Н.Ю. Прокопенко

ТЕОРИЯ КОДИРОВАНИЯ

Учебное пособие

Нижний Новгород
2023

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Нижегородский государственный архитектурно-строительный университет»

Н.Ю. Прокопенко

ТЕОРИЯ КОДИРОВАНИЯ

Утверждено редакционно-издательским советом университета в качестве учебного пособия

Нижний Новгород
ННГАСУ
2023

УДК 519(075)
ББК 22.17я73
П78

Публикуется в авторской редакции

Рецензенты:

И.Н. Цветкова – канд. физ.-мат. наук, доцент кафедры информатики и информационных технологий Нижегородского института управления Российской академии народного хозяйства и государственной службы при Президенте РФ (НИУ РАНХиГС)

О.А. Парфенова – зам. руководителя дивизиона «Корпоративные проекты 1С» ООО «Группа ЛАД»

Прокопенко Н.Ю. Теория кодирования [Электронный ресурс]: учеб. пособие / Н.Ю. Прокопенко; Нижегор. гос. архитектур. - строит. ун-т – Н. Новгород: ННГАСУ, 2023. – 94 с. 1 электрон. опт. диск (CD-RW)

ISBN 978-5-528-00533-1

В пособие включены классические разделы теории кодирования: алфавитное, оптимальное и помехоустойчивое кодирование. Каждый раздел начинается с изложения необходимого теоретического материала, затем приводятся и разбираются примеры, представлены типовые задачи с решениями.

Пособие предназначено для подготовки студентов бакалавриата по направлениям подготовки 09.03.03 Прикладная информатика и 09.03.04 Программная инженерия.

ББК 22.17я73

ISBN 978-5-528-00533-1

Оглавление

1. Введение в теорию кодирования

Теория кодирования– это раздел теории информации, изучающий способы отождествления сообщений с отражающими их сигналами. Задачей теории кодирования является согласование источника информации с каналом связи.

Кодирование буквально пронизывает информационные технологии и является центральным вопросом при решении самых разных (практически всех) задач программирования:

- Представление данных произвольной природы (например, чисел, текста, графики) в памяти компьютера;
- Защита информации от несанкционированного доступа;
- Обеспечение помехоустойчивости при передаче данных по каналам связи;
- Сжатие информации в базах данных.

Кодированием называют универсальный способ отображения информации при ее хранении, обработке и передаче в виде системы соответствий между сигналами и элементами сообщений, при помощи которых эти элементы можно зафиксировать.

Объектом кодирования служит как дискретная, так и непрерывная информация, которая поступает к потребителю через источник информации. Понятие кодирование означает преобразование информации в форму, удобную для передачи по определенному каналу связи.

Обратная операция – *декодирование* – заключается в восстановлении принятого сообщения из закодированного вида в общепринятый, доступный для потребителя.

В теории кодирования существует ряд направлений:

- статическое или эффективное кодирование;
- помехоустойчивое кодирование;
- корректирующие коды;

– циклические коды;

– арифметические коды.

Кодирование – это способ представления объектов одной природы (точек плоскости, чисел, слов какого-либо языка и т.д.) конечными последовательностями объектов другого множества (обычно конечного). Поскольку элементы этого второго конечного множества можно перенумеровать, то его всегда можно считать набором чисел, однако, это необязательно.

Десятичная позиционная система счисления – это способ кодирования натуральных чисел, двоичная система счисления – это другой способ кодирования этих чисел. Римские цифры – еще один способ кодирования.

Декартовы координаты – способ кодирования геометрических объектов.

Замена имен номерами является кодированием, запись текстового файла на съемный носитель – кодирование, представление фотографии в цифровом фотоаппарате в виде файла – кодирование.

Наборы символов, которыми пользуются при кодировании, называются алфавитами. Последовательности символов алфавита называются словами.

Письменность – это алфавитное кодирование. Например, можем закодировать слово «информатика» последовательностью нулей и единиц. Для этого можем заменить буквы их порядковыми номерами в алфавите, и затем номера записать в двоичном виде.

Примеры систем кодирования:

1. Двоичная, восьмеричная и шестнадцатеричная системы счисления

Двоичная система счисления: *каждое натуральное число однозначно представимо в виде суммы степеней 2, причем каждая степень в этой сумме появляется не больше одного раза, т. е. с коэффициентом 0 или 1. Эти коэффициенты составляют представление числа в двоичной системе счисления.*

Например, 83=64+16+2+1=1010011.

Таким образом, каждый набор из k нулей и единиц соответствует какому-либо числу в диапазоне от 0 до $2k-1$.

При переводе чисел из десятичной системы счисления в систему с основанием $P > 1$ обычно используют следующий алгоритм:

1) если переводится целая часть числа, то она делится на P, после чего запоминается остаток от деления. Полученное частное вновь делится на P, остаток запоминается. Процедура продолжается до тех пор, пока частное не станет равным нулю. Остатки от деления на P выписываются в порядке, обратном их получению;

2) если переводится дробная часть числа, то она умножается на P, после чего целая часть запоминается и отбрасывается. Вновь полученная дробная часть умножается на P и т. д. Процедура продолжается до тех пор, пока дробная часть не станет равной нулю. Целые части выписываются после запятой в порядке их получения. Результатом может быть либо конечная, либо периодическая дробь в системе счисления с основанием P. Поэтому, когда дробь является периодической, приходится обрывать умножение на каком-либо шаге и довольствоваться приближенной записью исходного числа в системе с основанием P.

Пример. Перевести из десятичной в двоичную систему счисления число 42.73.

Отдельно переведем целую и дробную части числа.

а) Для перевода целой части (=42) проводим ряд последовательных делений десятичного числа 42 на 2. При каждом таком делении находим частное (r) и остаток (q). На каждом шаге полученное частное вновь делим на 2.

42:2=21 (0)	r=21, q=0
21:2=10 (1)	r=10, q=1
10:2=5 (0)	r=5, q=0
5:2=2 (1)	r=2, q=1
2:2=1 (0)	r=1, q=0

1:2=0 (1)	r=0, q=1 (Stop)

Полученные остатки q выписываем, начиная с самого последнего и до первого. Это и будет двоичный код числа 42: $42_{10} = 101010_2$.

b) Для перевода дробной части (0.73) заданного десятичного числа в двоичное проводим ряд последовательных умножений на 2. В полученном произведении на каждом шаге отделяем целую часть (r) от дробной (q). Дробную часть вновь умножаем на 2 и т. д. Процесс заканчивается, если на очередном шаге дробная часть будет равна нулю. Здесь возможны случаи, при которых очередная дробная часть уже встречалась на предыдущих шагах процесса, то есть мы получим периодическую двоичную дробь. Если такой момент не наступает, то мы имеем с бесконечной двоичной дробью. Выписываем полученные целые части в порядке их появления от первой до последней – это и будет искомая двоичная дробь.

0.73·2=1. (46)	r=1, q=0.46
0.46·2=0. (92)	r=0, q=0.92
0.92·2=1. (84)	r=1, q=0.84
0.84·2=1. (68)	r=1, q=0.68
0.68·2=1. (36)	r=1, q=0.36
и т.д.	

Мы прервали перевод дроби 0.73 и получили ее приближенное представление в виде бинарной дроби: 0.73≈0.10111…

Окончательно получаем: 42.73≈101010.10111…

При переводе чисел из системы счисления с основанием *P* в десятичную систему счисления необходимо пронумеровать разряды целой части справа налево, начиная с нулевого, и в дробной части, начиная с разряда сразу после запятой слева направо.

Пример.

a) $1000001_{(2)}$.

$1000001_{(2)}=1\cdot2^6+0\cdot2^5+0\cdot2^4+0\cdot2^3+0\cdot2^2+0\cdot2^1+1\cdot2^0 = 64+1=65_{(10)}$.

Замечание. Очевидно, что если в каком-либо разряде стоит нуль, то соответствующее слагаемое можно опускать.

б) $1000011111{,}0101_{(2)}$.

$1000011111{,}0101_{(2)}=1\cdot2^9 + 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 + 1\cdot2^{-2} + 1\cdot2^{-4}= 512 + 16 + 8 + 4 + 2 + 1 + 0{,}25 + 0{,}0625 = 543{,}3125_{(10)}$.

Двоичная система счисления удобна для компьютера, но неудобна для человека – слишком длинные получаются записи чисел:

$37965_{10} = 1001010001001101_2$

Компромиссом между интересами человека и машины являются системы счисления с основаниями, близкими к 10, но являющимися степенью двойки – 8 и 16.

В восьмеричной системе, совершенно естественно, 8 цифр – 0, 1, 2, 3, 4, 5, 6, 7. Каждому разряду восьмеричной системы соответствуют 3 разряда двоичной системы, и переход от одной системы к другой очень прост:

0 — 000	2 — 010	4 — 100	6 — 110
1 — 001	3 — 011	5 — 101	7 — 111

$3796510 = 100101000100110{12} = 1121158$

Восьмеричная запись сейчас используется относительно редко. Шестнадцатеричная система, напротив, очень популярна.

Соответствие между шестнадцатеричными цифрами и десятичными числами

16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Соответствие между шестнадцатеричными цифрами и двоичным разложением:

0	—	0000	4	—	0100	8	—	1000	C	—	1100
1	—	0001	5	—	0101	9	—	1001	D	—	1101
2	—	0010	6	—	0110	A	—	1010	E	—	1110
3	—	0011	7	—	0111	B	—	1011	F	—	1111

$37965_{10} = 1001010001001101_2 = 112115_8 = 944D_{16}$

2. Кодирование текста

Текст – это последовательность символов, входящих в некоторый алфавит. Кодирование текста сводится к двоичному кодированию алфавита, на основе которого он построен. Чаще всего применяется байтовое кодирование алфавита. В этом случае максимальная мощность алфавита составляет 256 символов. Такой алфавит может содержать два набора буквенных символов (например, русский и латинский), цифры, знаки препинания и математические знаки, пробел и небольшое число дополнительных символов. Примером такого алфавита является код ASCII.

ASCII – это аббревиатура для American Standard Code for Information Interchange. Он стал мировым стандартом кодировки для практически всех компьютеров. В этом коде на каждый символ приходится по одному байту, причем стандарт фиксирует верхнюю половину таблицы с кодами от 0 до 127.

Представив байт двумя 16-ми цифрами, в таблице представлены коды от 00 до 7F.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2	sp	!	“	#	$	%	&	^	(	)	*	+	.	,	-	/
3	0	1	2	3	4	5	6	7	8	9						
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z					
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z					

Первые две строчки заполнены служебными символами (типа возврата каретки, перевода строки, табуляции, звонка, конца передачи). Следующая строка начинается с 20 – кода пробела. Дальше идут различные специальные знаки. Дальше идет строка цифр: от 30 для 0 до 39 для 9. Остаток заполнен спецзнаками. Строки 4 и 5 заполнены прописными латинскими буквами: 40 для @, 41 для A и т. д. вплоть до 5A для Z. Строки 6 и 7 аналогично заполнены строчными латинскими буквами. 60 используется для `. Полную таблицу можно посмотреть в справочнике.

Однако, ограниченный набор из 256 кодов символов сегодня уже не удовлетворяет возросшие потребности международного общения. Все большее распространение получает универсальная система 16-разрядного кодирования символов UNICODE.

Мощность алфавита в системе кодирования UNICODE составляет 2^{16}=65 536 разных кодов, из которых 63 484 кода соответствуют символам большинства алфавитов, а оставшиеся 2048 кодов разделены пополам и образуют таблицу размером 1024 столбцов х 1024 строк. В этой таблице более миллиона ячеек, в которых можно разместить еще более миллиона различных символов. Это символы «мертвых» языков, а также символы, не имеющие лексического содержания, указатели, знаки и т. п. Для записи этих дополнительных символов необходима пара 16-разрядных слов (16 разрядов для номера строки и 16 разрядов для номера столбца).

Таким образом, система UNICODE является универсальной системой кодирования всех символов национальных письменных систем и обладает возможностью существенного расширения.

3. Кодирование изображений

Еще одна очень важная трактовка набора из нулей и единиц – это кодирование геометрического изображения.

Рисунки, картинки, фотографии кодируются в растровом формате. В этом виде каждое изображение представляет собой прямоугольную таблицу, состоящую из цветовых точек. Цвет и яркость каждой отдельной точки выражаются в числовой форме, что позволяет использовать двоичный код для представления графических данных.

Черно-белые изображения принято представлять в градациях серого цвета, для этого используется модель GreyScale. Если яркость точки кодируется одним байтом, можно использовать 256 различных серых тонов. Такая точность согласуется с восприимчивостью человеческого глаза и возможностями полиграфической техники.

При кодировании цветных изображений применяют принцип декомпозиции цвета на составляющие, для этого используют модель RGB. Цветное изображение на экране получается путем смешивания трех базовых цветов: красного (Red, R), синего (Blue, B) и зеленого (Green, G).

Двухцветная картинка (черно-белая) – черный рисунок на белом фоне может трактоваться как растр – совокупность отдельных точек, расставленных на прямоугольной решетке.

Сопоставляя черным точкам единицы, а белым – нули, можно закодировать картинку в виде таблицы из нулей и единиц. Затем таблицу можно "вытянуть" в одну линию, соединяя ее строки или столбцы.

Пример. Рассмотрим решетку 16 х 16 и картинку на ней

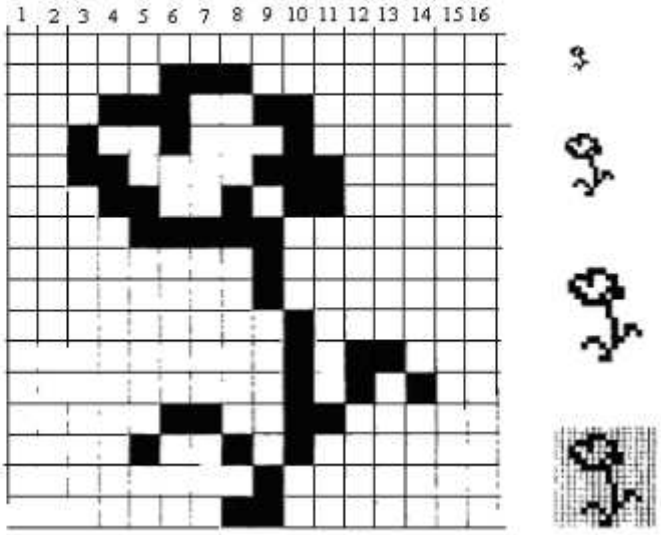


Рис. 1. Пример кодирования картинки

Кодируя столбцы двоичными цифрами, имеем

Столбец	Двоичный код	Шестнадцатеричный код
1	0000 0000 0000 0000	0000
2	0000 0000 0000 0000	0000
3	0001 1000 0000 0000	1800
4	0010 1100 0000 0000	2C00
5	0010 0110 0000 0100	2604

6	0111 0010 0000 1000	7208
7	0100 0010 0000 1000	4208
8	0100 0110 0000 0101	4605
9	0010 1011 1000 0011	2B83
10	0011 1100 0111 1100	3C7C
11	0000 1100 0000 1000	0C08
12	0000 0000 0011 0000	0030
13	0000 0000 0010 0000	0020
14	0000 0000 0001 0000	0010
15	0000 0000 0000 0000	0000
16	0000 0000 0000 0000	0000

4. Кодирование звука и видео

Приемы работы со звуковой информацией пришли в компьютерную технику позже всего. Аналитический метод кодирования, применимый к любым звуковым сигналам основан на аналогово-цифровом преобразовании. Исходный аналоговый сигнал представляют как последовательность цифровых сигналов, записанных в двоичном коде. Разрядность преобразования определяет объем данных, соответствующих отдельному цифровому сигналу. При воспроизведении звука выполняют обратное цифро-аналоговое преобразование.

Этот метод кодирования содержит погрешность, так что воспроизводимый сигнал несколько отличается от оригинала.

Метод кодирования на основе табличного синтеза применим только к музыкальным произведениям. В заранее подготовленных таблицах хранятся образцы (сэмплы) звуков различных музыкальных инструментов. Числовые коды определяют инструмент, ноту и продолжительность звучания.

При кодировании видеосигнала требуется записать последовательность изображений (кадров) и звук (звуковая дорожка). Формат видеозаписи позволяет включить оба потока данных в одну цифровую последовательность.

5. Кодирование вершин куба

Последовательностью из *k* нулей и единиц можно обозначать вершины *k*-мерного куба:

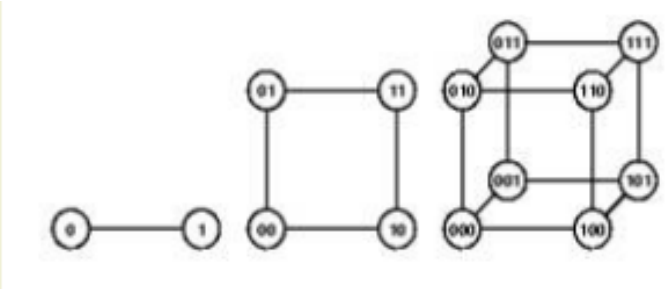


Рис. 2. Представление кубов с пронумерованными вершинами для k=1,2, 3

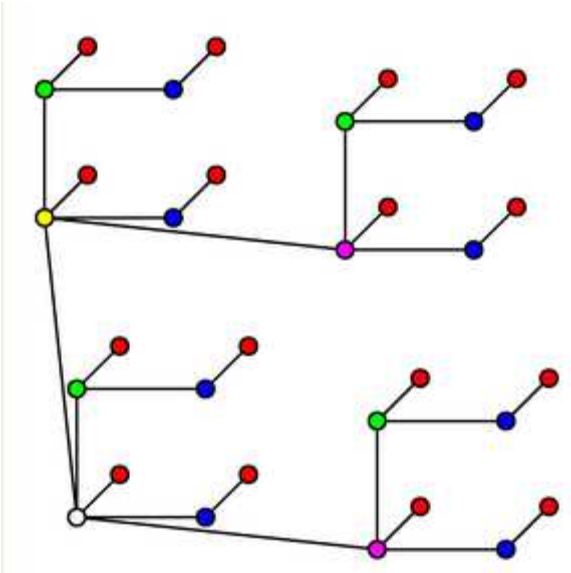


Рис. 3. Изображение 5-мерного куба, в котором проведены только ребра, необходимые для минимальной связи между вершинами

Каждый процессор кодируется набором из пяти 0–1 символов. Процессор, в коде которого последний символ 1, связывается с процессором, у которого код отличается только в последнем символе – там стоит 0. Например, процессор 10111 прицепляется к 10110. Процессор, код которого оканчивается на 10, прицепляется к процессору с кодом, отличающимся во втором с конца символе: так что 10110 связывается с 10100 и т. д. В общем случае, для каждого процессора можно определить всех его соседей – они отличаются от него только одним числом на какой-то позиции

6. Характеристический вектор множества

Пусть $B=\{0,1\}$. Тогда множество B^n состоит из последовательностей нулей и единиц длины *n*. Они называются *строкой бит* или *битовой строкой*. Строка бит применяется для моделирования операций на конечных множествах.

