства, то есть* $\sum_{k=1}^{m} \frac{1}{2^{l_k}} = 1$.

Код из примера 2.5 не является полным префиксным, так как к нему можно добавить кодовое слово $B_8 = 11$, или два кодовых слова 110 и 111 длины 3, или другие варианты большего числа добавленных слов большей длины.

Алгоритм построения префиксного кода с заданным спектром с помощью кодового дерева показан в примере 2.5. Существует другой, эквивалентный ему, алгоритм построения префиксного кода без привлечения кодовых деревьев. Этот алгоритм, основанный на двоичном разложении чисел, был предложен Шенноном.

В алгоритме Шеннона для спектра S, упорядоченного по возрастанию, рассматриваются числа

$$q_1 = 0, q_2 = \frac{1}{2^{l_1}}, q_3 = \frac{1}{2^{l_1}} + \frac{1}{2^{l_2}}, q_4 = \frac{1}{2^{l_1}} + \frac{1}{2^{l_2}} + \frac{1}{2^{l_3}}, \ldots, q_m = \sum_{i=1}^{m-1} \frac{1}{2^{l_i}}.$$

Из неравества Макмиллана вытекает ограничение сверху для всех этих чисел $0 \le q_i < 1$. Поэтому каждое из них можно записать в виде двоичной дроби, взяв в двоичном разложении числа q_i ровно l_i знаков после запятой:

$$q_1 = 0, 00 \ldots 0 = 0, c_{11} c_{12} \ldots c_{1 l_1},$$

$$q_2 = 0, c_{21} c_{22} \ldots c_{2 l_2}, \qquad \ldots$$

$$q_m = 0, c_{m1} c_{m2} \ldots c_{m l_m}.$$

В качестве кодовых слов B_i возьмем двоичные наборы, которые получаются из данных разложений отбрасыванием слева нуля и запятой.

По построению набор кодовых слов $\{B_1, B_2 \dots B_m\}$ имеет требуемый спектр. Если $j < k$, то по построению $q_k - q_j \geq \frac{1}{2^{l_j}}$. Это неравенство означает, что слово B_j не может быть началом слова B_k (которое не короче). Значит, код префиксный.

Пример 2.6. *Построить код Шеннона с заданным спектром* $S = \{2, 2, 2, 3, 4, 4\}$.

Поскольку $\frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} = 1$, то префиксный код с таким спектром существует, и он будет полным префиксным кодом. Последовательно вычисляем двоичные разложения чисел q_j с l_j знаками после запятой, которые выделим подчеркиванием. Эти подчеркнутые знаки и будут кодовыми словами требуемой длины для префиксного кода (рис. 7).

	$q_1 =$	$0 = 0,\underline{00}$	$(l_1 = 2)$
$+$		$1/4 = 0,01$	
	$q_2 =$	$1/4 = 0,\underline{01}$	$(l_2 = 2)$
$+$		$1/4 = 0,01$	
	$q_3 =$	$1/2 = 0,\underline{10}$	$(l_3 = 2)$
$+$		$1/4 = 0,01$	
	$q_4 =$	$3/4 = 0,\underline{110}$	$(l_4 = 3)$
$+$		$1/8 = 0,001$	
	$q_5 =$	$7/8 = 0,\underline{1110}$	$(l_5 = 4)$
$+$		$1/16 = 0,0001$	
	$q_6 =$	$5/16 = 0,\underline{1111}$	$(l_6 = 4)$

Рис. 7. Схема построения кода Шеннона

Таким образом, получили набор кодовых слов полного префиксного кода

$$\{00, 01, 10, 110, 1110, 1111\}.$$

2.5. Оптимальное кодирование. Код Хаффмана

Как уже отмечалось, при алфавитном кодировании сообщений, в котором буквы алфавита появляются с разной вероятностью (частотой), нужно придерживаться общего правила: чаще встречающиеся буквы кодируются более короткими словами. По такому принципу строится код Фано, который во многих случаях получается наиболее

экономным с точки зрения средней длины кодовых слов. Однако могут существовать и более оптимальные коды.

Определение 2.4. Для двоичного кодирования $\varphi : A \to B^*$ со спектром кодовых слов $S = (l_1, l_2, l_3 \dots l_m\}$ и заданными частотами $(w_1, w_2 \dots w_m\}$ появления букв исходного алфавита $A = (a_1, a_2 \dots a_m\}$ *ценой* (стоимостью) кодирования называется

$$c(\varphi) = \sum_{i=1}^{m} w_i l_i.$$

Определение 2.5. Взаимно однозначное (двоичное) кодирование φ_0 называется *оптимальным*, если выполняется $c(\varphi_0) \leq c(\varphi)$ для любого другого взаимно однозначного кода φ.

Определение 2.6. Процедура перехода от входного алфавита $A = (a_1, a_2 \dots a_{m-2}, a_{m-1}, a_m\}$ с частотами $(w_1, w_2 \dots w_{m-2}, w_{m-1}, w_m\}$ к другому входному алфавиту $A' = (a_1, a_2 \dots a_{m-2}, a'_{m-1}\}$ с частотами $(w_1, w_2 \dots w_{m-2}, w'_{m-1}\}$, где $w'_{m-1} = w_{m-1} + w_m$, называется *сжатием алфавита.*

Обратную процедуру опишем относительно префиксных кодов, а не алфавитов.

Определение 2.7. Если существует код $\varphi' : A' \to B^*$ для сжатия A' алфавита A с набором кодовых слов $\{B_1, B_2 \dots B_{m-2}, B'_{m-1}\}$, то соответствующий ему код $\varphi : A \to B^*$ исходного алфавита, совпадающий на первых $m-2$ буквах ($\varphi(a_j) = B_j = \varphi'(a_j)$ при $1 \leq j \leq m-2$), такой, что $\varphi(a_{m-1}) = B'_{m-1}0$, $\varphi(a_m) = B'_{m-1}1$, называется *расщеплением* кода.

В следующей теореме отмечаются свойства оптимальных кодов.

Теорема 2.5. *Пусть для известных частот w_i (или вероятностей p_i) букв исходного алфавита кодирование $\varphi : A \to B^*$ оптимальное.*

1. Если $w_i > w_j$, то $l_i \leq l_j$.

2. Исходный алфавит $A = \{a_1, a_2 \dots a_m\}$ можно упорядочить так, что $p_1 \geq p_2 \geq \dots \geq p_{m-1} \geq p_m$ и $l_1 \leq l_2 \leq \dots \leq l_{m-1} \leq l_m$.

3. Если φ – оптимальное префиксное кодирование, то найдутся два кодовых слова максимальной длины ($l_{m-1} = l_m$ в случае упорядоченного кода), различающиеся лишь в последнем символе, а именно вида $B0$ и $B1$.

Доказательство. 1. Если взять противоположный случай $w_i > w_j$ и $l_i > l_j$, то возьмем новый код φ', где кодовые слова B_i и B_j поменяем местами. Получим $c(\varphi') < c(\varphi)$, что противоречит оптимальности.

2. Упорядочим алфавит по убыванию частот появления букв. Для соседних букв с разными частотами $w_i > w_j$ нужное неравенство вытекает из пункта 1. Набор букв с одинаковыми частотами упорядочим требуемым способом по длинам их кодовых слов.

3. От противного. Пусть для оптимального префиксного упорядоченного кодирования среди кодовых слов встречается только одно из слов $B0$ и $B1$. Поскольку (согласно префиксности) слово B не кодовое, то, отбросив последний символ у $B0$ или $B1$, получим новый код с сохранением свойства префиксности. Поскольку цена кодирования при этом уменьшится, то получили противоречие.

Итак, любой префиксный код согласно пункту 2 теоремы 2.5 можно упорядочить.

Теорема 2.6 (редукции). *Пусть даны два взаимосвязанных кода* $\varphi : A \to B^*$ *и* $\varphi' : A' \to B^*$ *с совпадающими (за одним исключением* $p_{m-1} = p' + p''$*) наборами вероятностей (частот)* $p_1, p_2 \dots p_{m-2}$ *и кодовых слов* $B_1, B_2 \dots B_{m-2}$ *(за исключением кодового слова* B *в коде* φ' *и слова* $B0$ *и* $B1$ *в коде* φ*).*

1. Если φ *– оптимальное префиксное упорядоченное кодирование с условием, что* p', p'' *есть минимальные вероятности, то* φ' *после упорядочения становится оптимальным префиксным кодом.*

2. Если φ' *– оптимальное префиксное упорядоченное (*p_{m-1} *минимально) кодирование, то* φ *также оптимальное префиксное упорядоченное (при* $p' \geq p''$*) кодирование.*

Очевидно, что из префиксности одного из этих кодов следует префиксность другого. Для доказательства оптимальности нужно перебрать несколько вариантов.

На базе теоремы редукции строится *код Хаффмана*, алгоритм которого состоит из двух стадий. Перед первой стадией упорядочиваем по убыванию частоты (вероятности) появления букв исходного алфавита. На каждом шаге первой стадии проводим сжатие алфавита, объединяя две буквы с наименьшей частотой в группу букв, складывая их частоты. А эту группу далее снова называем буквой нового алфавита. Частоты нового алфавита вновь упорядочиваем по убыванию.

Повторяем шаг этой стадии до тех пор, пока алфавит не превратится в двухбуквенный.

На второй стадии строим, начиная с корня, дерево кодирования, повторяя шаги предыдущей стадии в обратном порядке по правилу расщепления кода. Как только в этом процессе выделяется отдельная буква первоначального алфавита, соответствующую вершину отмечаем как конец кодовой ветви (закрашиваем).

Всю процедуру построения можно упростить, отказавшись от построения дерева на второй стадии, если на первой стадии проводить построение кодового дерева в последовательности от кодовых вершин к корню. Тогда на второй стадии расставляем лишь коды ребер 0 и 1 и выписываем кодовые слова для каждой буквы исходного алфавита.

Пример 2.7. Исходный алфавит $A = \{a, b, c, d, e, f\}$ с частотами $w(a) = 8$, $w(b) = w(c) = 3$, $w(d) = w(e) = w(f) = 2$, которые уже упорядочены по убыванию.

На первом шаге перейдем к алфавиту $A' = \{a, b, c, d, \{e, f\}\}$ с частотами $w(a) = 8$, $w(b) = w(c) = 3$, $w(d) = 2$, $w(\{e, f\}) = 4$, которые расположим в порядке $w(a) = 8$, $w(\{e, f\}) = 4$, $w(b) = w(c) = 3$, $w(d) = 2$.

В результате выполнения второго шага получим упорядоченный алфавит $A'' = \{a, \{c, d\}, \{e, f\}, b\}$ с частотами $w(a) = 8$, $w(\{c, d\}) = 5$, $w(\{e, f\}) = 4$, $w(b) = 3$.

На третьем шаге опять объединим две последние буквы и упорядочим $A''' = \{a, \{e, f, b\}, \{c, d\}\}$ с частотами $w(a) = 8$, $w(\{e, f, b\}) = 7$, $w(\{c, d\}) = 5$.

После четвертого шага (последнего шага первой стадии) весь исходный алфавит разбился на две группы $\{a\}$ и $\{b, c, d, e, f\}$ с частотами $w(a) = 8$, $w(\{e, f, b, c, d\}) = 12$.

На второй стадии строим дерево кодирования, повторяя шаги в обратном порядке (построение рис. 8 снизу ввверх).