Пусть $S=\{s_1,s_2,\ldots,s_n\}$. Если $A\subseteq S$. Поставим подмножеству A в соответствие *n*-битную строку $(b_1,b_2,\ldots,b_n)$, где $b_i=1$, если $s_i\in A$ и $b_i=0$ в противном случае. Тогда строка бит называется *характеристическим вектором* подмножества *A*. Можно имитировать операции на множествах логическими операциями, применяемыми к соответствующим характеристическим векторам.

Пример. Пусть $S=\{1,2,3,4,5\}$, $A=\{1,3,5\}$ и $B=\{3,4\}$. Тогда характеристическим вектором множества *A* является $a=(1,0,1,0,1)$, а характеристическим вектором множества *B* является $b=(0,0,1,1,0)$.

Операции над множествами легко моделируются логическими операциями над их характеристическими векторами. Например,

a *или* $b=$ $(1,0,1,0,1)$ *или* $(0,0,1,1,0)=(1,0,1,1,1)$, тогда $A\cup B=\{1,3,4,5\}$

a *и* $b=$ $(1,0,1,0,1)$ *и* $(0,0,1,1,0)=(0,0,1,0,0)$, тогда $A\cap B=\{3\}$

$не b=не(0,0,1,1,0)=(1,1,0,0,1)$, тогда $\overline{B}=\{1,2,5\}$.

Задания

1. Переведите в двоичную систему два числа 34567, 16 777 351.
2. Переведите в 8-ричную и 16-ричную систему число 10101011100010100101010110.
3. Выполните все логические операции с наборами 101010010101001010101010 и 000010111101011101011101. (пересечение, объединение, разность, дополнение).

4. Закодируйте в ASCII и переведите в двоичную систему текст Hello, Dolly!
5. Дана кодовая таблица азбуки Морзе

А •–	Л •–••	Ц –•–•
Б –•••	М ––	Ч –––•
В •––	Н –•	Ш ––––
Г ––•	О –––	Щ ––•–
Д –••	П •––•	Ъ •––•–•
Е •	Р •–•	Ы –•––
Ж •••–	С •••	Ь –••–
З ––••	Т –	Э ••–••
И ••	У ••–	Ю ••––
Й •–––	Ф ••–•	Я •–•–
К –•–	Х ••••	

а) Расшифруйте (декодируйте), что здесь написано (буквы отделены друг от друга пробелами):

–·– ·–·· ·– ·–– ·· ·– – ··– ·–· ·–

б) Закодируйте с помощью азбуки Морзе слова: информатика, данные, алгоритм.

6. Нарисуйте черно-белую растровую картинку 16x16 (букву или цифру) и закодируйте ее по строчкам и по столбцам.
7. Узнайте, что изображено на картине, если известно только следующее:

0000, 0018, 0124, 01E2, 03C4, 04B8, 0250, 0370,027C, 5650, 6F4C, 0F92, 00A2, 0154, 01C8, 0000

2. Алфавитное кодирование

Теория кодирования – один из разделов дискретной математики, в котором исследуются отображения конечных или счетных множеств объектов произвольной природы в конечные множества символов (часто в множества последовательностей из чисел 0,1,2,...q-1, где q – некоторое натуральное число (в частности q=2). Такие отображения называются кодированиями.

Кодом называется система условных знаков (символов) для передачи, обработки и хранения (запоминания) различной информации.

Если все кодовые слова имеют одинаковую длину, то код называется *равномерным*, или *блочным*.

n	$e(n)$	$e_4(n)$
0	0	0000
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101

$e_4(n)$ – равномерный код

$e(n)$ не является равномерным

Пусть задано множество $A = \{a_1, \dots, a_n\}$, которое называется алфавитом. Элементы алфавита называют буквами. Конечная последовательность символов из A называется, словом, в алфавите A. A^+ – множество слов в A. $A^* = A^+ \cup \{\lambda\}$, где λ – пустое слово. Длиной слова называется число букв в нем. Соединение слов α и β обозначим через $\alpha\beta$, а соединение n одинаковых слов через α^n ($\alpha^0 = \lambda$).

Множество A называется *алфавитом сообщений*, а множество B – *кодирующим алфавитом*. Множество слов, составленных в алфавите B, обозначим B^*.

Обозначим через F отображение слов алфавита A в алфавит B. Тогда слово $b = F(a)$ называется *кодом* слова a.

Задачу кодирования можно сформулировать следующим образом: пусть заданы алфавиты $A = \{a_1, \dots, a_n\}$, $B = \{b_1, \dots, b_m\}$ и функция $F: S \to B^*$, где S – некоторое множество слов в алфавите A ($S \subset A^*$). Тогда функция F называется

кодированием, слова $\alpha \in S$ из S– сообщениями, а элементы $\beta = F(\alpha)$, $\beta \in B^*$– кодами соответствующих сообщений. Обратная функция F^{-1} (если она существует) называется *декодированием.*

Кодирование F или код $F(S)$ называется взаимно однозначным или *однозначно декодируемым*, если каждый код сообщения является кодом ровно одного сообщения. Если кодирование взаимно однозначно, то код называется *свободным*.

Если $|B| = q$, то F называется q-м кодированием. Наиболее распространенный случай $B = \{0,1\}$ – *двоичное* кодирование.

Примеры кодирования:

а)

n	$e(n)$	$e_4(n)$
0	0	0000
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101

$e(n)$ – не является свободным кодом
$\mathrm{e}_4(\mathrm{n})$ – свободный код.

б) Пусть $A = \{a, b, c, d\}$, двоичное кодирование его букв задается отображением:

$$\begin{cases} a \to 01 \\ b \to 100 \\ c \to 101 \\ d \to 0 \end{cases}$$

Данное кодирование не является взаимно однозначным, и значит код не является свободным, например, слово 0100 можно декодировать как *db* или *add*, слово 0010100 – как *ddcdd*, *daadd* или *dadb.*

в) азбука Морзе (не является свободным), поэтому дополняют пробелом (паузой).

Многие задачи теории кодирования формулируются следующим образом: при заданных алфавитах *A, B* и множестве сообщений *S* найти такое кодирование *F*, которое обладает определенными свойствами и оптимально в

некотором смысле. Критерий оптимальности, как правило, связан с минимизацией длин кодов. Свойства: существование декодирования; помехоустойчивость, заданная сложность (или простота) кодирования и декодирования. Некоторые задачи теории кодирования и подходы к решению этих задач мы будем рассматривать.

Определим алфавитное (побуквенное) кодирование. Пусть задано отображение Σ букв алфавита A в множество B^* вида Σ: $\begin{cases} a_1 \to B_1 \\ a_2 \to B_2 \\ \cdots \\ a_r \to B_r \end{cases}$.

Алфавитным кодированием, задаваемым схемой Σ, называется кодирование F_Σ, удовлетворяющее свойствам:

1) $F_\Sigma\ (a_i)$=B_i $i = 1, \cdots, r$;

2) $F_\Sigma\ (a_{i_1},\ a_{i_2},\ \cdots, a_{i_k}) = B_{i_1}B_{i_2} \ldots B_{i_k}$, где под произведением слов понимается приписывание слова B справа к слову A.

Множество кодовых слов $C(\Sigma) = \{B_{i_1}B_{i_2} \ldots B_{i_k}\}$ называется *кодом* алфавита в схеме Σ; $B_{i_1}B_{i_2} \ldots B_{i_k}$ – элементарные коды. Если все буквы a_i кодируются словами одинаковой длины, то код $C(\Sigma)$ называется *равномерным*. Можно считать, что в коде (множестве кодовых слов) содержатся только различные слова.

Длина кода – число знаков в коде без учета пробелов. *Спектр кода* – это упорядоченная по не возрастанию последовательность длин кодовых слов заданной схемы кодирования.

Код называется *разделимым*, если любое слово, составленное из элементарных кодов, единственным образом разлагается на элементарные коды, т.е. из каждого равенства в алфавите B вида $B_{i_1}B_{i_2} \ldots B_{i_k}$=$B_{j_1}B_{j_2} \ldots B_{j_l}$ следует, что $k = l$ и $i_t = j_t$, $t = 1,2, \ldots, k$.

Алфавитное кодирование является взаимно однозначным тогда и только тогда, когда оно задается с помощью разделимого кода.

Алфавитное кодирование с разделимой схемой допускает декодирование. Из определения следует также, что все слова разделимого кода различны и не пусты.

Одним из основных вопросов в кодировании является *проблема взаимной однозначности*, т. е. возможность по коду β сообщения α однозначно восстановить α.

Возникает вопрос: возможно ли по схеме алфавитного кодирования узнать, обладает ли алфавитное кодирование свойством однозначности или нет? Если решать эту задачу исходя из определения, то нужно перебрать бесконечное число слов, так как для каждого кода нужно установить, допускает или не допускает этот код однозначное кодирование.

Пример 1.

Задано алфавитное кодирование, для которого $A = \{a_1,\ a_2\}$, $B = \{b_1,\ b_2\}$, и схема Σ задается таблицей:

a_1	a_2
b_1	$b_1 b_2$

L=(1, 2) – спектр кода.

Проверим обладает ли эта схема кодирования свойством однозначности.

Решение.

Процесс декодирования осуществляется следующим образом: код слова α разбивается на элементарные коды. Для этого в слове β последовательно находят все буквы b_2 затем выделяют пары $b_1 b_2$, каждая такая пара соответствует букве a_2. Оставшиеся буквы b_1соответствуют букве a_1.

Так код β=$b_1 b_1 b_1 b_2 b_1 b_2 b_1 b_2 b_1 b_1 b_2$ разбивается однозначно на элементарные коды β=$(b_1)(b_1)(b_1 b_2)(b_1 b_2)(b_1 b_2)(b_1)(b_1 b_2)$. Следовательно, исходное сообщение есть слово α=$a_1 a_1 a_2 a_2 a_2 a_1 a_2$

Пример 2.

Задано алфавитное кодирование, для которого $A = \{a_1,\ a_2\}$, $B = \{b_1,\ b_2\}$, и схема Σ задается таблицей:

a_1	a_2	a_3
b_1b_1	$b_2b_1b_1$	$b_1b_1b_2$

L=(2, 3, 3) – спектр кода.

Покажем, что эта схема не обладает свойством однозначности.

Решение. Рассмотрим слово β=$b_1b_1b_2b_1b_1b_2b_1b_1$

Это слово допускает два декодирования: $\beta_1=(b_1b_1)(b_2b_1b_1)(b_2b_1b_1)$ и $\beta_2=(b_1b_1b_2)(b_1b_1b_2)(b_1b_1)$, тогда $\alpha_1=a_1a_2a_2$ и $\alpha_2=a_3a_3a_1$.

Следовательно, схема Σ не обладает свойством однозначности.

Утверждение (*неравенство Макмиллана*). Для того чтобы схема алфавитного кодирования была разделимой, т.е. для однозначной декодируемости кода в q-буквенном кодирующем алфавите *необходимо*, чтобы длины элементарных кодов удовлетворяли соотношению, известному как неравенство Макмиллана: $\frac{1}{q^{|B_1|}}+\frac{1}{q^{|B_2|}}+\cdots+\frac{1}{q^{|B_r|}}\leq 1.$

Здесь $|B_i|$ – длина кодового слова (*длиной слова* называется число букв в нем).

В частности, если задана схема алфавитного кодирования для кодирующего алфавита $B=\{0,1\}$:

$$\Sigma:\ \begin{cases} a_1 \to B_1 \\ a_2 \to B_2 \\ \quad \cdots \\ a_r \to B_r \end{cases}$$

тогда если эта схема будет разделимой, то справедливо неравенство:

$$\frac{1}{2^{|B_1|}}+\frac{1}{2^{|B_2|}}+\cdots+\frac{1}{2^{|B_r|}}\leq 1.$$

Пример 3.

Схема алфавитного кодирования для алфавитов$A=\{a,б\}$ и $B=\{0,1\}$

$\begin{cases} a \to 0 \\ б \to 01 \end{cases}$ является разделимой.

$|b_1|=1, |b_2|=2,$

и, значит, выполняется неравенство Макмиллана:

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

Данная схема префиксной не является, так как элементарный код буквы *a* является префиксом элементарного кода буквы *b***.**

Пример 4.

Азбука Морзе – это схема алфавитного кодирования:

$$\langle\ A\to 01, B\to 1000, C\to 1010, D\to 100, E\to 0, F\to 0010, G\to 110, H\to 0000,$$
$$I\to 00, J\to 0111, K\to 101, L\to 0100, M11, N\to 10, O\to 111, P\to 0110, Q\to 1101,$$
$$R\to 010, S\to 000, T\to 1, U\to 001, V\to 0001, X\to 1001, Y\to 1011, Z\to 1100\ \rangle$$

где по историческим и техническим причинам 0 называется точкой и обозначается знаком «•», а 1 называется тире и обозначается знаком «-». Имеем:

$$^1/_4+{}^1/_{16}+{}^1/_{16}+{}^1/_8+{}^1/_2+{}^1/_{16}+{}^1/_8+{}^1/_{16}+{}^1/_4+{}^1/_{16}+{}^1/_8+{}^1/_{16}+{}^1/_4+{}^1/_4$$
$$++{}^1/_8+{}^1/_{16}+{}^1/_{16}+{}^1/_8+{}^1/_8+{}^1/_2+{}^1/_8+{}^1/_{16}+{}^1/_8+{}^1/_{16}+{}^1/_{16}$$
$$+{}^1/_{16}+{}^1/_{16}==\ {}^2/_2+{}^4/_4+{}^7/_8+{}^{12}/_{16}=\ {}^{3+5}/_8>1.$$

Таким образом, неравенство Макмиллана для азбуки Морзе не выполнено, и эта схема не является разделимой. На самом деле в азбуке Морзе имеются дополнительные элементы – паузы между буквами (и словами), которые позволяют декодировать сообщения. Эти дополнительные элементы определены неформально, поэтому прием и передача сообщений с помощью азбуки Морзе, особенно с высокой скоростью, является некоторым искусством, а не простой технической процедурой.

3. Алгоритм Маркова распознавания взаимной однозначности алфавитного кодирования

Для определения однозначности или неоднозначности схемы алфавитного кодирования существует эффективный алгоритм, использующий понятие нетривиального разложения элементарных кодов.

Определение. Представление элементарного кода B_i схемы алфавитного кодирования Σ в виде $B_i=\beta'\omega\beta''$, где слово β' не может оканчиваться на элементарный код, а слово β'' начинаться с элементарного кода, будем называть *нетривиальным разложением* элементарного кода, при этом одно из слов β', ω или β'' может быть пустым λ.

Для наглядности будем слова в разложении отделять точками $B_i=\beta'.\omega.\beta''$.

Теорема (Марков А.А.) Алфавитное кодирование со схемой Σ не обладает свойством взаимной однозначности тогда и только тогда, когда граф G_Σ содержит ориентированный цикл (контур), проходящий через вершину λ. Т.е. алфавитное кодирование является взаимно однозначным (код $C(\Sigma)$ свободен) тогда и только тогда, если граф G_Σ не содержит ориентированного цикла, проходящего через вершину λ (в графе отсутствуют контуры и петли проходящие через вершину λ).

Замечание: напомним, что если схема алфавитного кодирования $\sum$ не обладает свойством взаимной однозначности, то это означает, что существуют слова из B^*, допускающие два декодирования. Одно из таких слов β легко находится по графу G_Σ. Для записи слова β нужно посмотреть ориентированный цикл, проходящий через вершину λ, начиная с λ, и выписать последовательно все слова, приписанные ребрам и вершинам, входящим в этот цикл.

Описание алгоритма Маркова распознавания взаимной однозначности алфавитного кодирования:

Пусть $C(\Sigma)$ – алфавитный код; P_Σ – множество слов β', обладающих следующим свойством: слово β' является собственным префиксом некоторого

кодового слова B; S_Σ – множество слов β'', обладающих следующим свойством: слово β'' является собственным суффиксом некоторого кодового слова B.

Пусть $V = \mathrm{P}_\Sigma \cap S_\Sigma \cup \{\lambda\}$ – множество вершин графа.

Вершины β' и β'' соединяем ориентированным ребром, если для некоторого элементарного кода B_i существует нетривиальное разложение $B_i=\beta'.\omega.\beta''$, где $\beta' \in \mathrm{P}_\Sigma$, $\beta'' \in S_\Sigma$, $\omega \in \mathrm{C}^*(\Sigma)$.

Ребру $(\alpha\beta)$ припишем значение ω. Множество всех ребер обозначим R, множество значений ребер $\omega(R)$. Полученный взвешенный граф для кода $\mathrm{C}(\Sigma)$ обозначим через G_Σ.

Алгоритм построения графа:

1. Для каждого элементарного кода выписываем все нетривиальные разложения $B_i=\beta'.\omega.\beta''$.

2. Выписываем множество P_Σ, состоящее из слов β', которые входят в качестве, начал в нетривиальные разложения элементарных кодов.

3. Выписываем множество S_Σ, состоящее из всех слов β'', которые являются окончанием нетривиальных разложений элементарных кодов.

4. Составляем множество встречающихся как в качестве начала, так и в качестве окончания в нетривиальных разложениях элементарных кодов. $V = \mathrm{P}_\Sigma \cap S_\Sigma \cup \{\lambda\}$.

5. Выписываем все разложения элементарных кодов, связанных с множеством V, т. е. все разложения элементарных кодов вида:
$\beta_i = \beta'\beta_{i1}\ldots\ \beta_{ik}\ \beta''$, где $\beta', \beta'' \in V$, а k может быть равно 0, т. е. $\beta_{i1}\ldots\ \beta_{ik}$ либо λ, либо кодовые слова.

6. По разложениям, полученным в пункте 5, строится ориентированный граф G_Σ следующим образом: вершины графа отождествляют с элементами множества V, пара вершин β'и β''соединяются ориентированными ребрами в том и только в том случае, если существует разложение $\beta'\beta_{i1}\ldots\beta_{ik}\beta''$ При этом ребру ($\beta'\ \beta''$) приписывается слово $\beta_{i1}\ldots\beta_{ik}$, соответствующее этому разложению.

7. По полученному графу $G_{\sum}$ легко проверить, обладает или нет исходная схема кодирования свойством взаимной однозначности (см. теорему Маркова).

Пример 5.

Пусть $A = \{a_1, a_2, a_3\}$ и $B = \{0, 1\} C(\Sigma) = \{1, 010, 101\} \sum \begin{cases} a_1 \to 1 \\ a_2 \to 010 \\ a_3 \to 101 \end{cases}$

1) Выпишем все нетривиальные разложения элементарных кодов:

Для B_1 нет нетривиальных разложений,

$B_2 = 010 = 0.B_1.0 = 01.\lambda.0 = 0.\lambda.10$

$B_3 = \lambda.B_1.01 = 10.B_1.\lambda = 1.\lambda.01 = 10.\lambda.1$

1) $P_\Sigma = \{0, 01, \lambda, 10, 1\}$
2) $S_\Sigma = \{0, 01, \lambda, 10, 1\}$
3) Составляем множество $V = P_\Sigma \cap S_\Sigma \cup \{\lambda\} = \{0, 1, 01, \lambda, 10\}$
4) В данном примере все разложения элементарных кодов, связанных с множеством V, уже получены в п. 1.
5) Строим ориентированный граф $G_{\sum}$ следующим образом: вершины графа – это элементы множества V, взвешенные ребра соответствуют нетривиальным разложениям, полученным в п.1.

Замечание: для построения ребер нужны только разложения $B_i = \beta'.\omega.\beta''$, где $\beta'\ \beta'' -$ из множества вершин, а $\omega \in C^*(\Sigma)$, т.е. является или λ, или каким-либо кодовым словом , или несколькими кодовыми словами.

Соответствующий коду граф изображен на рис. 4.

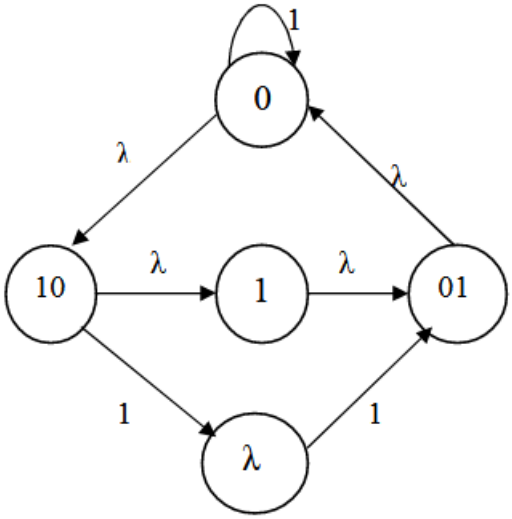


Рис. 4. Граф для примера 5

Граф содержит ориентированный цикл, проходящий через вершину λ.

По графу нетрудно построить двоичную последовательность, допускающую неоднозначное декодирование.

Выписывая слова, приписанные вершинам и ребрам этого цикла, получаем слово $\beta = 1010101$, соответствующее циклу, которое допускает две расшифровки: $b_1\ b_2\ b_3$ и $b_3\ b_2\ b_1$.

Пример 6.

Пусть $A = \{\mathrm{a}_1, \mathrm{a}_2, \mathrm{a}_3, \mathrm{a}_4, \mathrm{a}_5\}$ и $B = \{0, 1\} C(\Sigma) = \{1, 01, 100, 0100, 0000\} \sum \begin{cases} \mathrm{a}_1 \to 1 \\ \mathrm{a}_2 \to 01 \\ \mathrm{a}_3 \to 100 \\ \mathrm{a}_4 \to 0100 \\ \mathrm{a}_5 \to 0000 \end{cases}$

2) Выпишем все нетривиальные разложения элементарных кодов:

Для B_1 нет нетривиальных разложений,

$B_2 = 01 = 0.B_1.\lambda = 0.\lambda.1$

$B_3 = 100 = \lambda.B_1.00 = 1.\lambda.00 = 10.\lambda.0$

$B_4 = 0100 = 0.B_1.00 = \lambda.B_2.00 = 0.B_3.\lambda = 010.\lambda.0$

$B_5 = 0000 = 0.\lambda.000 = 00.\lambda.00 = 000.\lambda.0$

3) $\mathrm{P}_\Sigma = \{0, 1, 01, \lambda, 10, 100, 00, 000\}$

4) $S_\Sigma = \{0, 1, 00, 000, \lambda\}$

5) Составляем множество $V = \mathrm{P}_\Sigma \cap S_\Sigma \cup \{\lambda\} = \{0, 1, \lambda, 00, 000\}$

6) Соответствующий коду граф изображен на рис. 5.

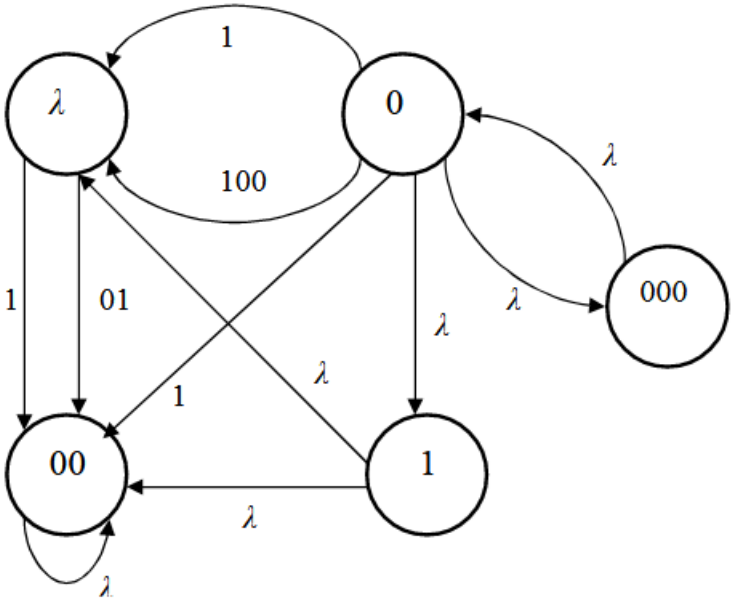


Рис. 5. Граф для примера 6

Граф не содержит ориентированный цикл, проходящий через вершину λ, следовательно код $C(\Sigma)$ является взаимно-однозначным.

Пример 7.

Пусть $A = \{a_1, a_2, a_3, a_4, a_5\}$ и $B=\{b_1, b_2, b_3\}\sum$ $\begin{cases} a_1 \to b_1 b_2 \\ a_2 \to b_1 b_3 b_2 \\ a_3 \to b_2 b_3 \\ a_4 \to b_1 b_2 b_1 b_3 \\ a_5 \to b_2 b_1 b_2 b_2 b_3 \end{cases}$

Смотрим код, сравниваем префиксы и суффиксы и составляем множество $P_\Sigma \cap S_\Sigma \cup \{\lambda\} = \{b_1 b_3, b_2, \lambda\}$ – это и будет множеством V вершин графа $G_\sum$.

Чтобы найти дуги графа, выписываем все разложения элементарных кодов, связанных с множеством V, т. е. все разложения элементарных кодов вида:

$\beta_i = \beta' \beta_{i1} \dots \beta_{ik} \beta''$, где $\beta', \beta'' \in V$, а k может быть равно 0.

B_1 и B_3 не имеют нетривиальных разложений.

$B_2 = b_1 b_3 b_2 = b_1 b_3 . \lambda . b_2$

$B_4 = b_1 b_2 b_1 b_3 = \lambda . B_1 . b_1 b_3$

$B_5 = b_2 b_1 b_2 b_2 b_3 = b_2 . B_1 B_3 . \lambda$

Соответствующий коду граф изображен на рис. 6.

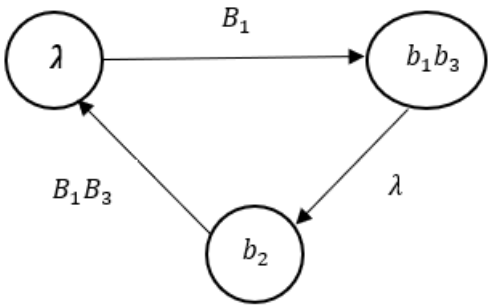


Рис. 6. Граф для примера 7

Граф содержит ориентированный цикл, проходящий через вершину λ: $\lambda, B_1B_3, B_2, \lambda$.

Выписывая слова, приписанные вершинам и ребрам этого цикла, получаем слово $\lambda B_1 b_1 b_3 \lambda b_2 B_1 B_3 \lambda$, которое имеет два разложения по элементарным кодам: $B_1B_2B_1B_3 = B_4B_5$.

Слова $A_1 = \mathrm{a}_4\mathrm{a}_5$ и $A_2 = \mathrm{a}_1\mathrm{a}_2\mathrm{a}_1\mathrm{a}_3$ кодируются одним словом $b_1b_2b_1b_3b_2b_1b_2b_2b_3$. Код $C(\Sigma)$ не является свободным.

Задания

1) Постройте кодовое дерево для кода
   $\Sigma=\{000, 001, 010, 0110, 0111, 1000, 1001\}$
2) Выяснить обладает ли код свойством префикса:
   a) $\mathrm{C} = \{\mathrm{a}, \mathrm{ва}, \mathrm{вв}, \mathrm{ввва}\}$
   b) $\mathrm{C} = \{\mathrm{ав}, \mathrm{ва}, \mathrm{вв}, \mathrm{аав}\}$
   c) $\mathrm{C} = \{\mathrm{ас}, \mathrm{с}, \mathrm{вв}, \mathrm{авс}, \mathrm{вас}, \mathrm{авв}, \mathrm{авсв}\}$
   d) $\mathrm{C} = \{\mathrm{a}, \mathrm{ва}, \mathrm{ввa}, \dots, (\mathrm{в})^n\mathrm{a}, \dots\}$
   e) $\mathrm{C} = \{\mathrm{a}, \mathrm{ва}, \mathrm{ввa}, \dots, \mathrm{в}(\mathrm{a})^n, \dots\}$
3) Существует ли префиксный код со спектром длин $L(V) = \langle 1,2,2,4,4,5 \rangle$ в алфавите B. Если да, то нужно построить префиксный код и кодовое дерево.
   a) в алфавите $B = \{0,1\}$;
   b) в алфавите $B = \{0,1,2\}$;
   c) в алфавите $B = \{0,1,2,3\}$.

4) Нужно, используя алгоритм 2, построить префиксный код и кодовое дерево, если задан спектр длин $L(V) = \langle 2,3,3,3,4,4,4 \rangle$ и кодирующий алфавит $B = \{0,1\}$.

5) Даны алфавиты $A = \{0,1,2,3,4,5,6,7,8,9\}$, $B = \{0,1\}$ и схема Σ, заданная таблицей:

0	1	2	3	4	5	6	7	8	8
0	1	10	11	100	101	110	111	1000	1001

Покажите, что эта схема кодирования не обладает свойством однозначности.

6) Для кода Σ найти слово минимальной длины, декодируемое неоднозначно:
   a) Σ={10, 01, 12, 012, 2100, 12011, 12010}
   b) Σ={0, 101010, 01010101}

7) Выясните, используя алгоритм Маркова, является ли код $C(\Sigma)$ с кодирующим алфавитом B = {0, 1, 2} однозначно декодируемым. Если код не является свободным, то найдите слово, которое декодируется двумя способами.
   a. $C(\Sigma) = \{01, 201, 112, 122, 0112\}$
   b. $C(\Sigma) = \{001, 021, 102, 201, 00112, 122, 0101210\}$

8) Выясните, используя алгоритм Маркова, является ли код $C(\Sigma)$ с кодирующим алфавитом B = {a, b, c} однозначно декодируемым. Если код не является свободным, то найдите слово, которое декодируется двумя способами.
   a. $C(\Sigma) = \{cc, cca, bcac, aa, ab\}$
   b. $C(\Sigma) = \{a, ab, acbb, bb, bbac\}$

9) Выясните является ли слово P в алфавите B = {0, 1, 2} кодом сообщения в кодировании, задаваемом схемой

$$\Sigma \begin{cases} 1 \to 10 \\ 2 \to 12 \\ 3 \to 012 \\ 4 \to 101 \\ 5 \to 2100 \end{cases}$$

Если да, то выяснить, является ли *P* кодом ровно одного сообщения:

a. *P*=10120121012100

b. *P*=1010122100

c. *P*=0121001210201

4. Построение префиксного кода

Пусть задано конечное множество $A=\{a_1,...,a_n\}$, которое называется алфавитом. Последовательность букв называется словом (в данном алфавите). Если $\alpha=\alpha_1\alpha_2$, то α_1 называется началом, или *префиксом, слова* α, а α_2 – окончанием, или *постфиксом*, слова α. Если при этом $\alpha_1 \neq \lambda$ (λ – пустое слово) (соответственно, $\alpha_2 \neq \lambda$), то α_1 (соответственно, α_2) называется собственным началом (соответственно, собственным окончанием) слова α.

Префикс (суффикс) слова называется *собственным,* если он отличен от пустого слова и от самого слова *B*.

Обозначим через P_α – множество всех собственных префиксов и S_α – множество всех собственных суффиксов слова α. Например, для $\alpha = 110$

$P_\alpha = \{1,\ 11\}$ и $S_\alpha = \{0,\ 10\}$.

Определение. Схема Σ (код $C(\Sigma)$, кодирование F_Σ) обладает *свойством префикса,* если для любых слов B_i и B_j ($i \neq j$) из $C(\Sigma)$ слово B_i не является префиксом слова B_j. Код $C(\Sigma)$, обладающий *свойством префикса*, называется еще *префиксным кодом.*

Определение. Схема алфавитного кодирования называется *префиксной***,** если элементарный код одной буквы не является префиксом элементарного кода другой буквы. $(\forall i \neq j \beta_i, \beta_j \in V) \Rightarrow (\forall \beta \in B * \beta_i \neq \beta_j \beta)$.

Определение. Код $C(\Sigma)$ называется *префиксным кодом*, если никакое кодовое слово не является началом никакого другого кодового слова.

Простейшим примером префиксного кода является равномерный код $e_4(n)$**,** он не требует разделителей, поскольку код каждой передаваемой буквы заканчивается, а код каждой следующей буквы начинается через определенное число кодирующих символов.

Теорема. Префиксная схема является разделимой.

Доказательство: от противного. Пусть кодирование со схемой Σ не является разделимым. Тогда существует такое слово $\beta \in F_\Sigma(A^*)$,что

$$\beta = \beta_{i1}\dots\beta_{ik} = \beta_{j1}\dots\beta_{jl} \& \left(\exists t \forall s\left(s < t \Rightarrow \beta_{is} = \beta_{js} \& \beta_{it} \neq \beta_{jt}\right)\right).$$

Поскольку $\beta_{it}\dots\beta_{ik} = \beta_{jt}\dots\beta_{jl}$, значит, $\exists\beta'\left(\beta_{it} = \beta_{jt}\beta' \vee \beta_{jt} = \beta_{it}\beta'\right)$ но это противоречит тому, что схема префиксная.

Замечание: Свойство быть префиксной является достаточным, но не является необходимым для разделимости схемы.

Например, код $\begin{cases} a \to 0 \\ б \to 01 \end{cases}$ двухбуквенного алфавита $A = \{a, б\}$ непрефексный, так как код буквы *а* $F(a)$=0 является префиксом кода $F(б)$=01, но этот является разделимым. В самом деле, если в некотором сообщении после символа 0 следует 0, то первый символ кодирует букву *а*, так как 00 не является началом никакого кодового слова; если же после 0 следует 1, то пара символов 01 – код буквы *б*, так как 0 не может кодировать *а* из-за того, что последующая 1 не является началом никакого кодового слова.

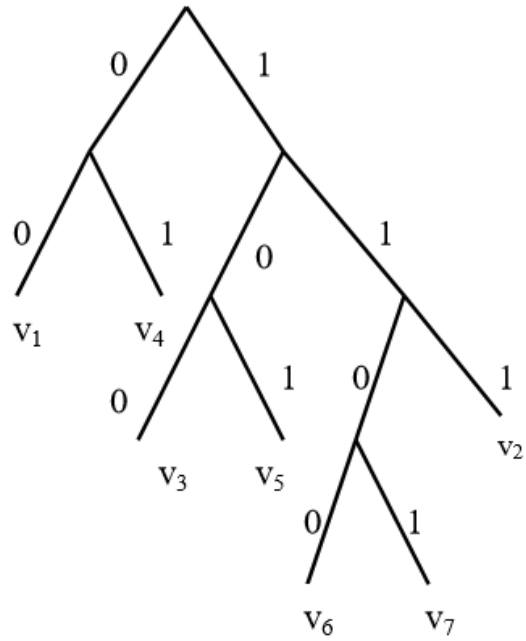
Префиксности кода можно добиться включением в качестве окончания любого кодового слова специального символа – разделителя. Например, в коде Морзе используют (паузу или пробел), а ввод в компьютер имени файла, имени данного, и т.д. должен завершаться разделителем, в частности нажатием клавиши <ENTER>.

4.1. Представление префиксного кода с помощью дерева

Пусть задан префиксный код $V = \{v_1, v_2 \dots v_n\}$ в двоичном алфавите B={0,1}.

Элементарные коды определяют бинарное кодовое дерево. Из каждой вершины дерева выходит не более двух ребер в вершины следующего яруса, левое из которых помечается символом 0, а правое – символом 1. Элементарным кодам соответствуют вершины дерева, определяемые путем, идущим от корня. Если код префиксный, элементарные коды расположены в листьях дерева. Дерево называется насыщенным, если из каждой вершины, не являющейся листом, в следующий ярус выходит ровно два ребра.

Пример 1.