Расставляя 0 у левой ветви, а 1 – у правой ветви, получаем код Хаффмана данного примера

$$a \sim 0,\ b \sim 100,\ c \sim 110,\ d \sim 111,\ e \sim 1010,\ f \sim 1011.$$

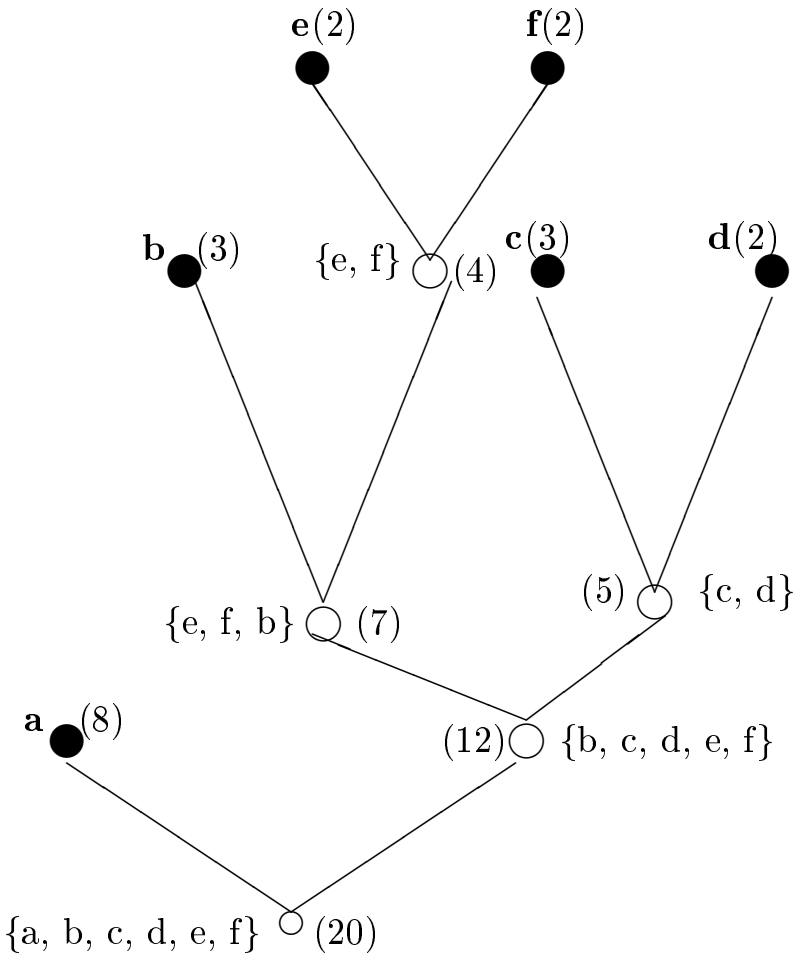

Рис. 8. Процесс построения дерева кода Хаффмана

Пример 2.8. Для условия из предыдущего примера 2.7 построим код Фано. Но здесь все построение идет в одну стадию снизу вверх, а не сверху вниз (как на первой стадии для кода Хаффмана). Все шаги восстанавливаются по рис. 9.

Расставляя 0 у левой ветви, а 1 – у правой ветви, получим код Фано данного примера

$$a \sim 00,\ b \sim 100,\ c \sim 110,\ d \sim 111,\ e \sim 101,\ f \sim 01.$$

Теперь вычислим цену кода Хаффмана φ_0 из примера 2.7

$$c(\varphi_0) = 8 \cdot 1 + 3 \cdot 3 + 3 \cdot 3 + 2 \cdot 3 + 2 \cdot 4 + 2 \cdot 4 = 48.$$

и цену кода Фано φ_1 из примера 2.8

$$c(\varphi_1) = 8 \cdot 2 + 3 \cdot 3 + 3 \cdot 3 + 2 \cdot 3 + 2 \cdot 3 + 2 \cdot 2 = 50.$$

Цена кода Фано оказалась выше цены оптимального кода Хаффмана. Однако встречается случай, когда оба кода окажутся оптимальными.

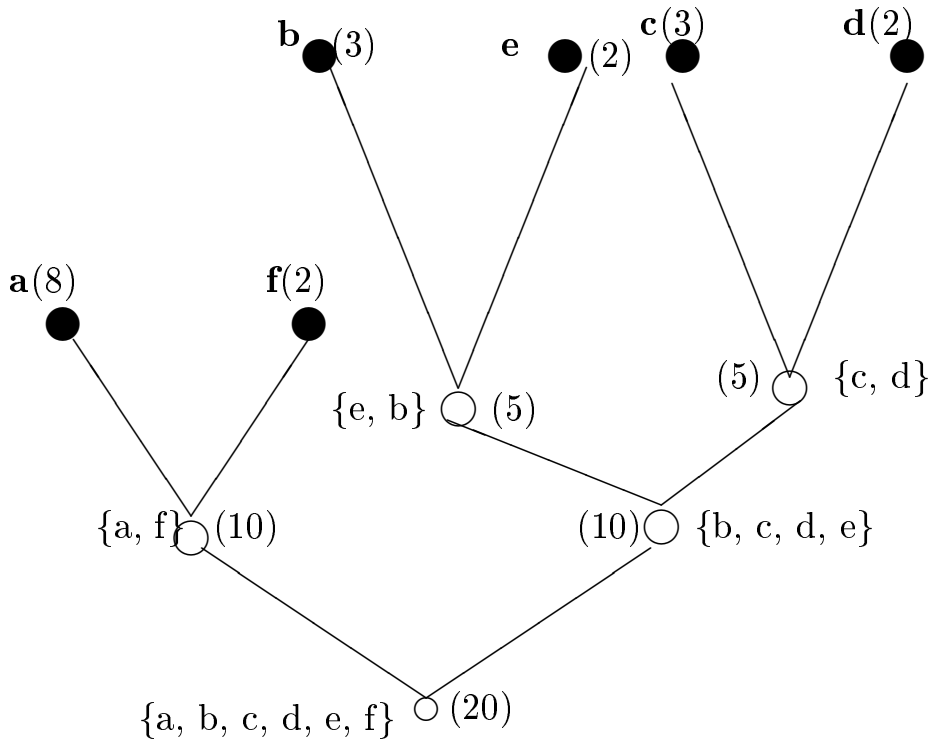

Рис. 9. Процесс построения дерева кода Фано

Теорема 2.7. *Если вероятности появления каждой из букв исходного алфавита A выражается как отрицательная степень двойки (то есть вида $p_i = 2^{-n_i}$), то все три кода: Фано, Шеннона и Хаффмана – оптимальные, поскольку их спектры совпадают и равны $S = \{n_1, n_2 \dots n_m\}$).*

При доказательстве опираются на равенство Макмиллана для полного префиксного кода, поскольку полнота кода вытекает из основного свойства дискретной случайной величины, утверждающего, что сумма вероятностей равна единице. Для кода Фано нужно правильно организовать перебор вариантов.

Вопросы для самопроверки

1. Какое кодирование называется алфавитным?
2. Когда алфавитное кодирование называется декодируемым?
3. Какой код называется равномерным?
4. Какой код называется префиксным?
5. При каких условиях равномерный код будет префиксным?
6. Верно ли, что любой равномерный код является суффиксным?
7. Может ли код быть одновременно префиксным и суффиксным? Обязан ли он при этом быть равномерным?

8. Какой из трех видов кодов – равномерный, префиксный, суффиксный – относится к классу разделимых кодов? Какой принцип разделения кодовых слов лежит в основе алгоритмов декодирования каждого из кодов?

9. Может ли равномерный код быть неполным?

10. Как выглядит дерево кодирования для полного равномерного кода; а как – для неполного равномерного кода?

11. Как выглядит дерево кодирования для префиксного кода?

12. Как строится код Фано?

13. Что называется спектром кодирования?

14. Какое кодирование считается оптимальным?

15. В чем заключается процедура сжатия алфавита?

16. Как определяется расщепление кода?

17. Какой из кодов – равномерный, Фано, Шеннона или Хаффмана – всегда будет оптимальным? Приведите примеры остальных типов кодов, не являющихся оптимальными.

Упражнения

2.1. Предложите равномерный код для алфавита из табл. 4 примера 2.1, сохраняющий два кодовых слова. Будет ли этот код полным? Закодируйте слово “муха” этим равномерным кодом. Опишите алгоритм декодирования.

2.2. Предложите равномерный код для кодирования вашего имени. Будет ли этот код полным?

2.3. Если от кода примера 2.1 оставить четыре буквы – а, м, у, х – с предложенными кодовыми словами, то к какому из трех типов кода, введеных в определении 2.1, он относится? Можно ли однозначно декодировать в этом коде закодированное слово “муха”?

2.4. Постройте графы равномерного и префиксного кодов для примера 2.3.

2.5. Вычислите спектр кода для графа на рис. 3.

2.6. Полный или неполный спектр кода получится при равномерном кодировании вашего имени? Проверьте для этого кода неравенство Макмиллана.

2.7. Постройте код со спектром $S = \{2, 3, 3, 3, 4\}$. Будет ли код полным?

Глава 3

Коды с исправлением ошибок

3.1. Общие принципы построения блоковых кодов

Коды, обнаруживающие и исправляющие ошибки, относятся к избыточным алфавитным двоичным кодам. Пусть исходное двоичное сообщение разбито на блоки длиной k, каждый из которых кодируется кодовым словом такой длины n, что $n > k$ – *избыточность* кода. Такие равномерные коды называются (n, k)-*коды*.

Обозначим $V_n = \{a = (a_1, a_2 \dots a_n) \quad | \quad a_i = 0 \text{ или } a_i = 1\}$ – множество всех двоичных наборов длины n.

Определение 3.1. *Расстоянием Хэмминга* между двумя двоичными наборами $a, b \in V_n$ называется число $d(a, b)$, равное количеству разрядов, на которых эти наборы различны.

Определение 3.2. *Шаром* радиуса r с центром в точке $a \in V_n$ называется подмножество $U_r(a) \subset V_n$ всех таких наборов $b \in V_n$, что $d(a, b) \leq r$.

Расстояние Хэмминга обладает на множестве V_n свойствами метрики (расстояния в геометрии):

1) $d(a, b) \geq 0$, причем $d(a, b) = 0$ тогда и только тогда, когда элементы множества совпадают, то есть $a = b$ (неотрицательность);

2) $d(a, b) = d(b, a)$ (симметричность);

3) $d(a, b) + d(b, c) \geq d(a, c)$ (неравенство треугольника).

При передаче по неустойчивому каналу связи закодированного сообщения могут возникнуть ошибки следующих видов:

1) замещение символа $0 \to 1$ или $1 \to 0$;

2) стирание символа $0 \to$ или $1 \to$;

3) появление символа $\to 0$ или $\to 1$.

Предполагаем, что ошибки разного типа вызываются разными причинами, которые не возникают одновременно. В случае равномерного кода с разделительным знаком между словами наличие ошибок второго и третьего вида сразу обнаруживается. Поэтому далее будем считать возможными только ошибки замещения. Другим ограничением

служит число возможных ошибок замещения. Если при передаче сообщения вместо слова $a \in V_n$ получено слово $b \in V_n$, то число ошибок равно $d(a, b)$.

Поскольку всего возможных сообщений 2^r, а возможных кодовых слов 2^n, то в качестве кодовых слов используют небольшую часть множества V_n. Задача состоит в оптимальном выборе множества K кодовых слов равномерного кода

$$K = \{B_1, B_2 \dots B_m\} \subset V_n. \tag{3.1}$$

Определение 3.3. Код K называется *кодом, обнаруживающим r ошибок*, если существует алгоритм, который при не более r ошибок типа замещения в любом кодовом слове B_j верно указывает число ошибок в предъявленном слове.