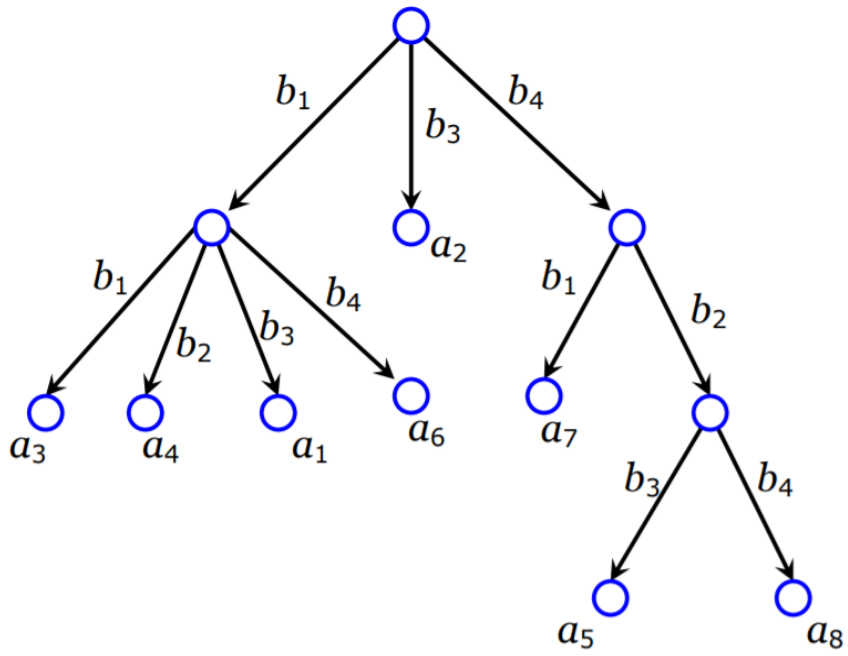
Код:	Кодовое дерево
$\begin{cases} b_1 \to 00 \\ b_2 \to 111 \\ b_3 \to 100 \\ b_4 \to 01 \\ b_5 \to 101 \\ b_6 \to 1100 \\ b_7 \to 1101. \end{cases}$	

Пусть задан префиксный код $V = \{v_1, v_2 \dots v_n\}$ в q-ом алфавите B={0,1,2,...q-1}. Тогда можно построить корневое q-ное дерево, у которого все q ребер, выходящих из каждой вершины занумерованы символами 0,1,2,...q-1 слева направо.

Для каждой вершины существует единственный путь, соединяющий корень дерева с этой вершиной. Путь порождает кодовое слово, которое

получается при выписывании всех значений ребер этого пути, начиная с корневой вершины.

Пример 2.

Дан код:	Кодовое дерево для q=4
$a_1 - b_1b_3$ $a_2 - b_3$ $a_3 - b_1b_1$ $a_4 - b_1b_2$ $a_5 - b_4b_2b_3$ $a_6 - b_1b_4$ $a_7 - b_4b_1$ $a_8 - b_4b_2b_4$	

Для префиксных кодов и только для них множество кодовых слов совпадает с множеством концевых вершин кодового дерева.

4.2. Алгоритмы построения префиксного кода по набору длин элементарных кодов

Пусть задан префиксный код $V = \{v_1, v_2 \dots v_n\}$ в алфавите $B = \{b_1, \dots, b_m\}$. Пусть $|v_i| = l_i$ – длины кодовых слов.

Определение. Упорядоченное множество $L(V) = \langle l_1, \dots, l_n \rangle$ называется *спектром длин кода* V. Коды v_1 и v_2 называются спектрально эквивалентными, если спектры длин у них совпадают, т.е. $L(V_1) = L(V_2)$.

Теорема. Если для чисел $l_1, \dots, l_n$ выполнено неравенство Макмиллана $\frac{1}{q^{l_1}} + \frac{1}{q^{l_2}} + \dots + \frac{1}{q^{l_n}} \leq 1$, то существует префиксный код $V = \{v_1, v_2 \dots v_n\}$ в алфавите $B = \{b_1, \dots, b_m\}$ со спектром длин элементарных

кодов $L(V) = \langle l_1, \dots, l_n \rangle$.

Доказательство. Будем полагать, что $l_1 \le l_2 \le \cdots \le l_n$. Например, $L(V) = \langle 1,3,3,4,4,4,4 \rangle$. Множество чисел разобьем на k непересекающихся классов, в каждом классе будут одинаковые числа. Например, $L_1 = \{1\}$, $L_2 = \{3,3\}$, $L_3 = \{4,4,4,4\}$. Обозначим через s_i – число, определяющее i-й класс, $|L_i|$ – мощность i-го класса (в примере $s_1 = 1$, $s_2 = 3$, $s_3 = 4$, $|L_1| = 1$, $|L_2| = 2$, $|L_3| = 4$).

Неравенство Макмиллана можно переписать в следующем виде:

$$\frac{|L_1|}{q^{s_1}} + \frac{|L_2|}{q^{s_2}} + \cdots + \frac{|L_k|}{q^{s_k}} \le 1 \qquad (1)$$

Это неравенство равносильно следующему:

$$|L_i| \le q^{s_i} - |L_1| \cdot q^{s_i - s_1} - |L_2| \cdot q^{s_i - s_2} - \cdots - |L_k| \cdot q^{s_i - s_k} \qquad (2)$$

Построение префиксного кода (алгоритм 1):

Шаг 1. Из множества B^{s_1} выберем произвольных $|L_1|$ слов. Обозначим их через $V_1 \subseteq B^{s_1}$. Возможность выбора следует из неравенства (2) при $i = 1$.

(Через B^{s_1} обозначено прямое произведение множеств s_1 раз).

Шаг 2. Из множества B^{s_2} выберем произвольных $|L_2|$ слов так, чтобы ни одно из слов V_2 не являлось бы префиксом выбранных. Обозначим их через V_2.

Выбор таких слов возможен в силу выполнимости неравенства (2), при $i = 2$, $|L_2| \le q^{s_2} - |L_1| \cdot q^{s_2 - s_1}$ или $V_2 \subseteq B^{s_2} / (V_1 \times B^{s_2 - s_1})$.

Пусть построены множества: $V_1, V_2, \dots, V_{i-1}$.

i-й шаг. Выберем произвольных $|L_i|$ слов из множества $B^{s_i} / ((V_1 \times B^{s_i - s_1}) \cup (V_2 \times B^{s_i - s_2}) \cup \dots (V_{i-1} \times B^{s_i - s_{i-1}}))$. Это множество обозначим через V_i.

Через k шагов алгоритм заканчивается. Код $V = \bigcup_{i=1}^{k} V_i$– префиксный по построению. ч.т.д.

Пример 3. Пусть $L(V) = \langle 1,3,3,4,4,4,4 \rangle$. Построим префиксный код в алфавите $B = \{0,1\}$ со спектром длин $L(V) = \langle 1,3,3,4,4,4,4 \rangle$.

Проверим, выполняется ли для $L(V) = \langle 1,3,3,4,4,4,4 \rangle$ неравенство

Макмиллана:

$$\frac{1}{2}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^4}+\frac{1}{2^4}+\frac{1}{2^4}+\frac{1}{2^4}\leq 1$$

$$\frac{1}{2}+\frac{2}{2^3}+\frac{4}{2^4}\leq 1$$

$$1\leq 1 \text{ – верно.}$$

Разобьем $L(V)=\langle 1,3,3,4,4,4,4\rangle$ на 3 непересекающихся класса, в каждом классе будут одинаковые числа. Например, $L_1=\{1\}$, $L_2=\{3,3\}$, $L_3=\{4,4,4,4\}$. Каждый класс определяется своим числом и мощностью: $s_1=1, s_2=3, s_3=4, |L_1|=1, |L_2|=2, |L_3|=4$.

Шаг 1. Строим множество $V_1\subseteq\{0,1\}$, $|V_1|=1$ (так как одна единица в спектре $L(V)=\langle 1,3,3,4,4,4,4\rangle$).

Пусть $V_1=\{1\}$.

Шаг 2. Строим множество $V_2\subseteq\{0,1\}^3/(V_1\times\{0,1\}^{3-1})$, $|V_2|=2$ (так как две тройки в спектре $L(V)=\langle 1,3,3,4,4,4,4\rangle$).

Пояснение: Формула $V_2\subseteq\{0,1\}^3/(V_1\times\{0,1\}^{3-1})$ означает, что из всевозможных последовательностей $(b_1\ b_2\ b_3)$ исключаются те, где на первом месте стоит 1.

Пусть $V_2=\{000,001\}$.

Шаг 3. Строим множество $V_3\subseteq\{0,1\}^4/((V_1\times\{0,1\}^{4-1})\cup(V_2\times\{0,1\}^{4-2}))$, $|V_3|=4$ (так как четыре 4-ки в спектре $L(V)=\langle 1,3,3,4,4,4,4\rangle$).

Пояснение: Формула $V_3\subseteq\{0,1\}^4/((V_1\times\{0,1\}^{4-1})\cup(V_2\times\{0,1\}^{4-2}))$ означает, что из всевозможных последовательностей $(b_1\ b_2\ b_3 b_4)$ исключаются те, которые начинаются 1, или на 000, 001.

Пусть $V_3=\{0100,0101,0110,0111\}$.

Искомый код $V=V_1\cup V_2\cup V_3=\{1,000,001,0100,0101,0110,0111\}$ префиксный по построению.

Замечание: Выполняя шаги алгоритма, можно было построить другой код, но все построенные коды были бы спектрально эквивалентны (с одним множеством длин кодовых слов).

Пример 4.

Для заданного спектра длин кодовых слов $L(V) = \langle 1,1,3 \rangle$ построим префиксный код 1) в алфавите $B = \{0,1\}$; 2) в алфавите $B = \{0,1,2\}$;

3) в алфавите $B = \{0,1,2,3\}$

Решение:

1) в алфавите $B = \{0,1\}$ не выполняется неравенство Макмиллана

$$\frac{1}{2} + \frac{1}{2} + \frac{1}{2^3} > 1$$

следовательно, префиксного кода не существует

2) в алфавите $B = \{0,1,2\}$

Проверим, выполняется ли неравенство Макмиллана:

$$\frac{1}{3} + \frac{1}{3} + \frac{1}{3^3} = \frac{19}{27} \leq 1$$

значит префиксный код существует

Например, Σ={$B_1 = 1, B_2 = 2, B_3 = 000$ } или Σ={$B_1 = 0, B_2 = 2, B_3 = 110$ }. Более подробно покажем, как получили Σ={$B_1 = 0, B_2 = 2, B_3 = 110$ } (по алгоритму 1):

Шаг 1. Строим множество $V_1 \subseteq \{0,1,2\}$, $|V_1| = 2$ (так как две единицы в спектре $L(V) = \langle 1,1,3 \rangle$).

Пусть $V_1 = \{0, 2\}$.

Шаг 2. Строим множество $V_2 \subseteq \{0,1,2\}^3 / (V_1 \times \{0,1,2\}^{3-1})$, $|V_2| = 1$ (так как одна тройка в спектре $L(V) = \langle 1,1,3 \rangle$).

Пусть $V_2 = \{110\}$.

Искомый код $V = V_1 \cup V_2 = \{0, 2, 110\}$ префиксный по построению.

3) в алфавите $B = \{0,1,2,3\}$

Проверим, выполняется ли неравенство Макмиллана:

$$\frac{1}{4} + \frac{1}{4} + \frac{1}{4^3} = \frac{33}{64} \leq 1,$$

значит префиксный код существует.

Например, Σ={$B_1 = 0, B_2 = 2, B_3 = 100$ } или Σ={$B_1 = 2, B_2 = 3, B_3 = 000$ }.

Алгоритм (2) К. Шеннона построения префиксного кода по набору длин:

Пусть задан набор чисел $l_1, \dots, l_n$, удовлетворяющих неравенству Макмиллана $\frac{1}{q^{l_1}} + \frac{1}{q^{l_2}} + \cdots + \frac{1}{q^{l_n}} \leq 1$.

В силу доказанной выше теоремы существует префиксный код со спектром $L(V) = \langle l_1, \dots, l_n \rangle$ – набором длин элементарных кодов.

Построим последовательность чисел $q_1, q_2, \dots q_k$ по следующим правилам:

q_1=0,

$$q_2 = q_1 + \frac{1}{2^{l_1}}$$

$$q_3 = q_2 + \frac{1}{2^{l_2}}$$

$$q_{i+1} = q_i + \frac{1}{2^{l_i}} \, (i = 1,2, \dots, m-1)$$

Очевидно, $0 \leq q_i < 1$ и q_i имеет единственное представление в виде двоичной дроби с l_i знаками после запятой:

$q_i = \sum_{j=1}^{l_i} c_j^{(i)} \cdot 2^{-l_i}$, где $c_j^{(i)} = 0$ или $c_j^{(i)} = 1$.

Рассмотрим код $V = \{ v_1, v_2 \dots v_n \}$, где $v_i = c_1^{(i)} c_2^{(i)} \dots c_{l_i}^{(i)}$.

Так как наборы длин упорядочены по неубыванию, при $h > i$ выполняются неравенства $l_h > l_i$ и $q_h > q_i + 2^{-l_i}$. Поэтому элементарный код v_h отличается от элементарного кода v_i в l_i первых разрядах. Следовательно, построенный код является префиксным.

Пример 5.

Пусть $L(V) = \langle 1,3,3,4,4,4,4 \rangle$. Построим префиксный код в алфавите $B = \{0,1\}$ со спектром длин $L(V) = \langle 1,3,3,4,4,4,4 \rangle$.

Неравенство Макмиллана проверено в примере 3.

$$\frac{1}{2} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^4} + \frac{1}{2^4} + \frac{1}{2^4} \leq 1$$

Построим последовательность чисел $q_1, q_2, q_3, q_4, q_5, q_6, q_7$, записывая их в двоичной системе счисления.

Указание: для перевода дробной части заданного десятичного числа в *Bin* проводим ряд последовательных умножений на 2. В полученном произведении на каждом шаге отделяем целую часть от дробной. Дробную часть вновь умножаем на 2 и т. д. Процесс заканчивается, если на очередном шаге дробная часть будет равна нулю. Здесь возможен случай, когда очередная дробная часть уже встречалась на предыдущих шагах, тогда мы получим бесконечную периодическую дробь. Выписываем полученные целые части в порядке их появления от первой до последней – это и будет искомая двоичная дробь.

$q_1 = 0{,}0$

$q_2 = 0 + 2^{-1} = 0{,}100$

$q_3 = 0{,}100 + 2^{-3} = 0{,}100$+0,001=0,101

$q_4 = 0{,}101 + 2^{-3} = 0{,}101$+0,001=0,1100

$q_5 = 0{,}1100 + 2^{-4} = 0{,}1100$+0,0001=0,1101

$q_6 = 0{,}1110 + 2^{-4} = 0{,}1101$+0,0001=0,1110

$q_7 = 0{,}1110 + 2^{-4} = 0{,}1110$+0,0001=0,1111

Построим схему алфавитного кодирования Σ, выбирая в качестве элементарного кода B_i последовательность из 0 и 1 длины l_i, образующую дробную часть числа q_i:

Σ=$\{B_1 = 0, B_2 = 100, B_3 = 101, B_4 = 1100, B_5 = 1101, B_6 = 1110, B_7 = 1111\}$

Нетрудно убедиться, что построенный код является префиксным, он спектрально эквивалентен коду, полученному в примере 3.

5. Оптимальное кодирование

Вопросы сжатия информации играют важную роль в информатике. Это связано с развитием вычислительной техники и средств связи и, как следствие, с необходимостью хранения и передачи больших объемов информации.

Одно из направлений теории кодирования, в котором изучаются вопросы сжатия информации, называется экономным кодированием.

Определение. Методы кодирования, которые позволяют построить (без потери информации) коды сообщений, имеющие меньшую длину по сравнению с исходным сообщением, называются *методами сжатия* (или упаковки) информации.

Качество сжатия обычно определяется коэффициентом сжатия, измеряется в процентах и показывает, насколько сжатое сообщение короче исходного.

Допустим, имеется некоторое сообщение, которое закодировано каким-то общепринятым способом и хранится в памяти компьютера. Например, текст в кодах ASCII. Заметим, что равномерное кодирование, используемое в кодах ASCII, не является оптимальным для текстов, так как в текстах обычно используется существенно меньше, чем 256 символов. Обычно это 60–70 символов, в зависимости от языка.

Несколько величин используются для вычисления эффективности алгоритмов сжатия.

1) Коэффициент сжатия определяется по формуле:

   $K_{\text{сжатия}} = \frac{\text{размер выходного файла}}{\text{размер входного файла}}$. Коэффициент 0.6 означает, что сжатые данные занимают 60% от исходного размера. Значения большие 1 говорят о том, что выходной файл больше входного (отрицательное сжатие).

2) Коэффициент сжатия принято измерять в bpb (bit per bit, бит на бит), так как он показывает, сколько в среднем понадобится бит сжатого файла для представления одного бита файла на входе.

3) Выражение $100 \cdot (1 - k)$, где k – коэффициент сжатия, тоже отражает качество сжатия. Его значение равное 60 означает, что в результате сжатия занимает на 60% меньше, чем исходный файл.

Возможности сжатия информации определяются ее вероятностными и структурными (синтаксическими) свойствами. С помощью вероятностей моделируются частоты появления различных букв в сообщениях, частоты фрагментов сообщений, частоты появления самих сообщений.

Дополнительные возможности для сжатия появляются, когда в качестве сообщений рассматриваются не любые последовательности символов, а только некоторые из них. Например, в текстах русского языка не могут встречаться фрагменты "ггг", "аь" и т. д., и это можно учитывать при кодировании.

Начало математическому исследованию вопросов экономного кодирования, учитывающего вероятностные и структурные свойства информации, было положено работой К. Шеннона «Математическая теория связи», опубликованной в 1948 году. При построении алгоритмов кодирования Шенноном учитывались главным образом вероятностные свойства сообщений, порождаемых источником с конечным числом состояний. Однако эти вероятностные свойства определялись как вероятностными свойствами состояний источника, так и синтаксическими (структурными) свойствами последовательностей символов, генерируемых источником.

Вопросы экономного кодирования с учетом структурных свойств информации рассматривались в работах Маркова А.А. Он изучал возможности сжатия сообщений, также генерируемых источниками с конечным числом состояний, при простом с алгоритмической точки зрения способе кодирования – алфавитном, или побуквенном кодировании.

Для осуществления возможности разделения кодовых слов применяется одна из следующих мер:

1) использование специальной буквы;

2) применение кодовых слов одинаковой длины без разделительных букв – равномерное кодирование;

3) кодовая таблица составляется так, чтобы никакое кодовое слово не являлось началом другого, более длинного.

Будем рассматривать алгоритмы построения эффективных взаимно однозначных кодирований при некоторых простейших предположениях относительно статистических свойств источника сообщений. При этом более эффективными считаются кодирования, у которых в среднем на одну букву сообщения приходиться меньшее число двоичных цифр.

При построении экономных кодов используется дополнительная информация о вероятностях появления букв в сообщениях.

Пусть на буквах алфавита $A = \{a_1, \dots, a_n\}$ задано распределение вероятностей $P = \{p_1, \dots, p_n\}, \quad p_i \geq 0, \quad \sum_{i=1}^{n} p_i = 1$ и пусть $V = \{v_1, v_2, \dots, v_n\}$ – свободный код.

Под стоимостью кодирования *f* понимается величина $C_f(P) = \sum_{i=1}^{n} p_i |v_i|$, здесь $|v_i|$ – длина элементарного кода буквы a_i.