В качестве такого алгоритма обычно берут алгоритм перебора расстояний полученного сообщения B от всех кодовых слов B_j для определения $\min_j d(B, B_j)$. Если $\min_j d(B, B_j) = 0$, то ошибок нет, сообщение B совпадает с одним из кодовых слов, которое указывается. Если $\min_j d(B, B_j) = 1$, то ошибка одна. В случае $r = 1$ эту ошибку часто указать невозможно, поскольку этот минимум достигается не на одном кодовом слове.

Определение 3.4. Код K называется *кодом, исправляющим r ошибок*, если существует алгоритм, который при наличии в любом кодовом слове B_j не более r ошибок типа замещения восстанавливает исходное кодовое слово B_j.

Определение 3.5. *Кодовым расстоянием* для кода K называется

$$d(K) = \min_{1 \le i,j \le m} d(B_i, B_j).$$

Кодовое расстояние $d = d(K)$ часто указывают как третий параметр в записи типа кода (n, k, d)-код (вместо (n, k)-код).

Общие требования к кодам:

– способность обнаруживать (или исправлять) как можно большее число ошибок,

– как можно меньшая избыточность,

– простота кодирования и декодирования.

Приведенные требования противоречат друг другу, поэтому при построении кодов выбирают компромисные решения. Организовать

обнаружение и исправление ошибок удается за счет избыточности кода. Блоковый код называется *систематическим*, если среди n координат кодовых слов выделены ровно k координат, значения которых в каждом кодовом слове повторяют k координат исходного сообщения. Эти k координат кодового слова называются *информационными* координатами. Наиболее простой и удобный случай, когда информационные координаты располагаются в начале кодового слова. В общем случае информационные координаты располагаются произвольно и упорядочены по-другому. Остальные $n-k$ координат систематического кода называются *проверочными*. Код называется *несистематическим*, если число информационных координат меньше k.

Число $n-k$ служит показателем избыточности кода. Другим показателем избыточности служит *скорость кодирования* $R = \frac{k}{n}$.

3.2. Коды, обнаруживающие ошибки

Обозначим $S_r(n)$ – число элементов V_n в шаре $U_r(a)$ радиуса r. Заметим, что это число $S_r(n)$ не зависит от центра шара $a \in V_n$.

Теорема 3.1. *Код K обнаруживает r ошибок, если ни одно из кодовых слов B_j не попадает в шар $U_r(B_i)$ с центром в другом кодовом слове, что соответствует условию $d(K) \geq r+1$.*

Доказательство. Процесс декодирования осуществляется сравнением полученного слова B со всеми кодовыми словами $B_j \in K$. Если совпадений не обнаружено, то при передаче информации допущены ошибки. Число ошибок оценивается величиной $\min_j d(B, B_j)$.

Случай, когда $r = d(K)$, допускает вариант неверного декодирования таких кодовых слов B_{j_1} и B_{j_2}, что $d(B_{j_1}, B_{j_2}) = d(K)$. Может быть отправлено B_{j_1}, а r ошибок передачи информации превратили его в B_{j_2}.

Если же $d(K) \geq r+1$ и допускается r ошибок, то при отправке слова B_j получаем слово B из шара $U_r(B_j)$, которое совпадает с кодовым словом только в случае $B = B_j$ (когда нет ошибок).

Теорема 3.2. *Число элементов шара радиуса r равно*

$$S_r(n) = C_n^0 + C_n^1 + \ldots + C_n^r.$$

Доказательство. Одна точка (просуммирована в виде $C_n^0 = 1$) соответствует центру шара. Количество точек на расстоянии 1 от центра равно числу координат $n = C_n^1$.

Количество точек на расстоянии 2 от центра шара равно числу $C_n^2 = \frac{n(n-1)}{2}$, то есть числу способов выбрать две позиции (координаты) из n. В шар радиуса 2 включаем все три этих вида точек $S_r(2) = C_n^0 + C_n^1 + C_n^2$ и так далее.

Заметим, что $C_n^0 + C_n^1 + \ldots + C_n^n = 2^n$, то есть шар радиуса n накрывает все множество V_n.

Итак, код K обнаруживает одну ошибку, если $d(K) \geq 2$. Поэтому в качестве такого кода с условием оптимальности (максимально большой по числу кодовых слов) можно предложить код с кодовыми словами, отстоящими от соседнего кодового слова на 2. Если все слова множества V_n разбить на два непересекающихся подмножества K_0 – четная сумма отсчетов и K_1 – нечетная сумма отсчетов, то внутри каждого подмножества это свойство выполняется. Легко показать, что $|K_0| = |K_1| = 2^{n-1}$. В качестве множества K кодовых слов можно взять любое из множеств K_0 или K_1. Таким образом получается код проверки на четность.

Код проверки на четность относится к блоковым $(n, n-1, 2)$-кодам, поскольку кодовое расстояние этого кода равно двум. Кодируется слово длины $k = n - 1$ в двоичном алфавите $\{0, 1\}$ словом длины n добавлением в конце символа 0 или 1 так, что сумма всех отсчетов получается четная (то есть из множества K_0). Таким образом, первые k координат информационные, а последняя координата проверочная. При декодировании сообщения B проверяется свойство четности и (при необнаружении ошибок) последний символ отбрасывается. Если же сумма отсчетов принятого сообщения B оказывается нечетной, то выдается указание о наличии ошибки. Существует два приема для исправления ошибки. Обычно запрашивают у отправителя повтор сообщения B. Если это затруднительно, то предлагают набор кодовых слов B_j на расстоянии 1 от принятого сообщения B в качестве возможного варианта вместо B. Существует k таких вариантов по числу координат исходного сообщения.

Описанный код удовлетворяет двум основным требованиям: минимальной избыточности и простоте кодирования – декодирования.

3.3. Коды, исправляющие ошибки

Обозначим $M_r(n)$ максимальное число слов длины n, образующих код, исправляющий r ошибок.

Теорема 3.3. *Код K исправляет r ошибок тогда и только тогда, когда шары $U_r(B_j)$ для разных j не пересекаются, что соответствует условию $d(K) \geq 2r+1$.*

Доказательство. Если $B \in U_r(B_j) \cap U_r(B_i) \neq \emptyset$, то по неравенству треугольника $d(B_j, B_i) \leq d(B, B_j) + d(B, B_i) \leq 2r$.

Все слова из шара $U_r(B_j)$ декодируются словом B_j. Тогда условие однозначного декодирования: $U_r(B_j) \cap U_r(B_i) = \emptyset$ при $i \neq j$. Если $d(K) = 2r$, то $d(B_j, B_i) = 2r$ для некоторых $i \neq j$. Эти $2r$ координат разобьем на два непересекающихся множества A и D по r координат. Если у сообщения B_i поменяем координаты из множества A, а у сообщения B_j поменяем координаты из множества D, то получим одно и то же сообщение B из разных шаров $U_r(B_j)$ и $U_r(B_i)$, что означает $B \in U_r(B_j) \cap U_r(B_i) \neq \emptyset$. Если же $d(K) = 2r+1$, то таким путем придем к сообщению $B \notin U_r(B_j) \cup U_r(B_i)$, поскольку останется одна незадействованная координата.

Таким образом, дополнительно установили, что при большем числе ошибок, чем r, данный код способен их обнаружить, хотя не может однозначно исправить.

Теорема 3.4. *Максимальное число кодовых слов для кода исправляющего r ошибок оценивается*

$$\frac{2^n}{S_{2r}(n)} \leq M_r(n) \leq \frac{2^n}{S_r(n)}.$$

Доказательство. Пусть код (3.1) исправляет r ошибок. По теореме 3.3 шары $U_r(B_1)$, $U_r(B_2)$ $\ldots U_r(B_m)$ не пересекаются и лежат в V_n, что влечет неравенство $m \cdot S_r(n) \leq 2^n$, которое перепишем в виде $M_r(n) \leq \frac{2^n}{S_r(n)}$.

Для оценки снизу опишем процедуру построения такого кода K. Первое кодовое слово $B_1 \in V_n$ возьмем произвольно. Следующее кодовое слово возьмем из условия $B_2 \in V_n \setminus U_{2r}(B_1)$ для того, чтобы не нарушалось условие $d(K) \geq 2r+1$. Значит, для выбора B_2 запрещенными оказались $S_{2r}(n)$ элементов V_n и т. д. Конечно, запрещенные

элементы могут повторяться. Но мы выбираем наихудший вариант, когда они (по возможности) не пересекаются, а их объединение накрывает все V_n, что приводит к неравенству $mS_{2r}(n) \geq 2^n$, которое перепишем в виде $M_r(n) \geq \frac{2^n}{S_{2r}(n)}$.

Заметим, что доказанная теорема не дает способа построения кода, исправляющего ошибки, но указывает принцип его построения.

3.4. Коды Хэмминга

Рассмотрим коды (Хэмминга), исправляющие одну ошибку типа замещения, и укажем эффективную процедуру кодирования и декодирования для них. В электронных источниках отмечено, что коды Хэмминга впервые были представлены в 1942 г. в статье Р. Фишера (Fisher R. A. The theory of coding in factor to th-r theory).

Пусть рассматриваются двоичные кодовые слова B_j длины n, отличной от степени двойки. Обозначим через s число разрядов в двоичной записи числа n, то есть $2^{s-1} < n < 2^s$, или $s = [log\ n] + 1$.

Разобьем множество $N = \{1, 2, 3 \ldots n\}$ натуральных чисел с верхней границей n на следующие s пересекающихся подмножеств:

$$D_i = \{m \quad | \quad 1 \leq m \leq n,\ m = (m_{s-1}m_{s-2}\ldots m_1m_0)_2,\ \text{где}\ m_i = 1\},$$

то есть множество образуют те числа, в двоичной записи которых

$$m = m_{s-1}2^{s-1} + m_{s-2}2^{s-2} + \ldots + m_1 2^1 + m_0 2^0 \qquad (3.2)$$

разряд 2^i входит с коэффициентом $m_i = 1$, а остальные разряды произвольны (коэффициенты 0 или 1). В частности,

$$D_0 = \{1, 3, 5 \ldots n-1 \text{ или } n\} - \text{все нечетные числа до } n;$$

$$D_1 = \{2, 3, 6, 7 \ldots\} - \text{все числа сравнимые с 2 или 3 по модулю 4;}$$

$$D_2 = \{4, 5, 6, 7, 12, 13, 14, 15 \ldots\} - \text{сравнимые с 4, 5, 6 или 7 по модулю 8;}$$

$$\ldots\ ; \quad D_{s-1} = \{2^{s-1}, 2^{s-1}+1, 2^{s-1}+2 \ldots n\}.$$

В каждом множестве D_j повторяющийся период из 2^j идущих подряд чисел начинается с числа 2^j в виде степени двойки и заканчивается перед числом 2^{j+1} в виде степени двойки.

Определение 3.6. *Кодом Хэмминга* порядка n называется множество всех двоичных слов длины n

$$B = (\alpha_1, \alpha_2 \dots \alpha_n) \in V_n,$$

удовлетворяющих системе сравнений по модулю 2

$$\sum_{j \in D_0} \alpha_j \equiv 0, \sum_{j \in D_1} \alpha_j \equiv 0 \dots \sum_{j \in D_{s-1}} \alpha_j \equiv 0.$$

Уравнение названо сравнением (как это принято в теории чисел), так как суммирование производится относительно операции $\oplus$ сложения в группе $\mathbb{Z}_2 = \{0, 1\}$, где $1 \oplus 1 = 0$.

Теорема 3.5. *Код Хэмминга порядка n допускает 2^{n-s} кодовых слов, где $s = [log\ n] + 1$, и исправляет любую единичную ошибку.*

Доказательство. Распишем систему сравнений из определения кода Хэмминга

$$\begin{cases} \alpha_1 \oplus & (\alpha_3 \oplus \alpha_5 \oplus \alpha_7 \oplus \dots) & \equiv 0 \\ \alpha_2 \oplus & (\alpha_3 \oplus \alpha_6 \oplus \alpha_7 \oplus \dots) & \equiv 0 \\ \alpha_4 \oplus & (\alpha_5 \oplus \alpha_6 \oplus \alpha_7 \oplus \dots) & \equiv 0 \\ \dots & & \\ \alpha_{2^{s-1}} \oplus & (\alpha_{2^{s-1}+1} \oplus \alpha_{2^{s-1}+2} \oplus \dots \oplus \alpha_n) & \equiv 0 \end{cases}$$

Согласно свойствам операции $a \oplus a = 0$ для $a \in \mathbb{Z}_2 = \{0, 1\}$ перенесем отделенные слагаемые вида α_{2^m} в правую часть

$$\begin{cases} \alpha_3 \oplus \alpha_5 \oplus \alpha_7 \oplus \dots & \equiv & \alpha_1 \\ \alpha_3 \oplus \alpha_6 \oplus \alpha_7 \oplus \dots & \equiv & \alpha_2 \\ \alpha_5 \oplus \alpha_6 \oplus \alpha_7 \oplus \dots & \equiv & \alpha_4 \\ \dots & & \\ \alpha_{2^{s-1}+1} \oplus \alpha_{2^{s-1}+2} \oplus \dots \oplus \alpha_n & \equiv & \alpha_{2^{s-1}} \end{cases}$$

Получили систему вычисления *проверочных* координат кодовых слов, номера которых есть степени двойки, через *информационные* координаты, номера которых отличны от степени двойки. Число проверочных координат равно s. Все проверочные координаты оказались в правой части сравнений по модулю 2. Все информационные переменные собраны в левой части уравнений, их число $n - s$. Исходные

сообщения могут иметь различный вид. Если же считать их словами длины $n-s$ в алфавите $\{0,1\}$, все возможные такие сообщения составляют слова, буквы которых есть упорядоченные в естественном порядке информационные координаты $\alpha_3, \alpha_5, \alpha_6, \alpha_7, \alpha_9 \ldots$ Всего таких слов 2^{n-s}, где s находится из условия $2^{s-1} < n < 2^s$, равносильного $s-1 < \log\ n < s$.

Допустим, что при передаче кодового слова $B_j = (\alpha_1, \alpha_2 \ldots \alpha_n)$ в разряде с номером d произошла ошибка типа замещения, в результате которой было получено сообщение $B = (\beta_1, \beta_2 \ldots \beta_n)$. Тогда $\beta_i = \alpha_i$ при $i \neq d$, $\beta_d = \alpha_d \oplus 1$. Требуется найти d, двоичное разложение которого называется *синдром*.

Обозначим $\delta_0 \equiv \sum_{j \in D_0} \beta_j$, $\delta_1 \equiv \sum_{j \in D_1} \beta_j \ldots \delta_{s-1} \equiv \sum_{j \in D_{s-1}} \beta_j$, а двоичную запись (3.2) числа $d = (\gamma_{s-1}\gamma_{s-2} \ldots \gamma_1\gamma_0)_2$.

Если $\gamma_i = 0$, то $d \notin D_i$, что означает $\delta_i = \sum_{j \in D_i} \beta_j = \sum_{j \in D_i} \alpha_j \equiv 0$. Если же $\gamma_i = 1$, то $d \in D_i$, откуда $\delta_i = \sum_{j \in D_i} \beta_j = \sum_{j \in D_i} \alpha_j \oplus 1 \equiv 0 \oplus 1$. Получили $d = (\delta_{s-1}\delta_{s-2} \ldots \delta_1\delta_0)_2$.

Итак, код Хэмминга является систематическим $(n, n-s)$-кодом.

Пример 3.1. *Построить код Хэмминга порядка 7.*

Поскольку $2^2 < 7 < 2^3$, то $s = 3$. В коде Хэмминга порядка 7 допускается $16 = 2^{7-3}$ кодовых слов.

Для кодового слова $B_j = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7)$ его разряды $\{\alpha_1, \alpha_2, \alpha_4\}$ проверочные, а разряды $\{\alpha_3, \alpha_5, \alpha_6, \alpha_7\}$ информационные. Составим проверочные множества $D_0 = \{1, 3, 5, 7\}$, $D_1 = \{2, 3, 6, 7\}$ и $D_2 = \{4, 5, 6, 7\}$, по которым построим систему сравнений

$$\begin{cases} \alpha_1 \oplus \alpha_3 \oplus \alpha_5 \oplus \alpha_7 & \equiv 0 \\ \alpha_2 \oplus \alpha_3 \oplus \alpha_6 \oplus \alpha_7 & \equiv 0 \\ \alpha_4 \oplus \alpha_5 \oplus \alpha_6 \oplus \alpha_7 & \equiv 0 \end{cases} \tag{3.3}$$

Можно перечислить все 16 возможных информационных наборов в качестве аргументов для вычисления через систему (3.3) проверочных значений и составления всех кодовых слов (табл. 6). Четыре первые колонки отведены для информационных координат, упорядоченных в стандартном виде, следующие три колонки вычислены по формулам (3.3). Код составлен из этих упорядоченных координат.

Построенный код Хэмминга есть систематический $(7, 4, 3)$-код, что проверяется вычислением $d(K) = 3$.

Таблица 6

Процесс составления кода Хэмминга порядка 7

α_3	α_5	α_6	α_7	α_1	α_2	α_4	К	о	д				
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	0	1	0	0	1
0	0	1	0	0	1	1	0	1	0	1	0	1	0
0	0	1	1	1	0	0	1	0	0	0	0	1	1
0	1	0	0	1	0	1	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0	1	0	1
0	1	1	0	1	1	0	1	1	0	0	1	1	0
0	1	1	1	0	0	1	0	0	0	1	1	1	1
1	0	0	0	1	1	0	1	1	1	0	0	0	0
1	0	0	1	0	0	1	0	0	1	1	0	0	1
1	0	1	0	1	0	1	1	0	1	1	0	1	0
1	0	1	1	0	1	0	0	1	1	0	0	1	1
1	1	0	0	0	1	1	0	1	1	1	1	0	0
1	1	0	1	1	0	0	1	0	1	0	1	0	1
1	1	1	0	0	0	0	0	0	1	0	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1

Пример 3.2. *Найти и исправить одиночную ошибку и декодировать, если получено сообщение* $B = (0011101)$ *в коде Хэмминга порядка 7.*

Решение. Вычисляем для $(\beta_1\beta_2\beta_3\beta_4\beta_5\beta_6\beta_7) = (0011101)$ показатели синдрома:

$$\delta_0 = \beta_1 \oplus \beta_3 \oplus \beta_5 \oplus \beta_7 = 0 \oplus 1 \oplus 1 \oplus 1 = 1,$$

$$\delta_1 = \beta_2 \oplus \beta_3 \oplus \beta_6 \oplus \beta_7 = 0 \oplus 1 \oplus 0 \oplus 1 = 0,$$

$$\delta_2 = \beta_4 \oplus \beta_5 \oplus \beta_6 \oplus \beta_7 = 1 \oplus 1 \oplus 0 \oplus 1 = 1.$$

Получили значение синдрома $d = (\delta_2\delta_1\delta_0)_2 = (101)_2 = 5$, указывающее на то, что ошибка в пятом разряде, а исправленное кодовое слово $B_j = (0011001)$ кодирует исходное сообщение (1001), записанное в информационных разрядах.

3.5. Линейные коды. Порождающая и проверочная матрицы

Описанные коды относятся к классу *линейных* блоковых кодов, когда выбирается вариант равномерного кодирования и наиболее простой вариант формирования множества кодовых слов длины n в терминах линейных пространств.

Заметим, что V_n есть n-мерное векторное пространство F_2^n в виде декартовой степени поля $F_2 = \{0, 1\}$ из двух элементов. Тогда в качестве множества кодовых слов K берем его k-мерное подпространство, которое определяется через базис, состоящий из k линейно независимых (над полем F_2) векторов из V_n. Если k базисных векторов, порождающих подпространство $K \in V_n$, построчно записать в матрицу G, то процедура формирования кодовых слов выполняется в виде умножения матриц $C \cdot G = K$, где матрица C размера $2^k \times k$ содержит полный набор всех различных слов длиной k в алфавите $\{0, 1\}$. Поэтому матрица G называется *порождающей* матрицей кода K, состоящего из 2^k кодовых слов. Такой код называется *линейным* (n, k)*-кодом*. При этом строки матрицы C и будут исходными кодируемыми сообщениями.

Ортогональным дополнением $K^\perp$ к векторному подпространству K служит $(n-k)$-мерное подпространство, которое также может быть задано базисом из $n-k$ линейно независимых элементов V_n, построчно записанных в матрицу H, которая называется *проверочной* матрицей.

Теорема 3.6. *Поскольку порождающая матрица G ортогональна проверочной H, что в матричном виде $G \cdot H^T = 0$, то все кодовые слова ортогональны над полем F_2 строкам проверочной матрицы H.*

Доказательство. Условие ортогональности кодовых слов K проверочной матрице вычислим в матричном виде

$$K \cdot H^T = C \cdot G \cdot H^T = C \cdot 0 = 0.$$

Переданное сообщение имеет вид $B = B_j \oplus \epsilon$ суммы кодового слова B_j и ошибки ϵ. Согласно линейности при проверке полученного сообщения

$$B \cdot H^T = B_j \cdot H^T \oplus \epsilon \cdot H^T = 0 \oplus \epsilon \cdot H^T = \epsilon \cdot H^T$$

получаем значение проверки на возможной ошибке. Поэтому на всех возможных ошибках проверочная матрица должна давать 1, что со-

ответствует условию их обнаружения. Если речь идет о единичных ошибках, то в качестве таковых рассматриваются все векторы стандартного базиса.

Поясним данные понятия на тех примерах, что были рассмотрены в предыдущих параграфах.

Пример 3.3. *Код проверки на четность для случая* $n = 4$.

Все множество V_4 из 16 элементов распадается на подмножества K_0 и K_1 по восемь элементов с четной и нечетной суммой координат. Нулевой элемент попал в K_0, которое можно представить в виде векторного пространства с базисом из трех векторов. В качестве базиса можно взять векторы (1100), (0110) и (0011), которые и составят порождающую матрицу

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

Поскольку у матрицы G три строки (подпространство кодовых векторов трехмерно), то все возможные коэффициенты линейных комбинаций можно собрать (и упорядочить в лексикографическом порядке) в матрице

$$C = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}. \tag{3.4}$$

Тогда множество кодовых слов (3.1) будут составлять строки следующей матрицы K, где умножение матриц проводим по правилам поля F_2, которые можно упростить так: каждая j-я строка произведения есть линейная комбинация строк второго сомножителя с коэффициентами, указанными в j-й строке первого сомножителя. По этому правилу строка с номером 0 будет нулевой, а строки с номерами 1 и 2 соответственно равны последней и средней строкам матрицы G, строка с номером 3 есть сумма $\oplus$ этих строк и т. д.

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

В качестве проверочной матрицы возьмем

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

Все кодовые слова ортогональны над полем F_2 данной проверочной строке H. Для установления этого факта достаточно было проверить, что строки матрицы G ортогональны строке H, или в матричном виде $G \cdot H^T = 0$. Все одиночные ошибки ϵ – это векторы стандартного базиса, которые при проверке с данной строкой H дают $\epsilon \cdot H^T = 1$. А вот две ошибки данной проверочной матрицей не выявляются, поскольку $d(K) = 2$. Таким образом, данный код проверки на четность является $(4, 3, 2)$-кодом, так как $n = 4$, $k = 3$, $d = 2$.

Замечание 1. Приведенный линейный код несистематический.