Стоимость кодирования показывает, во сколько раз в среднем увеличивается длина кодируемого сообщения, т. е. число букв, приходящееся на одну кодируемую букву.

Требование экономичности заключается в том, что необходимо добиваться при кодировании минимальной средней длины кодового слова.

Пример 1.

Пусть $A = \{a_1, a_2, a_3, a_4\}$, $P = \{0.4, 0.25, 0.2, 0.15\}$

Рассмотрим две схемы алфавитного кодирования и определим для них стоимости кодирования.

$$f_1: \begin{cases} a_1 \to 00 \\ a_2 \to 01 \\ a_3 \to 10 \\ a_4 \to 11 \end{cases} \qquad f_2: \begin{cases} a_1 \to 1 \\ a_2 \to 01 \\ a_3 \to 000 \\ a_4 \to 001 \end{cases}$$

Для f_1 стоимость кодирования: $C_{f_1}(P) = 0.4 \cdot 2 + 0.25 \cdot 2 + 0.2 \cdot 2 + 0.15 \cdot 2 = 2$

Для f_2 стоимость кодирования:

$C_{f_2}(P) = 0.4 \cdot 1 + 0.25 \cdot 2 + 0.2 \cdot 3 + 0.15 \cdot 3 = 1.95$

Таким образом, стоимость кодирования может изменяться при переходе от одной схемы кодирования к другой.

Положим $\mathrm{C}^*(P) = minC_f(P)$. Код V^* со схемой f^* такой, что $C_{f^*}(P) = C^*(P)$ называется оптимальным для набора вероятностей P. Можно показать, что величина $C^*(P)$ достигается при некоторой схеме f^* и может быть определена как $min_f C_f(P)$.

Оптимальные коды дают в среднем минимальное увеличение длин слов при соответствующем кодировании.

Существует несколько способов, позволяющих получать коды с малой избыточностью, причем все они обладают следующими свойствами:

1) Более вероятным буквам источника соответствуют более короткие кодовые слова (принцип статистического кодирования).

2) Никакое кодовое слово не является началом другого, более длинного.

3) Все буквы алфавита, используемого для передачи по каналу, приблизительно равновероятны.

4) Символы в последовательности на выходе кодера практически независимы.

Рассмотрим алгоритмы построения оптимальных и близких к оптимальным кодов. Коды с одинаковым спектром длин элементарных кодов имеют одинаковую стоимость. Поэтому будем искать оптимальные коды в классе префиксных кодов.

5.1. Алгоритм Фано

Алгоритм Фано строит код, близкий к оптимальному.

Упорядоченный в порядке не возрастания вероятностей список букв делится на две последовательные части так, чтобы суммы вероятностей входящих в них букв как можно меньше отличались друг от друга. Буквам из первой части приписываем символ 0, а буквам из второй части – символ 1.

Далее точно так же поступаем с каждой из полученных частей, если она содержит хотя бы две буквы.

Этот дихотомический процесс продолжается до тех пор, пока весь список не разобьется на части, содержащие по одной букве. Каждой букве ставится в соответствие последовательность символов, приписанных в результате этого процесса данной букве.

Применяя алгоритм Фано, символы имеют коды неодинаковой длины. Принцип очевиден – часто встречающиеся символы кодируются меньшим числом бит, редко встречающиеся – большим. Построенный код является префиксным.

Пример 2. Для списка сообщений с заданным распределением частот построим код Фано.

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
0,08	0,1	0,15	0,15	0,3	0,05	0,12	0,05

Решение.

	Алгоритм	Реализация для данного примера
1	Упорядочить список сообщений по убыванию частот. При равенстве частот порядок может быть любым.	<table><tr><td>a_5</td><td>a_3</td><td>a_4</td><td>a_7</td><td>a_2</td><td>a_1</td><td>a_6</td><td>a_8</td></tr><tr><td>0,3</td><td>0,15</td><td>0,15</td><td>0,12</td><td>0,1</td><td>0,08</td><td>0,05</td><td>0,05</td></tr></table>
2	Разделить группу событий на две части, различающиеся возможно меньше по суммарной частоте.	Разделение на два множества: $\{a_5, a_3\}$ и $\{a_4, a_7, a_2, a_1, a_6, a_8\}$ дает соотношение частот 0,45:0,55 с разницей 0,1. Другое разделение на два множества: $\{a_5, a_3, a_4\}$ и $\{a_7, a_2, a_1, a_6, a_8\}$ дает соотношение частот 06:0,4 с большей разницей 0,2.
3	Установить первый	<table><tr><td>a_5</td><td>0,3</td><td>0</td></tr></table>

<table>
<tr><td></td><td>символ кода 0 для сообщений первой группы, 1 – для второй группы.</td><td>
<table>
<tr><td>a_3</td><td>0,15</td><td>0</td></tr>
<tr><td>a_4</td><td>0,15</td><td>1</td></tr>
<tr><td>a_7</td><td>0,12</td><td>1</td></tr>
<tr><td>a_2</td><td>0,1</td><td>1</td></tr>
<tr><td>a_1</td><td>0,08</td><td>1</td></tr>
<tr><td>a_6</td><td>0,05</td><td>1</td></tr>
<tr><td>a_8</td><td>0,05</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>Повторить процедуру п.3 деления на две части в каждой из групп до тех пор, пока в группе больше одного элемента, добавляя каждый раз к коду справа 0 для первой части подразбиения и 1 для второй части</td><td>
<table>
<tr><td rowspan="2">a_5 0.3
a_3 0.15 } 0,45</td><td colspan="2">0.3</td><td>00</td></tr>
<tr><td colspan="2">0.15</td><td>01</td></tr>
<tr><td rowspan="7">a_4 0.15
a_7 0.12
a_2 0.1
a_1 0.08
a_6 0.05
a_8 0.05 } 0,55</td><td rowspan="2">0.15
0.12 } 0,27</td><td>0.15</td><td>100</td></tr>
<tr><td>0.12</td><td>101</td></tr>
<tr><td rowspan="5">0.1
0.08
0.05
0.05 } 0,28</td><td>0.1</td><td>110</td></tr>
<tr><td rowspan="3">0.08
0.05
0.05 } 0,18   0.08
0.05
0.05 } 0.1</td><td>1110</td></tr>
<tr><td>11110</td></tr>
<tr><td>11111</td></tr>
</table>

Для четырех последних строк выбрано разделение $\{a_2\}$ и $\{a_1, a_6, a_8\}$ с соотношением частот 0,1:0,18;
Возможно другое разбиение:
$\{a_2, a_1\}$ и $\{a_6, a_8\}$ с соотношением частот 0,18:0,1, что даст другой вариант кодирования этих сообщений:
<table>
<tr><td>a_2</td><td>0,1</td><td>1100</td></tr>
<tr><td>a_1</td><td>0,08</td><td>1101</td></tr>
<tr><td>a_6</td><td>0,05</td><td>1110</td></tr>
<tr><td>a_8</td><td>0,05</td><td>1111</td></tr>
</table>
</td></tr>
<tr><td>5</td><td>Определить стоимость построенного кода.</td><td>Для первого варианта кода стоимость равна
$2 \cdot (0{,}3 + 0{,}15) + 3 \cdot (0{,}15 + 0{,}12 + 0{,}1) + 4 \cdot 0{,}08 +$
$+5 \cdot (0{,}05 + 0{,}05) = 2{,}83$
Для второго варианта кода стоимость равна
$2 \cdot (0{,}3 + 0{,}15) + 3 \cdot (0{,}15 + 0{,}12) + 4 \cdot (0{,}1 + 0{,}08 + 0{,}05$
$+ 0{,}05) = 2{,}83$
В этом примере оба варианта имеют одинаковую стоимость.</td></tr>
</table>

5.2. Алгоритм Шеннона

Алгоритм Шеннона применим, когда все вероятности $p_i > 0$. Букве a_i ставится в соответствие последовательность из $l_i = \left\lceil log \frac{1}{p_i} \right\rceil$ двоичных символов (здесь ⌈x⌉ – ближайшее целое сверху числа x и log здесь и везде далее берется по основанию 2). Алгоритм Шеннона основан на том, что выбранные длины l_i ($i = 1, \dots, n$) удовлетворяют неравенству Макмиллана.

После выбора длин применятся рассмотренный ранее алгоритм Шеннона построения префиксного кода по набору длин элементарных кодов.

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

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

Пример 3.

Пусть $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$,

$P = \{0.08, 0.1, 0.15, 0.15, 0.3, 0.05, 0.12,\ 0.05\}$.

Вычислим набор длин для P: $l_1 = \left\lceil log \frac{1}{0{,}08} \right\rceil = 4,\ l_2 = \left\lceil log \frac{1}{0{,}1} \right\rceil = 4,\ l_3 = l_4 = \left\lceil log \frac{1}{0{,}15} \right\rceil = 3, l_5 = \left\lceil log \frac{1}{0{,}3} \right\rceil = 2, l_6 = l_8 = \left\lceil log \frac{1}{0{,}05} \right\rceil = 5, l_7 = \left\lceil log \frac{1}{0{,}12} \right\rceil = 4.$

Построим префиксный код по алгоритму Шеннона с вычисленными длинами элементарных кодов. Построим префиксный код в алфавите $B = \{0,1\}$ со спектром длин $L(V) = \langle 2,3,3,4,4,4,5,5 \rangle$

Построим последовательность чисел $q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8$, записывая их в двоичной системе счисления.

$q_1 = 0{,}00$

$q_2 = 0{,}00 + 2^{-2} = 0{,}010$

$q_3 = 0{,}010 + 2^{-3} = 0{,}010+0{,}001=0{,}011$

$q_4 = 0{,}011 + 2^{-3} = 0{,}011+0{,}001=0{,}1000$

$q_5 = 0{,}1000 + 2^{-4} = 0{,}1000+0{,}0001=0{,}1001$

$q_6 = 0{,}1001 + 2^{-4} = 0{,}1001+0{,}0001=0{,}1010$

$q_7 = 0{,}1010 + 2^{-4} = 0{,}1010+0{,}0001=0{,}10110$

$q_8 = 0{,}10110 + 2^{-5} = 0{,}1011+0{,}00001=0{,}10111$

Построим префиксный код Σ, выбирая в качестве элементарного кода B_i последовательность из 0 и 1 длины l_i, образующую дробную часть числа q_i:

$$\Sigma: \begin{cases} a_1 \longrightarrow B_1 = 1000 \\ a_2 \longrightarrow B_2 = 1001 \\ a_3 \longrightarrow B_3 = 010 \\ a_4 \longrightarrow B_4 = 011 \\ a_5 \longrightarrow B_5 = 00 \\ a_6 \longrightarrow B_6 = 10110 \\ a_7 \longrightarrow B_7 = 1010 \\ a_8 \longrightarrow B_8 = 10111 \end{cases}$$

Стоимость кодирования равна:

$C_\Sigma(P) = 4 \cdot (0{,}08 + 0{,}1) + 3 \cdot (0{,}15 + 0{,}15) + 2 \cdot 0{,}3 + 5 \cdot (0{,}05 + 0{,}05) +$

$+4 \cdot 0{,}12 = 3{,}2.$

5.3. Алгоритм Хаффмана

Алгоритм Хаффмана – адаптивный жадный алгоритм оптимального префиксного кодирования алфавита с минимальной избыточностью. Был разработан в 1952 году аспирантом Массачусетского технологического института Дэвидом Хаффманом.

Жадный алгоритм (Greedy algorithm) – алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

Сжатие данных по Хаффману применяется при сжатии фото и видеоизображений (JPEG, стандарты сжатия MPEG), в архиваторах (PKZIP, LZH и др.), в протоколах передачи данных MNP5 и MNP7.

Идея алгоритма Хаффмана состоит в следующем: зная вероятности символов в сообщении, можно описать процедуру построения кодов переменной длины, состоящих из целого количества битов. Символам с большей вероятностью ставятся в соответствие более короткие коды.

Коды Хаффмана обладают свойством префиксности (то есть ни одно кодовое слово не является префиксом другого), что позволяет однозначно их декодировать.

При построении оптимальных кодов можно применять как традиционный табличный способ кодирования, так и использовать «кодовые деревья».

Классический алгоритм Хаффмана на входе получает таблицу частот встречаемости символов в сообщении. Далее на основании этой таблицы строится дерево кодирования Хаффмана (Н-дерево).

1. Символы входного алфавита образуют список свободных узлов. Каждый лист имеет вес, который может быть равен либо вероятности, либо количеству вхождений символа в сжимаемое сообщение.

2. Выбираются два свободных узла дерева с наименьшими весами.

3. Создается их родитель с весом, равным их суммарному весу.

4. Родитель добавляется в список свободных узлов, а два его потомка удаляются из этого списка.

5. Одной дуге, выходящей из родителя, ставится в соответствие бит 1, другой – бит 0.

6. Шаги, начиная со второго, повторяются до тех пор, пока в списке свободных узлов не останется только один свободный узел. Он и будет считаться корнем дерева.

7. Прослеживаем путь к каждому листу дерева, помечая направление к каждому узлу (например, направо – 1, налево – 0). Полученная

последовательность дает кодовое слово, соответствующее каждому символу. Сообщения источника являются теперь концевыми узлами кодового дерева.

Алгоритм построения кода Хаффмана по таблице:

Пусть в исходном упорядоченном по не возрастанию списке вероятностей две последние вероятности p_n и p_{n-1}. Эти вероятности из списка исключаются, а их сумма вставляется в список таким образом, чтобы в получившемся новом списке вероятности не возрастали. Эта процедура повторяется до тех пор, пока не получиться список из двух вероятностей. После получения списка из двух вероятностей одной из них приписывается символ 0, а другой – символ 1 (оптимальный код для двухбуквенного алфавита сообщений при любом распределении вероятностей). Затем из оптимального кода для двух букв строиться оптимальный код для трех букв и т. д. Продолжая этот процесс, придем к искомому оптимальному коду для *n* букв.

Пример 4.

Пусть $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$,

$P = \{0.08, 0.1, 0.15, 0.15, 0.3, 0.05, 0.12, \ 0.05\}$.

a_5	0.3	0.3	0.3	0.3	0.3	**0.4**	**0.6**		0							00
a_3	0.15	0.15	**0.18**	**0.22**	**0.3**	0.3	0.4		1	00						010
a_4	0.15	0.15	0.15	0.18	0.22	0.3				01	10					011
a_7	0.12	0.12	0.15	0.15	0.18						11	010				100
a_2	0.1	0.1	0.12	0.15								011	110			110
a_1	0.08	**0.1**	0.1										111	100		111
a_6	0.05	0.08												101	1010	1010
a_8	0.05														1011	1011

Согласно алгоритму, будем последовательно складывать два наименьших элемента (две наименьшие вероятности) и заносить их в массив так, чтобы он остался отсортированным. В таблице проиллюстрирована эта процедура: сначала складываем 0,05 и 0,05 и

заносим в таблицу 0,1 (выделяем жирным), затем складываем 0,1 и 0,08 и заносим в таблицу 0,18 (выделено жирным) и так далее, пока не останется только два элемента.

Этап кодирования: последним двум оставшимся элементам присваиваем коды 0 и 1 соответственно. Затем возвращаемся на шаг назад, «разворачиваем» элемент 0,6 и полученным из него элементам 0,3 и 0,3 присваиваем код 0,6 с дописанным 0 и 1 соответственно, у элемента 0,4 код остается таким же, как на первом шаге – 1. Теперь повторяем этот шаг, «разворачивая» подобным образом выделенные жирным элементы. Результат этих действий приведен в правой таблице. В ней же в последнем столбце получаем искомый код.

Стоимость кодирования равна:

$C_{\Sigma}(P) = 2 \cdot 0{,}3 + 3 \cdot (0{,}15 + 0{,}15 + 0{,}12 + 0{,}1 + 0{,}08) + 4 \cdot (0{,}05 + 0{,}05) = 2{,}8$

– это средняя длина кодового слова.

Для оценки эффективности кода используют коэффициент эффективности: $\gamma = \frac{H(P)}{C_{\Sigma}(P)}$.

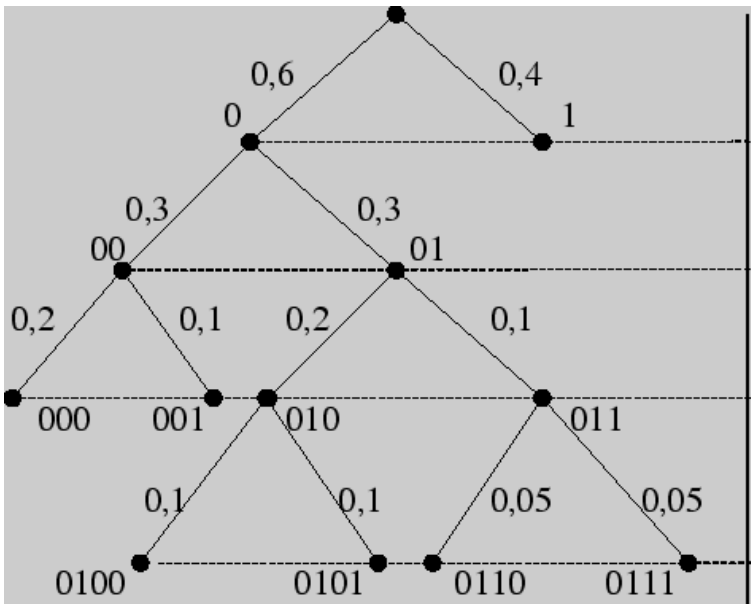
Энтропия источника сообщений: $H(P) = -\sum_{i=1}^{n} p_i \cdot log p_i$.

Коды Хаффмана являются оптимальными и префиксными.

Пример 5. Закодировать по Фано сообщения, имеющие следующие вероятности:

сообщение	1	2	3	4	5	6	7
вероятность	0,4	0,2	0,1	0,1	0,1	0,05	0,05

Решение (с использованием кодового дерева):



Коды	Сообщения
1	1
000	2
001	3
0100	4
0101	5
0110	6
0111	7

Построенный код является префиксным и ему соответствует насыщенное кодовое дерево (Дерево называется *насыщенным*, если из каждой вершины, не являющейся листом, в следующий ярус выходит ровно два ребра).

Листья кодового дерева представляют собой кодируемые сообщения с присвоенными им кодовыми словами.

Цена кодирования (средняя длина кодового слова) является критерием степени оптимальности кодирования.

$C_{\Sigma}(P) = 1 \cdot 0{,}4 + 3 \cdot 0{,}2 + 3 \cdot 0{,}1 + 4 \cdot (0{,}1 \cdot 2 + 0{,}05 \cdot 2) = 2{,}5.$

Энтропия источника сообщения:

$H(P) = -(0{,}4 \cdot log_2(0{,}4) + 0{,}2 \cdot log_2(0{,}2) + 3 \cdot 0{,}1 \cdot log_2(0{,}1) + 2 \cdot 0{,}05 \cdot log_2(0{,}05)) = 2{,}42$ бит на одну букву на выходе.

Полученный код имеет коэффициент эффективности: $\gamma = \frac{2{,}422}{2{,}5} = 0{,}9885$. Определив избыточность по формуле $1 - \gamma = 1 - 0{,}9885 = 0{,}011458$, видим, что возможно сокращение длины кода на 1,15%. Код близок к оптимальному, так как для оптимального двоичного кода $H(P) = C_{\Sigma}(P)$ и $\gamma = 1$.

Метод Хаффмана производит идеальное сжатие *(то есть, сжимает данные до их энтропии),* если вероятности символов точно равны отрицательным степеням числа 2. Результаты эффективного кодирования по методу Хаффмана всегда лучше результатов кодирования по методу Шеннона-Фано.

Пример 6.

Закодировать сообщения из предыдущего примера по методу Хаффмана.

Решение (табличный способ).

Первый шаг:

сообщения	p	p_1	p_2	p_3	p_4	p_5
1	0,4	0,4	0,4	0,4	0,4	0,6
2	0,2	0,2	0,2	0,2	0,4	0,4
3	0,1	0,1	0,2	0,2	0,2	
4	0,1	0,1	0,1	0,2		
5	0,1	0,1	0,1			
6	0,05	0,1				
7	0,05					

Вторым шагом производим кодирование, проходя по таблице справа налево (обычно это проделывается в одной таблице):

	A	A_1	A_2	A_3	A_4	A_5
1	0,4	0,4	0,4	0,4	0,4	0,6 0
2	0,2	0,2	0,2	0,2	0,4 00	0,4 1
3	0,1	0,1	0,2	0,2 000	0,2 01	
4	0,1	0,1	0,1 0010	0,2 001		
5	0,1	0,1 0000	0,1 0011			
6	0,05 00010	0,1 0001				
7	0,05 00011					

Решение (способ построения дерева).

Алгоритм построения орграфа Хаффмана:

1. Построение кодового дерева начинается не с корня, а с листьев. Символы исходного алфавита образуют вершины-листья. каждой вершине приписывается вес, равный количеству вхождений данного символа в сжимаемое сообщение (т. е. частота вхождения или вероятность).

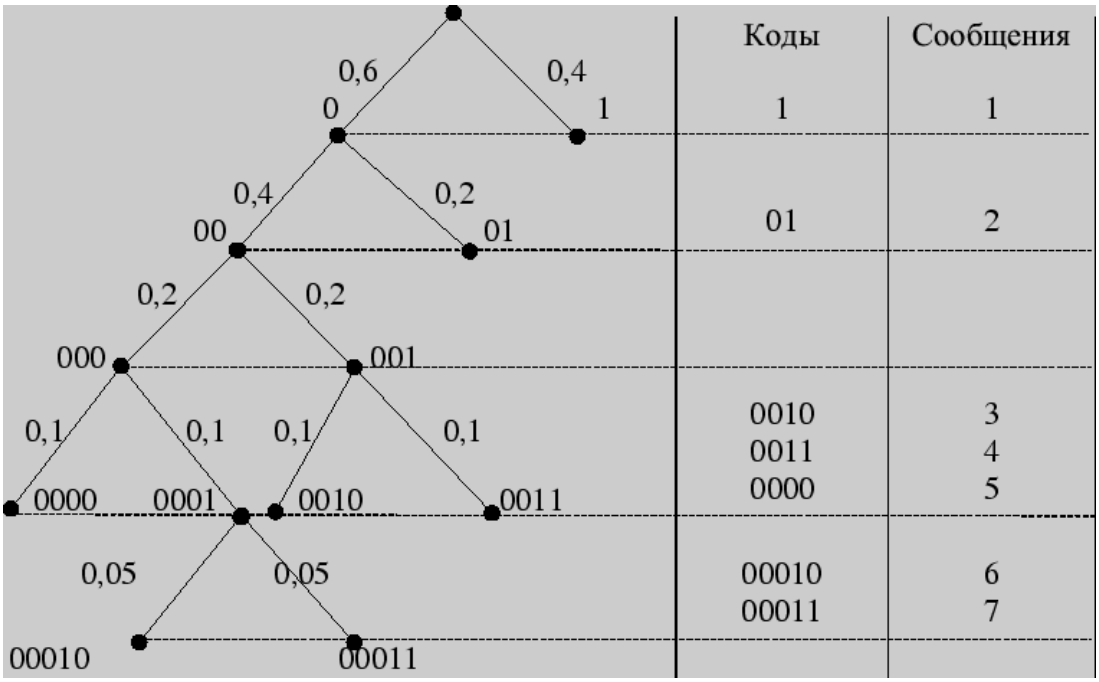
2. Среди этих вершин выбираются две с наименьшими весами (если таких пар несколько, выбирается любая из них).

3. Создается следующая вершина графа, из которой выходят две дуги к выбранным вершинам; одна помечается цифрой 0, другая символом – 1. Созданной вершине приписывается вес, равный сумме весов, выбранных на втором шаге вершин, а веса этих двух вершин стираются.

4. К вершинам, которым приписаны веса, применяются шаги 2 и 3 до тех, пор пока не останется одна вершина с весом равным сумме весов исходных символов.

5. Завершается алгоритм спуском по дереву и построением кодов всех символов.

Дерево, полученное в результате кодирования по Хаффману, имеет следующий вид:



Листья кодового дерева представляют собой кодируемые сообщения с присвоенными им кодовыми словами.

Таблица кодов имеет вид:

сообщение	1	2	3	4	5	6	7
код	1	01	0010	0011	0000	00010	00011

сообщение	1	2	3	4	5	6	7
вероятность	0,4	0,2	0,1	0,1	0,1	0,05	0,05

Цена кодирования здесь будет равна:

$$C_{\Sigma}(P) = 1 \cdot 0{,}4 + 2 \cdot 0{,}2 + 4 \cdot (0{,}1 \cdot 3) + 5 \cdot (0{,}05 \cdot 2) = 2{,}5.$$

В итоге получается, что символам с большей вероятностью появления соответствуют более короткие коды.

Для передачи данных сообщений можно перейти от побуквенного (поцифрового) кодирования к кодированию «блоков», состоящих из фиксированного числа последовательных букв.

Пример 7. Провести кодирование по методу Фано двухбуквенных комбинаций, когда алфавит состоит из двух букв А и Б, имеющих вероятности Р(А)=0.8 и Р(Б)=0.2.

Решение.

комбинации букв	вероят-ность	разбиение на подгруп-пы	код
АА	0,64	I	0
АБ	0,16	I	10
БА	0,16	II I	110
ББ	0,04	II II	111

Цена кода (стоимость кодирования одного блока)

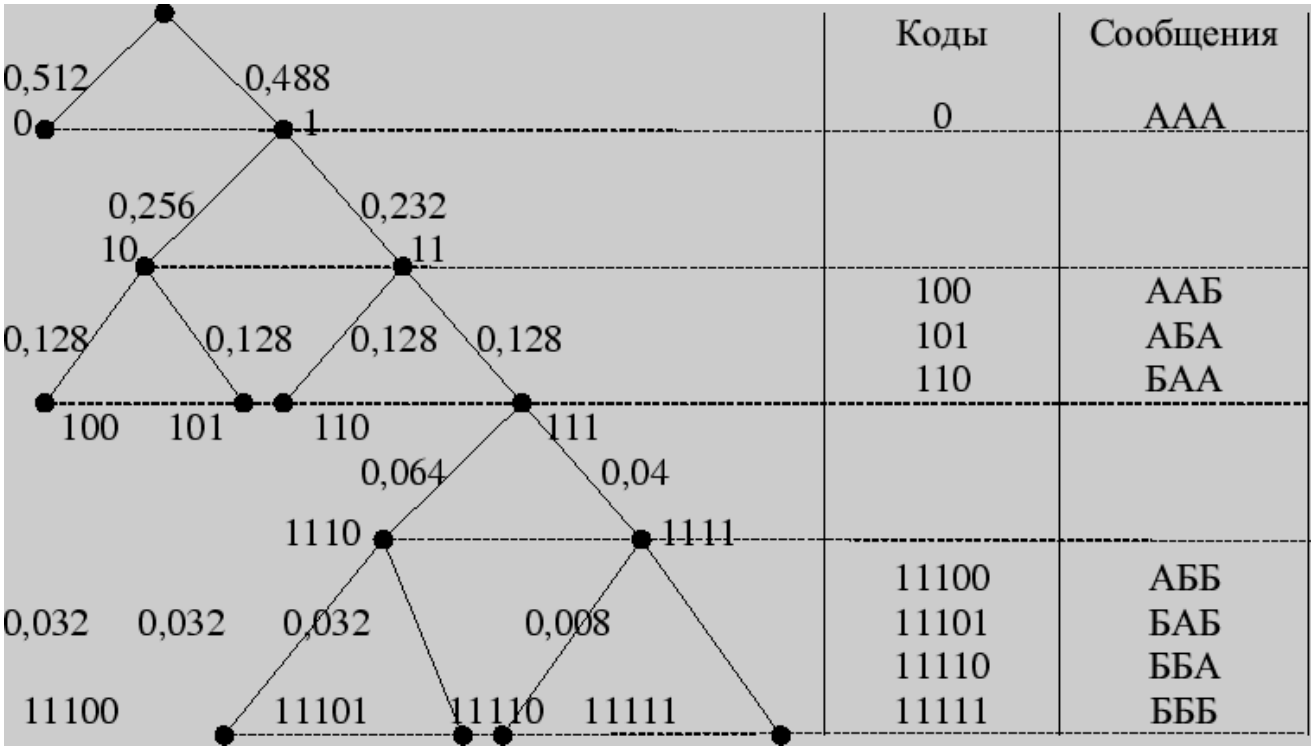
$C_{\Sigma}(P) = 1 \cdot 0{,}64 + 2 \cdot 0{,}16 + 3 \cdot (0{,}16 + 0{,}04) = 1{,}56.$

Стоимость кодирования одной буквы $\frac{C_{\Sigma}(P)}{2} = 0{,}78.$

Пример 8. Провести кодирование по Хаффману трехбуквенных комбинаций для алфавита из предыдущего примера.

комбина-ция букв	p	p_1	p_2	p_3	p_4	p_5	p_6
ААА	0,512	0,512	0,512	0,512	0,512	0,512	0,512
ААБ	0,128	0,128	0,128	0,128	0,232	0,256	0,488
АБА	0,128	0,128	0,128	0,128	0,128	0,232	
БАА	0,128	0,128	0,128	0,128	0,128		
АББ	0,032	0,04	0,064	0,104			
БАБ	0,032	0,032	0,04				
ББА	0,032	0,032					
ББББ	0,008						

Построим кодовое дерево.



Найдем цену кодирования:

$$C_{\Sigma}(P) = 1 \cdot 0{,}512 + 3 \cdot (0{,}128 \cdot 3) + 5 \cdot (0{,}032 \cdot 3 + 0{,}008) = 2{,}184.$$

Стоимость кодирования одной буквы $\frac{C_{\Sigma}(P)}{3} = 0{,}728$.

Большей оптимизации кодирования можно достичь еще и использованием трех символов – 0, 1, 2 вместо двух.

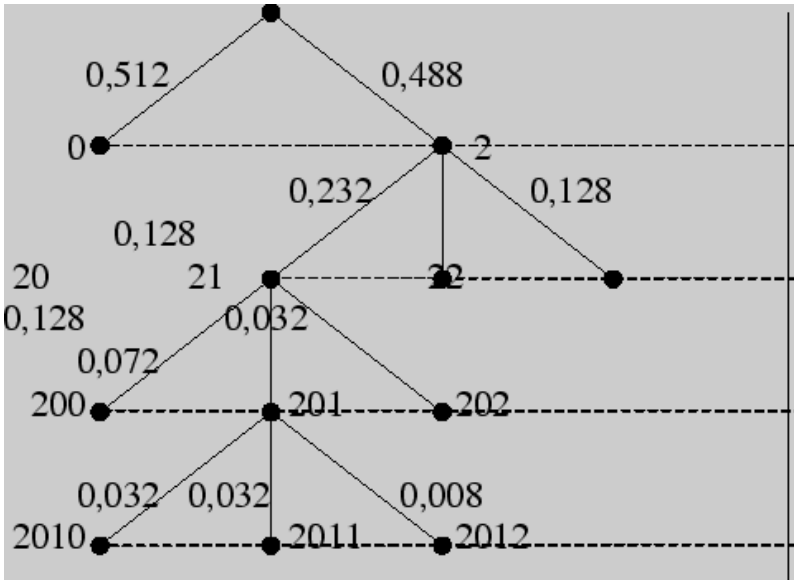
Тринарное дерево Хаффмана получается, если суммируются не две наименьшие вероятности, а три, и приписываются каждому левому ребру 0, правому – 2, а серединному – 1.

Пример 9. Построить тринарное дерево Хаффмана для кодирования трехбуквенных комбинаций из примера 8.

Решение. Составим таблицу тройного «сжатия» по методу Хаффмана

Сообщение	p	p_1	p_2	p_3
ААА	0,512	0,512	0,512	0,512
ААБ	0,128	0,128	0,232	0,488
АБА	0,128	0,128	0,128	
БАА	0,128	0,128	0,128	
АББ	0,032	0,072		
БАБ	0,032	0,032		
ББА	0,032			
БББ	0,008			

Тогда тринарное дерево выглядит следующим образом:



Коды	Сообщения
0	ААА
21	ААБ
22	АБА
200	БАА
202	АББ
2010	БАБ
2011	ББА
2012	ББББ

$C_{\Sigma}(P) = 1 \cdot 0{,}512 + 3 \cdot (0{,}128 \cdot 3 + 0{,}032 \cdot 3) + 3 \cdot 0{,}008 = 1{,}976.$

Стоимость кодирования одной буквы $\frac{C_{\Sigma}(P)}{3} = 0{,}659$

Эффективное кодирование устраняет избыточность, приводит к сокращению длины сообщений, а значит, позволяет уменьшить время передачи или объем памяти, необходимой для их хранения.

Важную роль для оценки эффективности кодирования играет энтропия вероятностного распределения:

$$H(P) = -\sum_{i=1}^{n} p_i \cdot log p_i$$

Пусть $C^*(P)$ – стоимость оптимального алфавитного кодирования.

Теорема. $H(P) \leq C^*(P) \leq H(P) + 1$.

При блочном кодировании, выбирая длину блока N достаточно большой, можно сделать стоимость кодирования на одну букву сообщения $C_N(P)$ сколь угодно близкой к H(P).

Сравним эффективность построенных кодов в примерах 7-8.

$H(P) = -0.8 \cdot log 0.8 - 0.2 \cdot log 0.2 = 0.723$

В примере 7 стоимость кодирования одной буквы $\frac{C_{\Sigma}(P)}{2} = 0{,}78.$

В примере 8 стоимость кодирования одной буквы $\frac{C_\Sigma(P)}{3} = 0{,}728$, т. е. при увеличении длины блока стоимость кодирования одной буквы приближается к $H(P)$.

Поскольку при построении дерева кода Хаффмана может возникнуть некоторый произвол, для выбора оптимального варианта кода используют дисперсию. Дисперсия показывает, насколько сильно отклоняются длины индивидуальных кодов от их средней величины. Лучшим будет код с наименьшей дисперсией. Дисперсия рассчитывается по формуле:

$$D = \sum_s p_s \left(l_s - l_{\text{ср}}\right)^2$$

Для уменьшения дисперсии кода существует правило: когда на дереве имеется более двух узлов с наименьшей вероятностью, следует объединять символы с наибольшей и наименьшей вероятностью; это сокращает общую дисперсию кода.

Самой большой сложностью является необходимость иметь таблицы вероятностей для каждого типа сжимаемых данных. Это не представляет проблемы, если известно, что сжимается английский или русский текст; мы просто предоставляем кодеру и декодеру подходящее для английского или русского текста кодовое дерево. В общем же случае, когда вероятность символов для входных данных неизвестна, статические коды Хаффмана работают неэффективно.

Решением этой проблемы является статистический анализ кодируемых данных, выполняемый в ходе первого прохода по данным, и составление на его основе кодового дерева. Собственно кодирование при этом выполняется вторым проходом.

Еще один недостаток кодов – это то, что минимальная длина кодового слова для них не может быть меньше единицы, тогда как энтропия сообщения вполне может составлять и 0.1, и 0.01 бит/букву. В этом случае код становится существенно избыточным. Проблема решается применением алгоритма к

блокам символов, но тогда усложняется процедура кодирования/декодирования и значительно расширяется кодовое дерево, которое нужно в итоге сохранять вместе с кодом.

Данные коды никак не учитывают взаимосвязей между символами, которые присутствуют практически в любом тексте. Например, если в тексте на английском языке нам встречается буква q, то с уверенностью можно сказать, что после нее будет идти буква u.

Задания

1. С помощью неравенства Макмиллана выясните, может ли набор чисел L быть набором длин элементарных кодовых слов взаимно-однозначного кода: 1) L = {1, 2, 2, 3}; 2) L = {2, 2, 2, 4, 4, 4}; 3) L = {1, 2, 4, 4, 4, 4}; 47 4) L = {2, 2, 3, 4, 4}; 5) L = {1, 2, 3, 4, 4, 5}.
2. Для заданного распределения частот проведите кодирование по методам Фано, Шеннона и Хаффмана. Сравните их эффективность, вычислив стоимости кодов.

a) $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$, P $= \{0.2, 0.2, 0.19, 0.12, 0.11, 0.09,\ 0.09\}$.

b) при кодировании алфавита из пяти букв, равновероятно встречающихся.

c) при кодировании алфавита из десяти букв, которые встречаются с вероятностями 0,3; 0,2; 0,1; 0,1; 0,1; 0,05; 0,05; 0,04; 0,03; 0,03.

3. Сообщение состоит из последовательности трех букв A, B и C, вероятности появления которых не зависят от предыдущего сочетания букв и равны P_A:=0.7, P_B :=0.2, и P_C :=0.1.

   Провести кодирование по алгоритму Фано и Хаффмана отдельных букв, двухбуквенных и трехбуквенных сочетаний. Сравнить коды по их эффективности.