Информационными столбцами в матрице $K = C \cdot G$ служат первый и последний столбцы, совпадающие с первым и последним столбцами матрицы C. Но для среднего столбца матрицы C нет такого же столбца в матрице $K = C \cdot G$. Поэтому в данном примере нет полного подразделения на информационные и проверочные переменные. Это наблюдение легко устанавливается по виду порождающей матрицы: в случае систематического линейного кода в порождающей матрице есть полный набор столбцов стандартного базиса.

Пример 3.4. *Систематический код проверки на четность для случая* $n = 4$.

В качестве порождающей матрицы предложим другую матрицу

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

и получим те же самые кодовые слова (но в другом порядке), в которых первые три координаты будут информационными, а последняя координата – проверочной. Проверочная матрица H остается прежней.

Замечание 2. Пространства K и $K^{\perp}$, базисы которых записаны в порождающей G и проверочной H матрицах, взаимно перпендикулярны и являются подпространствами различных вариантов пространства V_n. Если $K \subset V_n$, то $K^{\perp} \subset V_n^*$ берется в сопряженном пространстве. Но в данном случае сопряженное пространство V_n^* изоморфно V_n, что записывается в виде равенства пространств $V_n^* = V_n$. По этой причине пространства K и $K^{\perp}$, помещенные в одно пространство, могут пересекаться, а в некоторых случаях даже совпадать. В частности, в примере 3.3 есть кодовое слово (1111), совпадающее с H и ортогональное ему.

Согласно сделанному замечанию взаимно ортогональные пространства K и $K^{\perp}$ можно поменять местами, поменяв при этом местами порождающую и проверочную матрицы.

Пример 3.5. *Линейный код с порождающей матрицей* $G = (1111)$.

Если поменяем местами порождающую и проверочную матрицы примера 3.3, то получим проверочную матрицу

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

Для одной строки возможны всего две линейные комбинации, а именно с коэффициентами 0 и 1, что дает кодовые слова $B_0 = (0000)$ и $B_1 = (1111)$. Любую из координат (например, первую) можно считать информационной, а остальные координаты – проверочными. Три скалярных произведения полученного сигнала B с каждой из строк проверочной матрицы H составляют синдром полученного сигнала. Перебором возможных одиночных ошибок составляют таблицу их синдромов. Если бы в качестве проверочной H взяли матрицу из примера 3.4 (а не из 3.3), то синдром совпадал бы с номером ошибочного разряда. Поскольку кодовое расстояние $d(K) = 4$, то код (по теореме 3.1) способен обнаружить три ошибки, когда координаты полученного сообщения B будут различны; код (по теореме 3.3) способен исправить

лишь одну ошибку (в случае двух ошибок полученное сообщение B будет на расстоянии 2 от каждого кодового слова). Этот код типа $(4, 1, 4)$ называется кодом учетверения символа. Повышенная избыточность этого кода характеризуется низкой скоростью $R = \frac{1}{4}$.

Пример 3.6. *Метод кодирования удвоением символов.*

Рассмотрим код, при котором длина сигнала удваивается, примененный для случая исходных сигналов длины два (таких сигналов четыре). Получим табл. 7 кодируемых сигналов и кодовых слов.

Таблица 7

Таблица для метода удвоения символов

Сигнал	Кодовое слово
00	0000
01	0011
10	1100
11	1111

Ставится задача найти порождающую и проверочную матрицы.

Порождающая матрица состоит из двух строк, так как исходные сигналы двумерные $k = 2$. Проверочная матрица тоже состоит из двух строк, так как $n = 4$ – длина кодовых слов, а $n - k = 2$.

В качестве порождающей матрицы можно предложить

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

при этом информационными объявим первую и третью координаты (им соответствуют векторы стандартного базиса), тогда проверочными будут вторая и четвертая координаты.

В качестве проверочной матрицы H можно взять ту же матрицу G, то есть $H = G$. Первая строка H проверяет совпадение первых двух координат, а вторая строка проверяет совпадение последних двух координат. Это код типа $(4, 2, 2)$.

Этот код способен находить единичные ошибки, но не может их исправлять. Его скорость $R = \frac{1}{2}$.

Аналогичный код утроения отдельных символов способен уже исправлять ошибки. Его скорость $R = \frac{1}{3}$.

Пример 3.7. *Матричное задание кода Хэмминга порядка 7.*
Порождающая матрица для примера 3.1 будет

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

которую получили следующим образом. Поскольку координаты с номерами 3, 5, 6 и 7 информационные, то столбцы с этими номерами базисные и им соответствуют столбцы стандартного базиса. По формулам системы (3.3) проверочные переменные с номерами 1, 2 и 4 выражаются через базисные. По тем же формулам соответствующие им столбцы выражаются через базисные столбцы. Например, первая формула системы в виде $\alpha_1 = \alpha_3 \oplus \alpha_5 \oplus \alpha_7$ приводит к виду первого столбца как суммы столбцов с номерами 3, 5 и 7.

Произведение матрицы C в виде левой части табл. 6 на порождающую матрицу G дает все 16 кодовых слов, записанных в правой части табл. 6 в том же порядке.

Проверочная матрица H размера 3×7 (где $n-k = 7-4 = 3$ – число строк) полностью соответствует формулам проверки (3.3), поскольку в строках указаны коэффициенты этих проверочных формул

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

Поскольку в столбцах проверочной матрицы H записаны в лексикографическом порядке (3.2) двоичные коды чисел от 1 до 7, то это доказывает то, что синдром единичной ошибки кода Хэмминга $\epsilon \cdot H^T$ равен ошибочному разряду, записанному в двоичной системе.

3.6. Коды Уолша – Адамара

Для передачи информации по неустойчивому каналу связи применяют коды Уолша – Адамара типа $(2^k, k)$.

Множество кодовых слов этого кода совпадает с множеством дискретных функций Уолша в аддитивной записи. Поясним варианты

аддитивной и мультипликативной записей. Группу из двух элементов $\mathbb{Z}_2 = \{0, 1\}$ обычно рассматривают относительно операции $\oplus$ сложения по модулю 2. Действия в группе: $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$. Это аддитивная форма представления этой группы. Та же группа допускает мультипликативную форму представления $\mathbb{Z}_2 = \{1, -1\}$ с операцией умножения $\cdot$ в группе. Переход от аддитивной формы к мультипликативной форме осуществляется перекодировкой $0 \to 1$, $1 \to -1$, $\oplus \to \cdot$.

Векторное пространство $\mathbb{Z}_2^N$, состоящее из N-мерных булевых векторов (с элементами 0 и 1) традиционно рассматривают в аддитивной форме. Операция сложения в этом пространстве обозначается прежним символом $\oplus$, но понимается как покоординатное сложение по модулю 2. Коэффициенты линейных комбинаций берутся из поля $F_2 = \{0, 1\}$ и трактуются как выбор элемента (при коэффициенте 1) или отказ в выборе (при коэффициенте 0).

Кроме этого аддитивного представления векторного пространства $\mathbb{Z}_2^N$ возможно его мультипликативное представление с элементами в виде N-мерных векторов с координатами 1 и -1. Операция $\oplus$ в пространстве переходит в операцию $\bullet$ умножения по Адамару (покоординатное умножение векторов).

Все дискретные фукции Уолша уровня n (порядка $N = 2^n$) записаны в строках матрицы Сильвестра – Адамара $H_n = H^{n\otimes}$, полученной в виде n-й кронекеровой степени матрицы $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

Определение кронекерова произведения поясним примером с матрицей $D = \begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix}$

$$D \otimes H = \begin{pmatrix} H & 2H \\ 4H & 3H \end{pmatrix} = \begin{pmatrix} 1 & 1 & 2 & 2 \\ 1 & -1 & 2 & -2 \\ 4 & 4 & 3 & 3 \\ 4 & -4 & 3 & -3 \end{pmatrix},$$

где выписана блочная матрица с блоками в виде элемента первой матрицы, умноженного на вторую матрицу.

По этому определению получаем рекуррентную формулу вычисления матриц Сильвестра – Адамара

$$H_{n+1} = \begin{pmatrix} H_n & H_n \\ H_n & -H_n \end{pmatrix}.$$

Примеры этих матриц

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

$$H_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}.$$

Если произвести обратную перекодировку $1 \to 0$, $-1 \to 1$ из мультипликативной формы в аддитивную, то в строках матрицы получим все кодовые слова кода Уолша – Адамара выбранного уровня. Например, при $n = 2$ строки *аддитивной записи матрицы Уолша – Адамара*

$$A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

составляют набор кодовых слов кода Уолша – Адамара уровня 2.

Рассмотрим способ непосредственного вычисления матриц A_n без обращения к матрицам Сильвестра – Адамара H_n.

Рекуррентно определяются матрицы G_m размера $m \times 2^m$

$$G_1 = (0\ 1), \quad G_m = \begin{pmatrix} \mathbf{0} & \mathbf{1} \\ G_{m-1} & G_{m-1} \end{pmatrix}, \tag{3.5}$$

где $\mathbf{0} = (00\ldots0)$, $\mathbf{1} = (11\ldots1)$ – постоянные векторы соответствующей длины. В частности, матрица (3.4) есть транспонированная G_3.

По правилам действия в поле $F_2 = \{0, 1\}$ можно вычислить *аддитивную матрицу Уолша – Адамара* как произведение матриц

$$A_k = G_k^T \cdot G_k. \tag{3.6}$$

В формуле (3.6) первый множитель (транспонированную матрицу G_k) трактуем как набор всех возможных коэффициентов линейных комбинаций над полем F_2 (ранее обозначаемый как матрица C), а второй сомножитель трактуем как набор образующих, составляющий базис векторного подпространства $K \in V_{2^k}$ кодовых слов (ранее обозначаемый как порождающая матрица G). Таким образом, получилось, что длина кодовых слов $n = 2^k$, а кодируемых сообщений k. Значит, имеем $(2^k, k)$-код. Проверочная матрица H этого кода будет большого размера $(2^k - k) \times 2^k$, поэтому ее не принято использовать.

Основное преимущество этого кода состоит в том, что у него максимально возможное кодовое расстояние $d(K) = 2^{k-1}$. Покажем это.

Функции Уолша служат простейшим примером семейства ортогональных функций. Это свойство переносится и на дискретные функции Уолша, которые ортогональны относительно обычного (в пространстве $\mathbb{R}$) скалярного произведения. Утверждение об ортогональности для нас означает, что любые две различные дискретные функции Уолша на половине координат (на 2^{k-1} координатах) совпадают, а на половине координат (на 2^{k-1} координатах) различаются. Это свойство повторяется и при аддитивной форме записи, что означает $d(K) = 2^{k-1}$.

Итак, код Уолша – Адамара относится к классу $(2^k, k, 2^{k-1})$-кодов. Именно свойство максимальности кодового расстояния выделяет этот код в качестве удобного для зашумленного канала связи.

Например, если взять $k = 10$ (десять), то $d(K) = 512$, что означает возможность обнаружения факта внесения ошибки при искажении практически половины информации и возможность исправления ошибки (без повторного запроса) при искажении почти четверти информации (точнее, 255 ошибок в блоке). Именно это и послужило поводом для использования кода Уолша – Адамара при передаче информации во время космической связи.

Пример 3.8. *Код Уолша – Адамара уровня 3.*

Порождающей матрицей G_3 кода Уолша – Адамара уровня 3 служит транспонированная матрица (3.4), строки которой есть *дискретные функции Радемахера*

$$G_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

По формуле (3.6) получается матрица A_3, которую также получаем перекодировкой $1 \to 0$, $-1 \to 1$ приведенной выше матрицы H_3.