4. Источник статистически независимых сообщений задан таблицей:

x_j	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_7	x_7
$p(x_j)$	1/4	1/4	1/8	1/8	1/16	1/16	1/16	1/32	1/64	1/64

Закодировать сообщения источника двоичным кодом так, чтобы средняя длина кодового слова была минимальна. Оценить эффективность полученного кода.

5. Вероятности символов исходных сообщений заданы таблицей:

x_j	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
$p(x_j)$	0,2	0,25	0,15	0,1	0,05	0,05	0,14	0,06

Сообщения источника статистически независимы. Передача производится по двоичному каналу. Определить скорость передачи информации при использовании равномерного кода и кода Шеннона-Фано. Сравнить коды по их эффективности.

6. В качестве исходной строки текста выбрать «Фамилия Имя Отчество» студента. Сформировать алфавит фразы, посчитать количество вхождений символов и их вероятности появления.

Используя алгоритмы Фано, Шеннона и Хаффмана нужно определить коды символов. Вычислить стоимость кода для каждого алгоритма. Закодировать исходную строку. Рассчитать коэффициенты сжатия относительно кодировки ASCII и относительно равномерного кода.

6. Помехоустойчивое кодирование

Назначение помехоустойчивого кодирования – защита информации от помех и ошибок при передаче и хранении информации. При передаче информации по каналу связи возникают помехи, ошибки и небольшая часть информации теряется. Помехоустойчивое кодирование необходимо для устранения ошибок, которые возникают в процессе передачи, хранения информации.

Без использования помехоустойчивого кодирования было бы невозможно передавать большие объемы информации (файлы), т. к. в любой системе передачи и хранении информации неизбежно возникают ошибки.

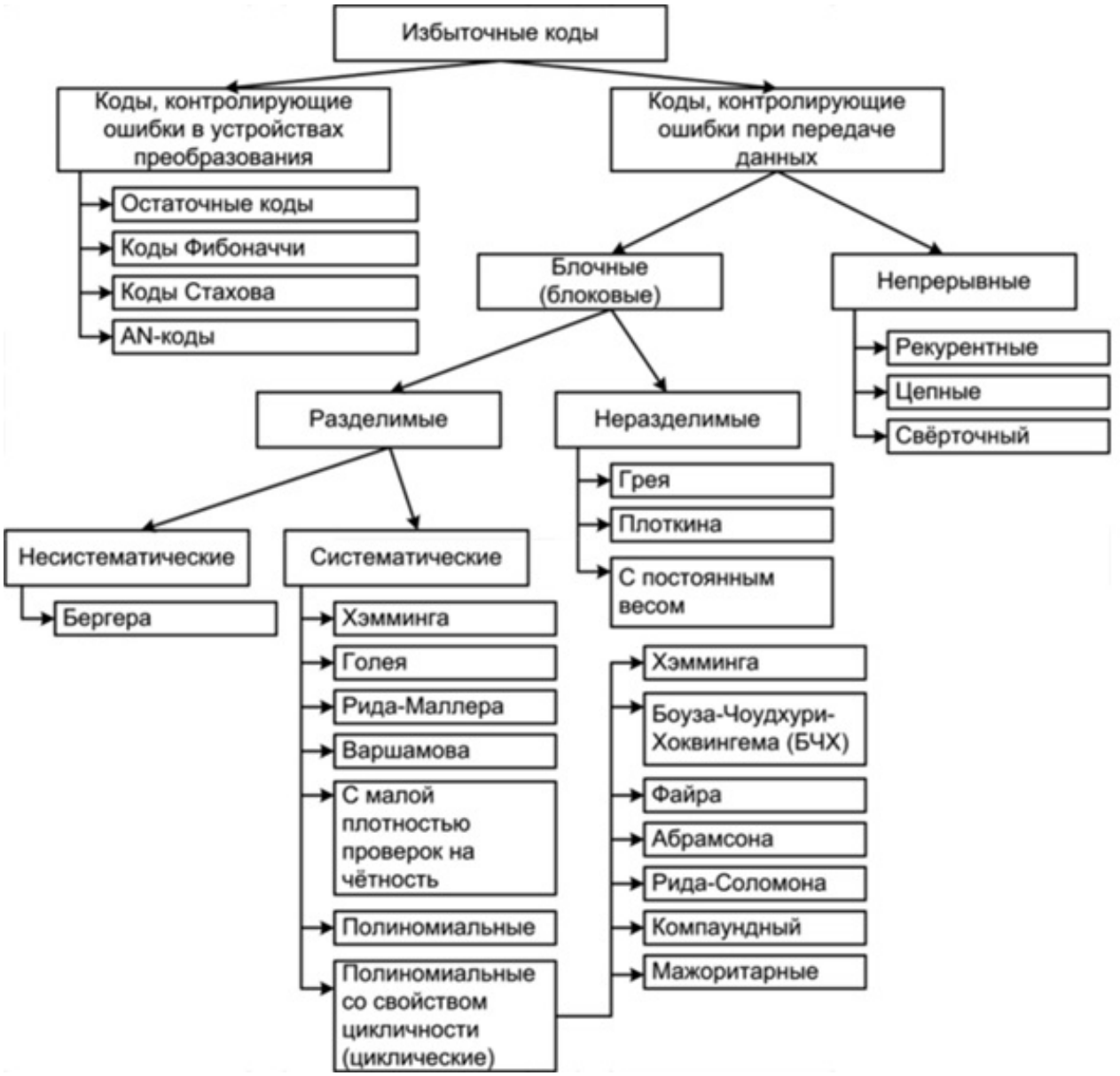
В зависимости от того используется в системе обнаружение или исправление ошибок с помощью помехоустойчивого кода, различают следующие варианты:

- запрос повторной передачи (Automatic Repeat reQuest, ARQ): с помощью помехоустойчивого кода выполняется только обнаружение ошибок, при их наличии производится запрос на повторную передачу пакета данных;
- прямое исправление ошибок (Forward Error Correction, FEC): производится декодирование помехоустойчивого кода, т. е. исправление ошибок с его помощью.

Возможен также гибридный вариант, чтобы лишний раз не гонять информацию по каналу связи, например получили пакет информации, попробовали его исправить, и если не смогли исправить, тогда отправляется запрос на повторную передачу.

Любое помехоустойчивое кодирование добавляет избыточность, за счет чего и появляется возможность восстановить информацию при частичной потере данных в канале связи (носителе информации при хранении). В случае эффективного оптимального кодирования убирали избыточность, а в помехоустойчивом кодировании добавляется контролируемая избыточность.

6.1. Классификация помехоустойчивых кодов



- *Непрерывные* – процесс кодирования и декодирования носит непрерывный характер. Сверточный код является частным случаем непрерывного кода. На вход кодера поступил один символ, соответственно, появилось несколько на выходе, т. е. на каждый входной символ формируется несколько выходных, так как добавляется избыточность.
- *Блочные* (блоковые) – процесс кодирования и декодирования осуществляется по блокам. С точки зрения понимания работы, блочный код проще, разбиваем код на блоки и каждый блок кодируется в отдельности.

По используемому алфавиту:

- *Двоичные о*перируют битами.

- *Не двоичные* (код Рида-Соломона). Оперируют более размерными символами. Если изначально информация двоичная, нужно эти биты превратить в символы. Например, есть последовательность 110 110 010 100 и нужно их преобразовать из двоичных символов в не двоичные, берем группы по 3 бита — это будет один символ, 6, 6, 2, 4 — с этими не двоичными символами работают не двоичные помехоустойчивые коды.

Блочные коды делятся на:

- *Систематические* – отдельно не измененные информационные символы, отдельно проверочные символы. Если на входе кодера присутствует блок из k символов, и в процессе кодирования сформировали еще какое-то количество проверочных символов, то проверочные символы ставят рядом к информационным в конец или в начало. Выходной блок на выходе кодера будет состоять из информационных символов и проверочных.
- *Несистематические* – символы исходного сообщения в явном виде не присутствуют. На вход пришел блок k, на выходе получили блок размером n. В случае систематических кодов выходной блок в явном виде содержит в себе то, что пришло на вход, а в случае несистематического кода, глядя на выходной блок нельзя понять, что было на входе.

Смотря на картинку выше, код 1 1 0 0 0 1 0 0 | 1 является систематическим, на вход поступило 8 бит, а на выходе кодера 9 бит, которые в явном виде содержат в себе 8 бит информационных и один проверочный.

Параметры помехоустойчивого кодирования:

Первый параметр, *скорость кода R* характеризует долю информационных («полезных») данных в сообщении и определяется выражением:

$R = m/n = m/(m+k)$,

- где n – количество символов закодированного сообщения (результата кодирования);
- k – количество проверочных символов, добавляемых при кодировании;
- m – количество информационных символов.

Параметры n и m часто приводят вместе с наименованием кода для его однозначной идентификации. Например, код Хэмминга (7,4) значит, что на вход кодера приходит 4 символа, на выходе 7 символов.

Второй параметр *кратность обнаруживаемых ошибок* – количество ошибочных символов, которые код может обнаружить.

Третий параметр *кратность исправляемых ошибок* – количество ошибочных символов, которые код может исправить (обозначается буквой t).

Избыточность кода: $R = k / m$.

В соответствии с этим соотношением можно сформулировать основную проблему кодирования: необходимо найти коды с максимальной возможной скоростью передачи информации при минимальной избыточности кода. Это задача оптимизации.

Самый простой метод помехоустойчивого кодирования – это добавление одного бита четности. Есть некое информационное сообщение, состоящее из 8 бит, добавим девятый бит.

Если нечетное количество единиц, добавляем 0: 1 0 1 0 0 1 0 0 | **0.**

Если четное количество единиц, добавляем 1: 1 1 0 **1** 0 1 0 0 | **1.**

Если принятый бит чётности не совпадает с рассчитанным битом чётности, то считается, что произошла ошибка: 1 1 0 **0** 0 1 0 0 | **1**.

Под кратностью понимается, всевозможные ошибки, которые можно обнаружить. В этом случае кратность исправляемых ошибок 0, так как мы не можем исправить ошибки, а кратность обнаруживаемых 1.

Есть последовательность 0 и 1, и из этой последовательности составим прямоугольную матрицу размера 4 на 4. Затем для каждой строки и столбца посчитаем бит четности.

Прямоугольный код – код с контролем четности, позволяющий исправить одну ошибку:

сформированное сообщение

1	1	0	0	1
0	1	1	1	0
0	0	0	1	0
1	1	1	1	1
1	0	1	0	1

принятое сообщение

1	1	0	0	1
0	1	1	1	0
0	1	0	1	0
1	1	1	1	1
1	0	1	0	1

И если в процессе передачи информации допустим ошибку (ошибка ноль вместо единицы, выделено желтым цветом), начинаем делать проверку. Нашли ошибку во втором столбце и третьей строке. Чтобы исправить ошибку, просто инвертируем 1 в 0, тем самым ошибка исправляется.

Этот прямоугольный код исправляет все однобитные ошибки, но не все двухбитные и трехбитные.

Рассчитаем скорость кода для: 1 1 0 **0** 0 1 0 0 | **1**. Здесь R=8/9=0,88.

Для прямоугольного кода: R=16/24=0,66 (двадцать пятую единичку (бит четности) не учитываем).

Более эффективный с точки зрения скорости является первый вариант, но зато мы не можем с помощью него исправлять ошибки, а с помощью прямоугольного кода можно. Сейчас на практике прямоугольный код не используется, но логика работы многих помехоустойчивых кодов основана именно на прямоугольном коде.

Одним из важнейших параметров кода является кодовое расстояние.

Кодовым расстоянием или просто *расстоянием* кода Σ называется минимальное расстояние между двумя различными кодовыми словами, т. е.

$d_{\Sigma}=\min d(x,y)$ $x\neq y$ и $x,y\in\Sigma$.

Вес кодового слова – это число ненулевых символов этого слова.

Для двоичного кода Σ под расстоянием понимается расстояние Хэмминга.

Расстояние Хэмминга – число позиций, в которых соответствующие символы двух кодовых слов одинаковой длины различны.

Кратность исправляемых ошибок и обнаруживаемых, связано минимальным расстоянием Хэмминга. Любой помехоустойчивый код добавляет избыточность с целью увеличить минимальное расстояние Хэмминга. Именно минимальное расстояние Хэмминга определяет помехоустойчивость.

Например, d(010; 010)=0, так как все цифры в соответствующих позициях равны, а вот d(010101; 011011)=3.

Примеры:

1011001	25687914	пакет
1001101	24657934	багет
d=2	d=3	d=2

Теорема 1. Код Σ исправляет все комбинации из t или менее ошибок, если и только если кодовое расстояние не меньше, чем $2t+1$, т. е. $d_{\Sigma} \geq 2t+1$.

Теорема 2. Код Σ обнаруживает все комбинации из t или менее ошибок, если и только если кодовое расстояние не меньше, чем $t+1$, т.е. $d_{\Sigma} \geq t+1$.

Доказательство теорем основано на построении сфер с радиусом t вокруг каждого кодового слова. Для того чтобы каждая ошибка кратности t была исправлена, слово с такой ошибкой должно содержаться внутри сферы, описанной только вокруг одного кодового слова. Таким образом, расстояние между центрами двух сфер, т. е. различными кодовыми словами должно быть не меньше $2t+1$. Геометрическая иллюстрация приведена на рис. 7.

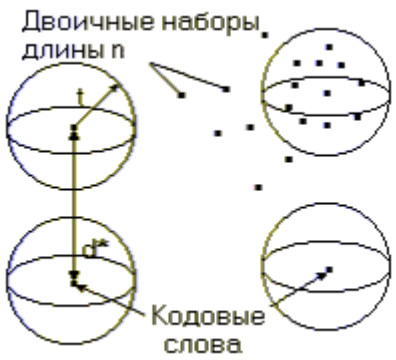


Рис 7. Сферы, центрами которых являются кодовые слова

Для того, чтобы каждая ошибка кратности *t* была обнаружена, слово с такой ошибкой не должно быть кодовым, т. е. расстояние между любыми различными кодовыми словами должно быть больше *t*.

Например, код с расстоянием 3 исправляет все одиночные ошибки и обнаруживает все двукратные ошибки. Код с расстоянием 4 обнаруживает все ошибки кратности 3 и менее, однако такой код по-прежнему исправляет только одну ошибку.

Пример 1. Дана схема кодирования:

$$\Sigma: \begin{cases} a_1 \to 00000 \\ a_2 \to 10101 \\ a_3 \to 01011 \\ a_4 \to 11110 \end{cases}$$

Матрица расстояний:

a_{ij}	1	2	3	4
1	0	5	4	4
1	5	0	5	5
3	4	5	0	4
4	4	5	4	0

$d_\Sigma = \min d_{ij} = 4$ – кодовое расстояние, тогда кратность обнаруживаемых ошибок $4 \geq t+1$, $t \leq 3$, а кратность исправляемых ошибок $4 \geq 2t+1$, $t \leq 1{,}5$. Следовательно, приведенный код позволяет обнаруживать всевозможные однократные, двукратные и трехкратные ошибки и исправлять любые однократные ошибки.

Определим избыточность кода, полагая буквы источника равновероятными. Для передачи равновероятных сигналов a_1, a_2, a_3, a_4 достаточно передавать кодовые слова 00, 10, 01 и 11 соответственно. Такой код не имеет избыточности, но не позволяет обнаруживать и, тем более, исправлять

ошибки. Для обнаружения и исправления ошибок введены пять избыточных символов, т.е. количественно избыточность кода равна $R = \frac{7-2}{7}$=71%.

Задача построения реального кода, т. е. кода, длина слов в котором достигает сотен символов, с заданным кодовым расстоянием не решается полным перебором даже с использованием современной вычислительной техники. Теория кодирования предлагает методы для построения таких кодов.

Пример 2.

Пусть мы кодируем 4 буквы следующим образом:

$A \to 10100$,

$B \to 01000$,

$C \to 00111$,

$D \to 11011$.

Чтобы найти минимальное расстояние между различными кодовыми словами, построим таблицу попарных расстояний.

	A	**B**	**C**	**D**
A	—	3	3	4
B	3	—	4	3
C	3	4	—	3
D	4	3	3	—

Минимальное расстояние $d_{min} = 3$, а значит $3 \geq 2t+1$, откуда получаем, что такой код может исправить до $t = 1$ ошибок. Обнаруживает же он две ошибки.

Рассмотрим пример: $A \to 10100 \quad \to 1011\underline{0}$.

Чтобы декодировать полученное сообщение, посмотрим, к какому символу оно ближе всего.

A: $d(10110; 10100) = 1$,

B: $d(10110; 01000) = 4$,

$C: d(10110; 00111) = 2,$

$D: d(10110; 11011) = 3.$

Минимальное расстояние получилось для символа A, значит вероятнее всего передавался именно он: $A \rightarrow 10100 \rightarrow 10\underline{1}10 \rightarrow A$.

6.2. Код Хэмминга

Коды, в которых возможно автоматическое исправление ошибок, называются самокорректирующимися. Код Хэмминга – наиболее известный из первых самоконтролирующихся и самокорректирующихся кодов.

Код Хемминга – двоичный код, он работает на алфавите {0; 1}. Поэтому количество возможных вариантов так называемых входных слов 2^m при m=5 количество возможных вариантов входных слов будет $2^5 = 32$.

Код Хемминга позволяет автоматически обнаруживать наиболее вероятные ошибки при передаче данных.

Пусть дано слово длины m. В кодовом слове должно быть $n = m + k$ разрядов, где m – количество информационных разрядов, k – количество контрольных разрядов.

Количество контрольных разрядов k должно быть выбрано так, чтобы удовлетворялось неравенство $2^{k-1} \leq n < 2^k$ или $m + k \leq 2^k - 1$.

Таким образом, зная длину закодированного сообщения n, число k можно найти по формуле ($k = [log_2(n)] + 1$).

Минимальные значения k при заданных значениях n, найденные в соответствии с этим неравенством, приведены в таблице 1:

Таблица 1

n	k_{min}
2-3	2
4-7	3
8-13	4

Если известна только длина исходного сообщения m, то число k можно найти по формуле $m + k \leq 2^k - 1$.

Минимальные значения k при заданных значениях m, найденные в соответствии с этим неравенством, приведены в таблице 2:

Таблица 2

m	k_{min}
2-4	3
5-11	4
12-26	5

Таблица 3

Число информационных разрядов	m	1	4	11	26	57	120	247	502	1013
Число проверочных разрядов	k	2	3	4	5	6	7	8	9	10
Полное число разрядов	n	3	7	15	31	63	127	255	511	1023

Для произвольного слова $A=a_1a_2..a_n$ положим:

$H(A) = a_1e_k(1)\oplus a_2e_k(2) \ldots \oplus a_ne_k(\mathrm{n})$, где $H(A)$ представляет собой вектор длины k, полученный в результате сложения векторов, являющихся двоичными записями (с помощью k цифр) номеров единичных символов слова A (например, $e_6(5) = 000101$).