Строки матрицы G_3 составляют базис трехмерного пространства, состоящего из восьми кодовых слов.

Теорема 3.7. *Для $N = 2^n$ множество дискретных функций Уолша есть подмножество элементов мультипликативного представления векторного пространства $\mathbb{Z}_2^N$, изоморфное векторному пространству $\mathbb{Z}_2^n$.*

Доказательство. В примере 3.8 $n = 3$, $N = 8$, а множество дискретных функций Уолша составляют строки матрицы H_3. Эти строки образуют подпространство мультипликативного представления векторного пространства $\mathbb{Z}_2^N$ при $N = 8$. Формулу (3.6) можно переписать в виде

$$A_3 = G_3^T \cdot E \cdot G_3,$$

вставив единичную матрицу E третьего порядка. Строки этой матрицы E составляют стандартный базис векторного пространства $\mathbb{Z}_2^n$ при $n = 3$. Все возможные линейные комбинации строк (над полем F_2) матрицы E заданы набором коэффициентов в матрице G_3^T. Точно так же все возможные линейные комбинации строк матрицы $E \cdot G_3$ заданы набором коэффициентов в матрице G_3^T, что и доказывает изоморфизм аддитивных представлений всего пространства $\mathbb{Z}_2^n$ и n-мерного подпространства $\mathbb{Z}_2^N$ с базисом в виде строк матрицы G_3. Заканчиваем доказательство переводом аддитивного представления n-мерного подпространства $\mathbb{Z}_2^N$ в мультипликативное. $\square$

На практике при космической связи использовалось следующее усовершенствование кода Адамара, известное под названием *дополненного кода Адамара.*

Порождающая матрица дополненного кода Адамара получается добавлением к матрице G_k одной строки $\mathbf{1}$ соответствующей длины. При этом число кодовых слов удвоится (станет равным 2^{k+1}), а их длина сохранится. Дополнительно к кодовым словам в виде строк матрицы A_k добавятся строки матрицы $\overline{A_k}$, которые противоположны исходным строкам, $a \oplus \mathbf{1} = \overline{a}$.

Поскольку строки матрицы H_k ортогональны, то строки матрицы $-H_k$ тоже ортогональны и взаимно ортогональны со строками исходной матрицы H_k. Поскольку матрица $-H_k$ есть аддитивный вариант матрицы $\overline{A_k}$, то кодовое расстояние равно 2^{k-1}. Итак, дополненный код Адамара есть линейный $(2^k, k+1, 2^{k-1})$-код.

Он строится согласно простому наблюдению о том, что множество дискретных функций Уолша, взятых с общим знаком минус, также составляет ортогональный набор и ортогональны исходному набору.

3.7. Коды Рида – Маллера нулевого и первого порядков

Хорошо изученным семейством линейных кодов с избыточностью, позволяющих осуществлять обнаружение и исправление ошибок, служат (предложенные в 1954 г. Дэвидом Э. Маллером) двоичные коды Рида – Маллера (Рида – Мюллера), которые обозначаются $RM(r, m)$. Первый индекс r есть порядок кода. По второму индексу m определяется длина кода, которая равна степени двойки, а именно 2^m.

В примере 3.5 был рассмотрен код $RM(0, 2)$ Рида – Маллера нулевого порядка с порождающей матрицей $G_0(2) = (1111)$. Порождающей матрицей кода $RM(0, 3)$ будет $G_0(3) = (11111111)$ и т. д.

Интерес представляют коды $RM(r, m)$ при $r \geq 1$.

Дополненный код Адамара является кодом Рида – Маллера первого порядка $RM(1, m)$.

Порождающая матрица кода $RM(1, m)$ получается по формуле (3.5) добавлением в начало матрицы G_m строки $\mathbf{1}$. Получим матрицы

$$G_{1,1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \; G_{1,2} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix},$$

$$G_{1,3} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Поскольку по сравнению с кодом Уолша – Адамара число строк порождающей матрицы увеличилось на 1, то набор возможных линейных комбинаций в матрице $C = (G_{m+1})^T$. Если множество кодовых слов кода $RM(1,m)$ обозначим $CRM(1,m)$, то по формуле $K = C \cdot G$

$$CRM(1,m) = (G_{m+1})^T \cdot G_{1,m}.$$

Предложим второй метод построения множества кодовых слов кода $RM(1,m)$. Обозначим $\overline{CRM(1,m)}$ множество противоположных (в смысле перехода к $\overline{a} = a \oplus \mathbf{1}$) слов. Тогда множество кодовых слов следующего уровня $m+1$ в строках следующей блочной матрицы

$$CRM(1,m+1) = \begin{pmatrix} CRM(1,m) & CRM(1,m) \\ CRM(1,m) & \overline{CRM(1,m)} \end{pmatrix}. \qquad (3.7)$$

Например, для $m = 1$ через матрицу возможных линейных комбинаций $C = (G_2)^T$ и порождающую матрицу $G_{1,1}$ вычислим множество кодовых слов

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

По формуле отрицания

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

Отсюда по формуле (3.7)

$$CRM(1,2) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

Существует третий метод вычисления множества кодовых слов через матрицы Сильвестра – Адамара H_m и их аддитивный вариант A_m

$$CRM(1,m) = \begin{pmatrix} \overline{A_m} \\ A_m \end{pmatrix}.$$

В частности, строки матрицы $CRM(1,2)$ повторяют строки следующих матриц:

$$A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}, \quad \overline{A_2} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

Обозначим символом J квадратную матрицу, все элементы которой равны 1. Тогда $\overline{A_m} + A_m = J$, $\overline{A_m} - A_m = H_m$. Обратная взаимосвязь $A_m = \frac{1}{2}(J - H_m)$, $\overline{A_m} = \frac{1}{2}(J + H_m)$. Отсюда

$$H_m = J - 2A_m = 2\overline{A_m} - J, \quad -H_m = 2A_m - J. \tag{3.8}$$

Поскольку кодовое расстояние $RM(1,m)$-кода равно 2^{m-1}, то он способен обнаружить $2^{m-1}-1$ ошибку или исправить $2^{m-2}-1$ ошибок, что служит очень высоким показателем. Плата за это – невысокая скорость передачи информации, равная $\frac{m+1}{2^m}$.

Кроме утверждения о возможности декодирования с исправлением ошибок предложим алгоритм этой процедуры для $RM(1,m)$-кода по методу максимального правдоподобия.

Пусть B – принятое сообщение длины 2^m. Требуется найти B_{j_0} среди кодовых слов $B_j \in CRM(1,m)$ такое, что $d(B, B_{j_0}) = \min d(B, B_j)$.

Обозначим перекодированные (из аддитивной формы в мультипликативную с предварительным отрицанием) сообщения

$$\hat{B} = 2B - \mathbf{1}, \quad \hat{B}_j = 2B_j - \mathbf{1},$$

где координата, равная 0, заменяется на -1. Поскольку кодовые слова размещались в матрицах A_m и $\overline{A_m}$, то согласно (3.8) перекодированные слова попадут в $-H_m$ и в H_m соответственно. Перебор всех кодовых слов B_j в аддитивной форме (строк матриц A_m и $\overline{A}_m$) соответствует перебору строк $\hat{B}_j$ (в мультипликативной форме) матриц $-H_m$ и H_m, что можно совместить в один перебор.

Расстояние Хэмминга как число различающихся координат не зависит от кодировки

$$d(B, B_j) = d(\hat{B}, \hat{B}_j).$$

Обозначим $N = 2^m$ длину кодовых слов, N_0 – число совпадающих координат, а $N_1 = d(B, B_j)$ – число различающихся координат. Введем также обычное скалярное произведение векторов $\hat{B}$ и $\hat{B}_j$, обозначенное $< \hat{B}, \hat{B}_j >$. Тогда (произведение различающихся координат равно -1)

$$N = N_0 + N_1, \; < \hat{B}, \hat{B}_j >= N_0 - N_1 = N - 2d(\hat{B}, \hat{B}_j).$$

При этом минимум расстояния будет соответствовать максимуму скалярного произведения, поскольку

$$\min d(\hat{B}, \hat{B}_j) = \min \frac{1}{2}(N - < \hat{B}, \hat{B}_j >) = \frac{1}{2}(N - \max < \hat{B}, \hat{B}_j >).$$

Алгоритм поиска и исправления ошибок

1. В качестве синдрома вычисляем дискретное преобразование Уолша от сигнала $\hat{B}$: $\quad \delta = H_m \cdot \hat{B}$.

2. Находим максимальную по модулю координату полученного вектора δ.

3. Если эта координата положительна, то ближайшее кодовое слово ищем в матрице $\overline{A_m}$; если же эта координата отрицательна, то ближайшее кодовое слово ищем в матрице A_m.

4. Номер (нумерация с нуля) этой максимальной по модулю координаты есть номер ближайшего кодового слова, которое и принимается за передаваемое сообщение, в выбранной матрице A_m или $\overline{A_m}$.

Кроме алгоритма декодирования также представляет интерес алгоритм восстановления исходного сообщения $x = (x_0, x_1 \dots x_{m-1})$ по кодовому слову $y \in CRM(1, m)$.

Информационными координатами служат y_0, y_1, y_2, $y_4 \dots y_{2^{m-1}}$.

Алгоритм декодирования, поиска исходного сообщения по кодовому слову

1. Полагаем $x_0 := y_0$.

2. Для i от 0 до $m-1$ вычисляем $x_{m-i} = x_0 \oplus y_{2^i}$.

Доказательство этого алгоритма основывается на том факте, что координаты с номерами в виде степеней двойки однозначно определяют вид дискретной функции Уолша (как в мультипликативной, так и в аддитивной форме). Дополнительно значение y_0 определяет принадлежность сигнала y к той или иной части матрицы $CRM(1, m)$: если $y_0 = 0$, то $y \in A_m$; если $y_0 = 1$, то $y \in \overline{A_m}$.

Пример 3.9. *Для полученного сообщения* $B = (11100011)$ *найти ближайшее кодовое слово кода* $RM(1, 3)$ *и исходное сообщение.*

1. Вычисляем синдром этого сигнала $\hat{B}$ по формуле $\delta = H_3 \cdot \hat{B} =$

$$= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ -2 \\ -2 \\ 2 \\ 2 \\ 6 \\ -2 \end{pmatrix}.$$

2. Предпоследняя координата (равная 6) максимальна по модулю.

3. Поскольку эта координата положительна ($6 > 0$), то ближайшее кодовое слово в матрице $\overline{A_3}$.

4. Выбираем предпоследнее (с номером 6) слово в матрице $\overline{A_3}$ вида $y = (11000011)$.

Сравнением слов y и B отмечаем, что мы исправили одну ошибку. Действительно, кодовое расстояние $RM(1, 3)$ кода равно $2^{m-1} = 4$. Этот алгоритм способен обнаружить три ошибки, но исправляет только одну ошибку.

На вход следующего алгоритма подается слово $y = (11000011)$. Восстановим исходное сообщение.

1. Поскольку $y_0 = 1$, то $x_0 = 1$.

2. Координаты $y_1 = 1$, $y_2 = 0$, $y_4 = 0$ с номерами в виде степени двойки позволяют вычислить остальные координаты сообщения в обратном порядке

$x_3 = x_0 \oplus y_1 = 1 \oplus 1 = 0$, $x_2 = x_0 \oplus y_2 = 1 \oplus 0 = 1$,$x_1 = x_0 \oplus y_4 = 1 \oplus 0 = 1$.

Итак, восстановили сообщение $x = (1\ 1\ 1\ 0)$.

3.8. Коды Рида – Маллера r-го порядка

Начнем с кодов Рида – Маллера второго порядка $RM(2, m)$. Для сравнения, для кода первого порядка $RM(1, m)$ в порождающей матрице выделяем две группы строк в количестве $1 = C_m^0$ и $m = C_m^1$, нумеруя группы верхним индексом числа сочетаний с нуля. Нулевая группа состоит из одного постоянного слова $v_0 = \mathbf{1} = (1\ 1\ 1 \ldots 1\ 1)$. Первая группа – из m линейно независимых над полем F_2 максимально друг от друга отстоящих в метрике Хэмминга строк v_1, $v_2 \ldots v_m$. В порождающую матрицу G кода Рида – Маллера второго порядка добавляется новая (вторая) группа образующих строк в количестве C_m^2, состоящая из возможных произведений по Адамару пар строк первой группы v_1, $v_2 \ldots v_m$.

Пример 3.10. *Построить порождающую и проверочную матрицы кода Рида – Маллера второго порядка* $RM(2, 3)$.

Дополним матрицу $G_{1,3}$ тремя строками $v_1 \bullet v_2$, $v_1 \bullet v_3$, $v_2 \bullet v_3$

$$G_{2,3} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Поскольку в матрице $G = G_{2,3}$ семь линейно независимых строк, то множество кодовых слов состоит из $2^7 = 128$ слов длиной 8. Всего слов длиной 8 в два раза больше $2^8 - 256$. Все слова порождающей

матрицы G содержат четное число единиц. Поэтому и все кодовые слова (как линейные комбинации этих образующих слов) с четным числом единиц. Все слова длиной 8 делятся на две равные по объему (в 128 слов) группы слов с четным и нечетным числом единиц.

Таким образом, установили, что код $RM(2,3)$ есть код проверки на четность слов длиной 8. Проверочная матрица этого кода состоит из одной строки $H = (11111111)$.

Порождающая матрица кода Рида – Маллера третьего порядка при $m > 3$ получается из кодовой матрицы $G_{2,m}$ добавлением третьей группы образующих строк в количестве C_m^3, состоящих из возможных произведений по Адамару трех различных строк первой группы v_1, v_2 $\ldots v_m$.

По этому правилу порождающая матрица кода $G_{3,3}$ получается присоединением (в качестве последней строки) к матрице $G_{2,3}$ строки (00000001), что приводит к совпадению множества кодовых слов со множеством всех слов длиной 8. По этой причине в формулировку добавлено ограничение $m > 3$.

По данному принципу процесс построения кодов Рида – Маллера более высокого порядка можно продолжить.

Количество всех строк порождающей матрицы кода $RM(r,m)$ равно $k = C_m^0 + C_m^1 + \ldots + C_m^r$, что приводит к числу кодовых слов 2^k. Длина каждого кодового слова $n = 2^m$. Число строк проверочной матрицы $n - k = C_m^{r+1} + C_m^{r+2} + \ldots + C_m^m$.

Кодовое расстояние кода $RM(r,m)$ равно 2^{m-r}. Значит, может быть обнаружена $2^{m-r} - 1$ ошибка, а исправлена $2^{m-r-1} - 1$ ошибка. Например, код $RM(2,4)$ имеет кодовое расстояние 4, обнаруживает три ошибки, а исправляет одну ошибку.

Вопросы для самопроверки

1. Приведите определение расстояния Хэмминга.
2. Какой код называется кодом, обнаруживающим одну ошибку?
3. Какой код называется кодом, обнаруживающим две ошибки?
4. Какой код называется кодом, исправляющим одну ошибку?
5. Если кодовое растояние равно 3, то сколько ошибок способен обнаружить код? Можно ли этот код рассматривать в качестве кода, исправляющего ошибки?

6. К какому типу кодов – обнаруживающих или исправляющих ошибки – относится код проверки на четность?

7. Сколько ошибок способен исправить код Хэмминга?

8. В линейном блоковом (n, k)-коде поясните значение символа n и символа k.

9. Как с помощью порождающей матрицы построить множество всех кодовых слов кода?

10. Какой линейный код называется систематическим?

11. Для чего применяется проверочная матрица: для обнаружения или исправления ошибок?

12. Какого типа линейным кодом служит код Уолша – Адамара?

Упражнения

3.1. Постройте код Хэмминга порядка 6.

3.2. Найдите и исправьте одиночную ошибку, если получено сообщение $B = (0011101)$ в коде Хэмминга порядка 6.

3.3. Какие координаты являются информационными в коде Хэмминга порядка 6? Декодируйте сообщение с одной ошибкой из предыдущего упражнения.

3.4. Составьте порождающую матрицу кода Хэмминга порядка 6.

3.5. Составьте проверочную матрицу кода Хэмминга порядка 6.

3.6. Составьте матрицу Уолша – Адамара A_3.

3.7. Продолжите матрицу A_3 до дополненного кода Адамара матрицей $\overline{A_3}$.

3.8. Составьте порождающие матрицы для кодов из двух предыдущих упражнений. Вычислите кодовые расстояния этих кодов. Сколько ошибок способен обнаружить и сколько исправить каждый из этих кодов? Сколько кодовых слов в каждом из этих кодов?

3.9. Постройте код Рида – Маллера первого порядка RM(1,3).

3.10. Если при кодировании кодом Рида – Маллера RM(1,3) в виде дополненного кода Адамара получено сообщение $B = (00011100)$, то найдите и исправьте ошибку, а также декодируйте сообщение.

3.11. Если при кодировании кодом Рида – Маллера RM(2,3) получены сообщения $B = (00011100)$ и $C = (00010000)$, то какое из них с ошибкой?

Глава 4

Элементы криптографии

Часто кодирование совмещается с шифрованием информации. Основная цель кодирования – облегчить пересылку информации, которая сейчас осуществляется с привлечением компьютеров. Поэтому информация и конвертируется в двоичные коды, удобные и понятные компьютеру. Используется общедоступный канал пересылки информации. В связи с этим после кодирования информацию следует защитить от воздействий при передаче информации. В предыдущем разделе рассмотрены методы борьбы со случайными помехами.

Кроме случайных помех возможно преднамеренное искажение информации сторонними лицами. Задачи защиты информации решает *криптография*. Криптографы используют термин "кодирование" в несколько ином смысле – как процесс преобразования текста путем замены одних слов другими. Процедурой дешифровки перехваченных сообщений занимаются криптоаналитики.

Первоначально в криптографии использовали табличные методы кодирования букв алфавита теми же самыми буквами, но в другом порядке. Таким образом, таблица выступала в качестве ключа кодирования. Криптоаналитики составили частоты появления букв, что позволило с привлечением компьютера вычислить алгоритм декодирования для данного метода шифрования.

В дальнейшем был сформулирован основной принцип криптографической системы. Процедура кодирования $\psi : A^* \to B^*$ исходных сообщений в виде слов выходного алфавита (который теперь не обязательно двоичный) должна быть достаточно простой, а процедура декодирования $\psi^{-1} : B^* \to A^*$ должна составлять сложную математическую задачу.

Поэтому для построения реальной криптографической системы потребуется серьезный математический аппарат.

В 1976 г. молодые американские ученые У. Диффи и М. Хеллман предложили способ обмена зашифрованными сообщениями без обмена секретными ключами. Этот метод использует модульную арифметику,

а также свойства простых чисел, о которых расскажем далее.

На базе алгоритма Диффи – Хеллмана в 1977 г. Рон Риверст, Ади Шамир и Лен Адлеман предложили новый криптографический метод, известный теперь под названием RSA (по первым буквам фамилий авторов) и являющийся стандартом для электронной подписи.

4.1. Определения и утверждения из теории чисел

Напомним, что *простым* числом p называется число, делящееся только на 1 и на себя. Число 1 простым не считают и отделяют как особое число. Остальные натуральные числа являются *составными.*

Натуральные числа a и b называются *взаимно простыми*, если у них нет общего делителя, кроме единицы (тривиального). Поскольку в теории чисел наибольший общий делитель чисел a и b обозначается (a, b), то для взаимно простых $(a, b) = 1$.

В теории чисел работают с целыми числами. *Факторизация* числа N есть задача разложения этого числа на простые множители (это разложение единственно)

$$N = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}. \tag{4.1}$$

Пусть $a, b, q \in \mathbb{N}$ (натуральные). Если выполнено $a = bq$, то говорят: a кратно b и b делит a, и обозначают $b|a$.

Если a не кратно b, то деление целых чисел проводится с остатком: из равенства $a = bq + c$, где $0 \leq c < b$ – остаток от деления a на b, получаем формулу *сравнения* $a \equiv c \ (mod\ b)$. При этом a, q могут быть и отрицательными. Понятие *сравнение по модулю* b можно расширить на целые числа a, d, полагая сравнимыми $a \equiv d \ (mod\ b)$, если $a - d = bq$.

Целое положительное m, остатки от деления на которое сравниваем, называется *модулем.*

Если зафиксируем m модуль сравнения, то указание $(mod\ m)$ можно опустить. Свойства сравнений с фиксированным модулем:

1) если $a \equiv b$, $b \equiv c$, то $a \equiv c$;

2) если $a \equiv b$, $c \equiv d$, то $a + c \equiv b + d$;

3) если $a \equiv b$, $c \equiv d$, то $ac \equiv bd$.

Свойства сравнений с разными модулями:

4) если $a \equiv b \ (\ mod\ m)$, то $ak \equiv bk \ (mod\ mk)$;

5) если $ak \equiv bk\ (mod\ mk)$, то $a \equiv b\ (mod\ m)$;

6) если $a \equiv b\ (mod\ m_1)$ и $a \equiv b\ (mod\ m_2)$, то $a \equiv b\ (mod\ m)$, где m – наименьшее общее кратное m_1 и m_2.

Множество сравнимых по модулю m целых чисел образуют класс вычетов по модулю m. Основной представитель c этого класса удобнее выбрать из условия $0 \leq c < m$ и обозначить класс $\mathbf{c}$. Тогда *полная система вычетов* состоит из m классов: $\mathbf{0}, \mathbf{1} \ldots \mathbf{m-1}$.

Определение 4.1. *Функция Эйлера* $\varphi(n)$ натурального n $(n > 1)$ равна числу натуральных k таких, что $k < n$ и $(n, k) = 1$.

Пример 4.1. *Функция Эйлера начальных чисел.*

В дополнение к определению 4.1 полагаем $\varphi(1) = 1$. По определению 4.1 легко вычислить

$$\varphi(2) = 1, \varphi(3) = 2, \varphi(4) = 2, \varphi(5) = 4, \varphi(6) = 2, \varphi(7) = 6.$$

Теорема 4.1. *Через каноническое разложение (4.1) функция Эйлера вычисляется*

$$\varphi(N) = N(1 - \frac{1}{p_1})(1 - \frac{1}{p_2}) \ldots (1 - \frac{1}{p_k}),$$

что равносильно

$$\varphi(N) = (p_1^{a_1} - p_1^{a_1 - 1})(p_2^{a_2} - p_2^{a_2 - 1}) \ldots (p_k^{a_k} - p_k^{a_k - 1}).$$

Если $(M, N) = 1$, *то* $\varphi(M \cdot N) = \varphi(M) \cdot \varphi(N)$.

Следствие 4.1. *Имеем* $\varphi(p) = p - 1$, $\varphi(p^k) = p^k - p^{k-1}$ *в случае простого* p.

Пример 4.2. *Функция Эйлера для отдельных чисел.*

По следствию $\varphi(11) = 10$, $\varphi(81) = 81 - 27 = 54$. По теореме

$$\varphi(60) = 60(1 - \frac{1}{2})(1 - \frac{1}{3})(1 - \frac{1}{5}) = (4 - 2)(3 - 1)(5 - 1) = 16,$$

$$\varphi(660) = \varphi(60) \cdot \varphi(11) = 160.$$

Теорема 4.2 (малая теорема Ферма) *Если* p *– простое число, то для любого натурального* a *справедливо*

$$a^p \equiv a\ (mod\ p).$$

Доказательство. По биному Ньютона, где отделили первое и последнее слагаемые,

$$(a+1)^p = a^p + \sum_{k=1}^{p-1} C_p^k a^k + 1.$$

Поскольку p – простое, то все биномиальные коэффициенты C_p^k (записанные в сумме) делятся на p. Значит, и вся сумма делится на p. Получили сравнение $(a+1)^p \equiv a^p + 1 \ (mod\ p)$. Подставляя вместо a последовательно $0, 1, 2 \ldots p-2$ получим цепочку сравнений

$$2^p \equiv 2,\ 3^p \equiv 3,\ \ldots,\ (p-1)^p \equiv p-1,$$

которую можно и продолжить. Для a, кратного p, теорема также верна. Для остальных положительных a утверждение теоремы верно по свойствам сравнений.

Следствие 4.2. *Для любого натурального $a < p$, где p – простое, верно $a^{p-1} \equiv 1$.*

Малая теорема Ферма в курсе теории чисел обычно рассматривается как следствие следующей теоремы.

Теорема 4.3 (Эйлера) *Если $m > 1$ и $(a, m) = 1$, то*

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

4.2. Алгоритм RSA

Сообщения передаются в виде чисел. Схема RSA относится к схеме шифрования *с открытым ключом*. Каждый получатель обладает своим открытым ключом, с помощью которого он шифрует сообщение, а потом декодирует.

Модулем RSA называется произведение двух различных простых чисел $N = pq$ (каждое из которых выбирается очень большим).

Функция Эйлера для него равна $\varphi(N) = (p-1)(q-1)$. Сложность задачи декодирования как раз и состоит в вычислении функции Эйлера по известному модулю RSA. Для этого надо факторизовать модуль RSA, то есть "всего лишь" разложить N на два множителя. Когда Мартин Гарднер опубликовал алгоритм RSA, он предложил за приз в 100 долларов факторизовать число N с 129 десятичными знаками. Правильный ответ был получен через 17 лет.

Алгоритм RSA.

1. Джон выбирает два простых p, q и вычисляет функцию Эйлера $\varphi(N) = (p-1)(q-1)$ для модуля $N = pq$, которая ему нужна для задания *открытой экспоненты* e взаимно простой с функцией Эйлера от модуля, $(\varphi(N), e) = 1$. Из этих данных открытыми являются только N и e. Пару (N, e) Джон пересылает по открытому каналу связи получателю Алисе. Числа p и q, называемые числами RSA, Джон не доверяет никому.

2. Джон также вычисляет *закрытую экспоненту* d по формуле $d \cdot e \equiv 1 \ (mod\, \varphi(N))$, которая служит обратным элементом к открытой экспоненте e по секретному модулю $\varphi(N)$ (для существования d требовали $(\varphi(N), e) = 1$).

3. Алиса (или любой другой получатель) использует открытый ключ (N, e) для шифрования сообщения $m \in \mathbb{N}$ с помощью функции $M = m^e \ (\, mod\, N)$ и пересылает Джону по открытому каналу.

4. Джон обрабатывает вернувшееся сообщение M с помощью закрытого ключа $M^d \ (mod\, N)$ и тем самым восстанавливает сообщение Алисы

$$(m^e)^d = m^{de} = m^{1+K\varphi(N)} = m \cdot (m^{\varphi(N)})^K \equiv m \ (mod\, N).$$

Пример 4.3 *работы алгоритма RSA*.

1. Возьмем простые $p = 3$, $q = 11$ и вычислим модуль $N = 33$ и $\varphi(33) = \varphi(3) \cdot \varphi(11) = (3-1)(11-1) = 20$ функцию Эйлера для него. Выберем в качестве открытой экспоненты $e = 7$. Отправим открытый ключ $(33, 7)$.

2. Вычислим закрытый ключ $d = 3$, так как $7 \cdot 3 = 21 \equiv 1 \ (mod\, 20)$.

3. Алиса решила отправить нам сообщение $m = 9$. Для кодирования она использует открытый ключ $(33, 7)$ и получает

$$9^7 = 4782969 \equiv 15 \ (mod\, 33).$$

4. Получив сообщение $M = 15$, мы декодируем его с помощью закрытой экспоненты

$$15^3 = 3375 \equiv 9 \ (mod\, 33).$$

В этой схеме требуется исключить возможность подлога, когда полученное Джоном сообщение M якобы от Алисы придет от другого

лица. Диффи и Хеллман заметили, что алгоритм RSA и другие подобные алгоритмы обладают симметрией, что позволяет небольшим усложнением алгоритма добиться подтверждения подлинности отправителя (*электронная подпись*).

Схема симметричного варианта алгоритмов типа RSA.

1. Отправитель Алиса шифрует сообщение с помощью открытого ключа получателя Джона (это шаг, гарантирующий конфиденциальность).

2. Отправитель Алиса снова шифрует сообщение с помощью своего закрытого ключа (для подтверждения подлинности, письмо "подписано").

3. Получатель Джон с помощью открытого ключа Алисы расшифровывает сообщение второго шага (тем самым проверяя его подлинность).

4. Получатель Джон с помощью своего закрытого ключа расшифровывает сообщение первого шага.

Есть другие усовершенствования схемы с симметричными алгоритмами. Одной из таких схем служит схема шифрования с сеансовым ключом. В этом случае описанная выше симметричная схема используется только для передачи открытого сеансового ключа, сгенерированного случайным образом. Один из адресатов шифрует этим сеансовым ключом, симметричным алгоритмом, а другой дешифрует закрытой частью этого сеансового ключа. По окончании сеанса ключ m уничтожается. В этом случае повтор сообщения при новом сеансе будет выглядеть совершенно иначе.

Вопросы для самопроверки

1. Что называется модулем сравнения?

2. Как называется и обозначается для натурального n количество натуральных чисел, которые меньше n и взаимно простые с n?

3. Является ли функция Эйлера монотонно возрастающей?

4. В каком случае функция Эйлера произведения двух чисел равна произведению функций Эйлера этих чисел?

5. Приведите формулировку малой теоремы Ферма.

6. Откуда появилось название алгоритма RSA?

7. Что называется модулем RSA? Почему модуль RSA берется очень большим?

8. Что в алгоритме RSA называется открытой экспонентой, а что закрытой экспонентой? Чем объясняются эти названия?

9. Как часто меняют модули RSA при практическом применении алгоритма?

10. Алгоритм RSA применяется для проверки подлинности отправителя или для электронной подписи?

Упражнения

4.1. Разложите на множители числа 15, 25, 75, 225. Какое из этих чисел может быть модулем RSA?

4.2. Вычислите функцию Эйлера чисел 15, 25, 75, 225.

4.3. Если модуль RSA $N = 15$, то какие из чисел 6, 7, 9, 17 можно брать в качестве открытой экспоненты?

4.4. Вычислите закрытую экспоненту d, если модуль RSA $N = 15$ и открытая экспонента $e = 7$.

4.5. Для открытого ключа $(N = 15, e = 7)$ закодируйте сообщение $m = 10$ числом M. Декодируйте M с помощью закрытого ключа.

4.6. Для другого открытого ключа $(N_1 = 21, e_1 = 5)$ вычислите закрытую экспоненту d_1.

4.7. Закодируйте закрытым ключом $(N_1 = 21, d_1)$ из упражнения 4.6 сообщение $m = 10$, получив подпись P.

4.8. Декодируйте подпись P открытым ключом $(N_1 = 21, e_1 = 5)$, тем самым проверив подлинность сообщения.

4.9. Установите порядок и принадлежность действий упражнений 4.4 – 4.8 при передаче сообщения с электронной подписью двух лиц.

ЗАКЛЮЧЕНИЕ

Предмет “Теория кодирования” включает в себя разные разделы прикладной информатики. В данном курсе представлены отдельные базовые разделы, изложенные в рамках современной компьютерной математики. В пособие не были включены исторически первые принципы кодирования и шифрования в виде замены одних символов другими. Также не затрагивались различные применяемые в компьютерах принципы и системы кодирования.

В главе “Арифметическое кодирование” были рассмотрены три конкретные системы счисления, которые основаны на базовых принципах кодирования, но слабо представлены в учебной литературе.

Главу “Алфавитное кодирование” можно назвать основной в представленном курсе, поскольку она включила в себя традиционные темы теории кодирования. Графическое представление кодов делает весь процесс кодирования более наглядным. В этой главе были рассмотрены основные коды Фано, Шеннона и Хаффмана, представлен принцип оптимальности кодирования.

Важный прикладной аспект теории кодирования изучается в главе “Коды с исправлением ошибок”. Здесь мы были вынуждены отказаться от принципов оптимальности для достижения других прикладных целей. Выделены основные идеи и подходы кодирования с исправлением ошибок.

Прикладному направлению в теории кодирования посвящена отдельная глава по криптографии.

Надеемся, что приведенный материал заинтересует студентов и стимулирует их на дальнейшие исследования в данном направлении.

Библиографический список

1. Биркгоф, Г. Современная прикладная алгебра / Г. Биркгоф, Т. Барти. – М. : Мир, 1976. – 400 с.
2. Витерби, А. Д. Принципы цифровой связи и кодирование / А. Д. Витерби, Дж. К. Омура. – М. : Радио и связь, 1982. – 536 с.
3. Воробьев, Н. Н. Числа Фибоначчи / Н. Н. Воробьев. – М. : Наука, 1992. – 192 с.
4. Грэхем, Р. Конкретная математика. Основание информатики / Р. Грэхем, Д. Кнут, О. Паташник. – М. : Мир, 1998. – 708 с.
5. Марков, А. А. Введение в теорию кодирования / А. А. Марков. – М. : Наука, 1982 – 192 с.
6. Нечаев, В. И. Элементы криптографии. Основы теории защиты информации / В. И. Нечаев. – М. : Высш. шк., 1999. – 109 с.
7. Новиков, Ф. А. Дискретная математика для программистов / Ф. А. Новиков. – СПб. : Питер, 2001. – 304 с. – ISBN 5-272-00183-4.
8. Питерсон, У. Коды, исправляющие ошибки / У. Питерсон, Э. Уэлдон. – М. : Мир, 1976. – 596 с.
9. Романовский, И. В. Дискретный анализ / И. В. Романовский. – СПб. : Невский диалект, 2003. – 320 с. – ISBN 5-7940-0114-3.
10. Соловьева, Ф. И. Введение в теорию кодирования / Ф. И. Соловьева. – Новосибирск : Новосиб. гос. ун-т, 2006. – 127 с.
11. Яблонский, С. В. Введение в дискретную математику / С. В. Яблонский. – М. : Высш. шк., 2001. – 384 с. – ISBN 5-06-003951-X.

Учебное электронное издание

БЕСПАЛОВ Михаил Сергеевич
БУРКОВ Владимир Дмитриевич

ЭЛЕМЕНТЫ ТЕОРИИ КОДИРОВАНИЯ

Учебное пособие

Редактор Е. А. Платонова
Технический редактор Ш. Ш. Амирсейидов, Н. В. Пустовойтова
Компьютерная вёрстка М. С. Беспалова, Л. В. Макаровой
Корректор О. В. Балашова
Выпускающий редактор А. А. Амирсейидова

Системные требования: Intel от 1,3 ГГц; Windows XP/7/8/10; Adobe Reader; дисковод CD-ROM.

Тираж 9 экз.

Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых
Изд-во ВлГУ
rio.vlgu@yandex.ru

Институт информационных технологий и электроники
кафедра функционального анализа и его приложений
bespalov@vlsu.ru