Теорема 3. Код Хэмминга H_n, состоящий из всех слов $X=x_1x_2..x_n$ таких, что $H(\mathrm{X}) = (00 \ldots 0)$ является кодом с исправлением одного замещения.

Число элементов кода Хэмминга H_n равно $|H_n| = 2^{n-k}$, и позиции с номерами $2^0, 2^1, \ldots, 2^{k-1}$ являются проверочными, а все остальные информационными. Таким кодовым множеством можно кодировать все слова длины $m = n - k$.

Пример 3.

Для X=010101 и У=110100, n=6, k=3 ($k = [log_2(6)] + 1$).

$H(\mathrm{X}) = (010) \oplus (100) \oplus (110) = (000)$ (сложение по модулю 2).

$H(\mathrm{У}) = (001) \oplus (010) \oplus (100) = (111)$.

В рассмотренном примере 2 $X \in H_6$, $У \notin H_6$.

Пример 4.

Пусть в слове $X = 010101 \in H_6$ произошло замещение пятого символа, в результате получилось слово T=010111. Так как $H(T) = (010) \oplus (100) \oplus (101) \oplus (110) = (101)$, то числовое значение слова $H(T)$ равно 5, что и определяет номер замещенного символа.

Покажем на примерах, как проводиться кодирование по Хэммингу.

Пример 5.

Пусть кодируемое слово $A = 1100$, m=4, k=3 ($k = [log_2(5)] + 1$)). Закодированное сообщение длины n=m+k=4+3=7 будет иметь вид $p_1 p_2 1 p_4 100$. Проверочные символы p_1, p_2, p_4 вычисляются следующим образом: p_i равно сумме по модулю 2 тех информационных символов, номера которых имеют единицу в двоичном представлении там же, где и номер p_i в i-м разряде справа, т.е. p_1 – в первом разряде, p_2 – во втором разряде, p_3 – в третьем разряде (см. таблицу 4). $p_1 = p_3 \oplus p_5 = 0$, $p_2 = p_3 = 1$, $p_4 = p_5 = 1$.

Таблица 4

i	*i* в двоичном представлении	Информац. позиции	Провероч. позиции	Расчет проверочных символов позиции	Сообщение после кодирования
1	001		0	$p_3 \oplus p_5$	0
2	010		1	p_3	1
3	011	1			1
4	100		1	p_5	1
5	101	1			1
6	110	0			0
7	111	0			0

Пример 6.

Пусть кодируемое слово $A = 1001110110$, m=10, k=4 ($k = [log_2(10)] + 1$)). Закодированное сообщение длины n=m+k=10+4=14 будет иметь вид $A = B =$ 01110010110110=$p_1p_2 1p_4 100p_8 110110$. Проверочные символы p_1, p_2, p_4, p_8 вычислены в таблице 5.

Таблица 5

i	i в двоичном представлении	Информац. позиции	Провероч. позиции	Расчет проверочных символов позиции	Сообщение после кодирования
1	0001		0	$p_3 \oplus p_7 \oplus p_9 \oplus p_{13}$	0
2	0010		1	$p_3 \oplus p_7 \oplus p_{10}$	1
3	0011	1			1
4	0100		1	$p_7 \oplus p_{12} \oplus p_{13}$	1
5	0101	0			0
6	0110	0			0
7	0111	1			1
8	1000		0	$p_9 \oplus p_{10} \oplus p_{12} \oplus p_{13}$	0
9	1001	1			1
10	1010	1			1
11	1011	0			0
12	1100	1			1
13	1101	1			1
14	1110	0			0

Пример 7. Построение кодов Хемминга (подробное описание алгоритма кодирования и декодирования)

Для заданного сообщения $X = 0110101$ построим код Хэмминга X'

№	Алгоритм	Конкретное соответствие задания заданному алгоритму
1	Определить число контрольных	m=7, вычисляем число контрольных разрядов $k = [log_2(7)] + 1$=4, n=7+4=11

<table>
<tr><td></td><td>разрядов и длину кода Хэмминга</td><td></td></tr>
<tr><td>2</td><td>Подготовить строку для сообщения X', отводя места с номерами. равными 2^i для проверочных символов, а остальные разряды – для информационных символов.</td><td>Разряды 1, 2, 4, 8 – проверочные; разряды 3, 5, 6, 7, 9, 10, 11 – информационные
–
<table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td></tr><tr><td>п</td><td>п</td><td>и</td><td>п</td><td>и</td><td>и</td><td>и</td><td>п</td><td>и</td><td>и</td><td>и</td></tr></table></td></tr>
<tr><td>3</td><td>Разместить знаки сообщения X в информационных разрядах</td><td><table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td></tr><tr><td></td><td></td><td>0</td><td></td><td>1</td><td>1</td><td>0</td><td></td><td>1</td><td>0</td><td>1</td></tr></table></td></tr>
<tr><td>4</td><td>Построить таблицу с двоичными номерами разрядов и внести в информационные разряды знаки сообщения X.</td><td><table><tr><td>1</td><td>0001</td><td></td></tr><tr><td>2</td><td>0010</td><td></td></tr><tr><td>3</td><td>0011</td><td>0</td></tr><tr><td>4</td><td>0100</td><td></td></tr><tr><td>5</td><td>0101</td><td>1</td></tr><tr><td>6</td><td>0110</td><td>1</td></tr><tr><td>7</td><td>0111</td><td>0</td></tr><tr><td>8</td><td>1000</td><td></td></tr><tr><td>9</td><td>1001</td><td>1</td></tr><tr><td>10</td><td>1010</td><td>0</td></tr><tr><td>11</td><td>1011</td><td>1</td></tr></table></td></tr>
<tr><td>5</td><td>Для каждого из проверочных разрядов с номерами</td><td><table><tr><td>№</td><td>Двоичное Представление номера строки</td><td>Инф. знаки</td><td>Пров. знаки</td><td>Номера строк, формирующих провер. знак</td></tr></table></td></tr>
</table>

<table>
<tr>
<td></td>
<td>2^i определить строки, формирующие проверочные знаки кода X':
1) информационный знак кода X' равен 1
2) в двоичном представлении номера строки i-й разряд равен 1.</td>
<td>
<table>
<tr><td>1</td><td>0001</td><td></td><td>1</td><td>5+9+11</td></tr>
<tr><td>2</td><td>0010</td><td></td><td>0</td><td>6+11</td></tr>
<tr><td>3</td><td>0011</td><td>0</td><td></td><td></td></tr>
<tr><td>4</td><td>0100</td><td></td><td>0</td><td>5+6</td></tr>
<tr><td>5</td><td>0101</td><td>1</td><td></td><td></td></tr>
<tr><td>6</td><td>0110</td><td>1</td><td></td><td></td></tr>
<tr><td>7</td><td>0111</td><td>0</td><td></td><td></td></tr>
<tr><td>8</td><td>1000</td><td></td><td>0</td><td>9+11</td></tr>
<tr><td>9</td><td>1001</td><td>1</td><td></td><td></td></tr>
<tr><td>10</td><td>1010</td><td>0</td><td></td><td></td></tr>
<tr><td>11</td><td>1011</td><td>1</td><td></td><td></td></tr>
</table>
</td>
</tr>
<tr>
<td>6</td>
<td>Вычислить значения проверочных разрядов</td>
<td>Значения проверочных разрядов получаются сложением по модулю 2 значений соответствующих информационных разрядов. Значение проверочного разряда определяется четностью числа слагаемых. Результат расчета в предпоследнем столбце таблицы в п.5</td>
</tr>
<tr>
<td>7</td>
<td>Объединяя проверочные и информационные разряды, получить искомое сообщение X'.</td>
<td>Построенное закодированное по Хэммингу сообщение $X' = 10001100101$.
Проверим правильность кодирования, вычислив значение $\mathrm{H}(X)$. Выпишем в столбец двоичные номера строк, в которых знак сообщения X' равен 1. Сумма по модулю 2 знаков номеров в каждом разряде должна быть равной 0.
<table>
<tr><th>№</th><th>Двоичное Представление номера строки</th><th>Знаки сообщ. X'</th></tr>
<tr><td>1</td><td>0001</td><td>1</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td><table>
<tr><td>5</td><td>0101</td><td>1</td></tr>
<tr><td>6</td><td>0110</td><td>1</td></tr>
<tr><td>9</td><td>1001</td><td>1</td></tr>
<tr><td>11</td><td>1011</td><td>1</td></tr>
<tr><td></td><td>0000</td><td></td></tr>
</table></td></tr>
<tr><td>8</td><td>Внести ошибку замещения в один из разрядов и провести декодирование.</td><td>Пусть при передаче сообщения $X' = 10001110101$ произошла ошибка в 7-м разряде, в результате получилось слово $X'' = 10001100101$. Вычислим значение $\mathrm{H}(X'')$. Выпишем в столбец двоичные номера строк, в которых знак сообщения X'' равен 1.
Суммируем по модулю 2 знаки номеров в каждом разряде.<table>
<tr><td>№</td><td>Двоичное Представление номера строки</td><td>Знаки сообщ. X''</td></tr>
<tr><td>1</td><td>0001</td><td>1</td></tr>
<tr><td>5</td><td>0101</td><td>1</td></tr>
<tr><td>6</td><td>0110</td><td>1</td></tr>
<tr><td>7</td><td>0111</td><td>1</td></tr>
<tr><td>9</td><td>1001</td><td>1</td></tr>
<tr><td>11</td><td>1011</td><td>1</td></tr>
<tr><td>сумма</td><td>0111</td><td></td></tr>
</table>Полученное двоичное число 0111 равно номеру разряда 7, в котором произошла ошибка. Заменяя в сообщении X'' значение 7-го разряда на противоположное, восстанавливаем X', вычеркивая из X' проверочные разряды, получаем искомое сообщение.</td></tr>
</table>

Докажем, что для любых двух различных слов $X,\ Y$, их закодированные по Хэммингу образы $X',\ Y'$ различаются не менее чем в трех разрядах, что и обеспечивает исправление единичной ошибки.

1) $d(X,Y) \geq 3$, т. е. $X,\ Y$ различаются в трех и более разрядах: при кодировании эти различия сохраняются; могут добавиться различия в проверочных разрядах, т. е. $d(X',Y') \geq 3$.

2) $d(X,Y) = 2$, т. е. $X,\ Y$ различаются в двух разрядах. Пусть в закодированных словах эти позиции $\alpha \neq \beta$. Числа α,β в двоичной записи различаются хотя бы в одном i-м разряде, который для α равен 0, а для β равен 1 или наоборот. При вычислении проверочного символа, соответствующего i-му разряду, т. е. позиции 2^i, только для одного из слов $X',\ Y'$ в сумме по модулю 2 участвует единица; по остальным позициям, отличным от α,β слова X,Y совпадают. Значит в 2^i-й проверочной позиции слова $X',\ Y'$ также различаются: в итоге $d(X',Y') \geq 3$.

3) $d(X,Y) = 1$, т. е. $X,\ Y$ различаются ровно в одном разряде. Номер этого информационного разряда в словах X,Y не равен целой степени числа 2, т. е. его двоичная запись содержит по крайней мере две единицы. Тогда в соответствующих двух проверочных разрядах слова $X,\ Y$ различаются, т. е. снова $d(X',Y') \geq 3$.

Если при передачи кодированного слова X произошла одна ошибка, то при декодировании полученного слова X' вектор $\mathrm{H}(X')$, прочитанный как двоичная запись, дает номер поврежденного разряда.

Код Хэмминга позволяет обнаружить и устранить одну ошибку. Можно построить и такой код, который обнаруживал бы двойные ошибки. Для этого к самокорректирующемуся коду, рассчитанному на исправление одиночных ошибок, нужно приписать ещё один контрольный разряд (разряд двойного контроля). Полное количество разрядов кода при этом будет $m+k+1$. Цифра в разряде двойного контроля устанавливается такой, чтобы общее количество

единиц во всех m +k +1 разрядах кода было четным. Этот разряд не включается в общую нумерацию и не входит ни в одну контрольную группу.

6.3. Линейно блочные коды

Линейные блочные коды – это класс кодов с контролем четности, которые можно описать парой чисел (n, m). В процессе кодирования блок из m символов сообщения (вектор сообщения) преобразуется в б*о*льший блок из n символов кодового слова (кодовый вектор), образованный с использованием элементов данного алфавита. Если алфавит состоит только из двух элементов (0 и 1), код является двоичным и включает двоичные разряды (биты).

Линейным блочным (n, m)-кодом называется множество N последовательностей длины n, называемых кодовыми словами, которое характеризуется тем, что сумма двух кодовых слов является кодовым словом, а произведение любого кодового слова на элемент поля (скаляр) также является кодовым словом.

Обычно $N = q^m$, где m – некоторое целое число. Если q=2, линейные коды называются групповыми, так как кодовые слова образуют математическую структуру, называемую группой. При формирование этого кода линейной операцией является суммирование по $mod\ 2$.

Линейный (n, m)-код называется высокоскоростным, если отношение m/n близко к 1, и низкоскоростным, если отношение m/n << 1– близко к нулю.

Благодаря линейности для запоминания или перечисления всех кодовых слов достаточно хранить в памяти кодера или декодера существенно меньшую их часть, а именно только те слова, которые образуют базис соответствующего линейного пространства. Это существенно упрощает реализацию устройств кодирования и декодирования и делает линейные коды весьма привлекательными с точки зрения практических приложений.

Способы задания линейных кодов:

1. Перечислением кодовых слов, т. е. составлением списка всех кодовых слов.

Пример 8. Все кодовые слова (5,2)-кода (a_i – информационные, а b_i – проверочные символы) представлены в таблице:

№	a_1	a_2	b_1	b_2	b_3
1	0	0	0	0	0
2	0	1	0	1	1
3	1	0	1	0	1
4	1	1	1	1	0

2. Системой проверочных уравнений, определяющих правила формирования проверочных символов по известным информационным символам: $b_j = \sum_{i=1}^{m} a_i h_{ij}$, где j – номер проверочного символа; i – номер информационного символа; h_{ij} – коэффициенты, принимающие значения 0 или 1 в соответствии с правилами формирования конкретных групповых кодов.

Пример 9. Для кода (5,2) проверочные уравнения имеют вид:

$b_1 = a_1, b_2 = a_2, b_3 = a_1 + a_2$.

3. Матричный способ, основанный на построении порождающей и проверочной матриц.

Определение. *Порождающей матрицей* кода называется $G(n,m)$-матрица, строки которой состоят из координат векторов $\bar{g}_1, \bar{g}_2, \cdots \bar{g}_m$ каком-нибудь заданном базисе. Название порождающей матрицы объясняется тем, что любое кодовое слово кода С является линейной комбинацией строк матрицы *G*. Из определения вытекает, что порождающая матрица кода определена неоднозначно.

Для исключения неоднозначности в записи $G(n,m)$ вводят понятие о канонической или систематической форме матрицы, которая имеет вид $G(n,m) = [I_m, Q_{m\times k}])$, где I_m – единичная матрица, содержащая информационные символы; $Q_{m\times k}$ – прямоугольная матрица, составленная из проверочных символов.

Пример 10. Порождающая матрица в систематическом виде для (5,2)-кода:

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} g^{(1)} \\ g^{(2)} \end{pmatrix}$$

Порождающая матрица $G(n, m)$ в систематическом виде может быть получена из любой другой порождающей матрицы посредством элементарных операций над строками (перестановкой двух произвольных строк, заменой произвольной строки на сумму ее самой и ряда других) и дальнейшей перестановкой столбцов.

Два кода называются эквивалентными, если их порождающие матрицы отличаются перестановкой координат, т. е. порождающие матрицы получаются одна за другой перестановкой столбцов и элементарных операций над строками.

Кодовые слова – это 2^m линейных комбинаций строк порождающей матрацы (кодовые слова получаются сложением строк матрицы в различных сочетаниях):

$\bar{s} = \sum_{i=1}^{m} a_i g^{(i)}$.

Для (5,2)-кода: $\bar{s} = a_1 g^{(1)} + a_2 g^{(2)}$.

При $a_1 = 0$ и $a_2 = 0$ $\bar{s} = (00000)$; при $a_1 = 0$ и $a_2 = 0$ $\bar{s} = (10101)$; при $a_1 = 0$ и $a_2 = 1$ $\bar{s} = (01011)$; при $a_1 = 1$ и $a_2 = 0 = 1$ $\bar{s} = (11110)$.

Кодовая схема Σ: $\begin{cases} 00 \to 00000 \\ 10 \to 10101 \\ 00 \to 01011 \\ 00 \to 11110 \end{cases}$

Расстояние линейного кода – это минимальный вес его ненулевых кодовых слов или равным образом минимальное расстояние между всеми парами различных кодовых слов.

Доказано, что расстояние между нулевым кодовым словом и одним из кодовых слов, входящих в порождающую матрицу (строки порождающей матрицы линейного блочного кода сами являются кодовыми словами, по определению), равно d_{min}. Но расстояние от любого кодового слова до

нулевого равно весу Хемминга этого слова (числу ненулевых символов этого слова). Тогда d_{min} равно минимальному весу Хемминга для всех строк порождающей матрицы кода.

В данном примере $d_{min} = \min_{s \neq 0} w(\bar{s})$=3, и код может исправить не более одной ошибки. Если $n = 5$ и мы хотим исправлять две ошибки, то кодовая схема будет содержать не более двух кодовых слов $\bar{s} = (11111)$ и $\bar{s} = (00000)$.

$\bar{s}$	$w(\bar{s})$
00000	0
10101	3
01011	3
11110	4

Таким образом, требование увеличения способности исправлять ошибки для кода фиксированной длины приводит к уменьшению максимального числа кодовых слов.

4. Проверочная матрица в систематическом виде имеет вид $H(n, m) = [Q^T_{m\times k}, I_k])$; где I_k – единичная матрица; $Q^T_{m\times k}$ – прямоугольная матрица в транспонированном виде матрицы $Q_{m\times k}$ из порождающей матрицы.

Пример 11. Проверочная матрица (5,2)-кода:

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

Пример 12. Зная проверочные уравнения, можно найти проверочную и порождающую матрицы.

Предположим, что для кода с $n = 7$ и $m = 4$ мы имеем следующие проверочные уравнения:

$b_1 = a_1 + a_2 + a_3$,

$b_2 = a_1 + a_2 + a_4$,

$b_3 = a_1 + a_3 + a_4$.

Тогда проверочная матрица H имеет вид:

$$\mathrm{H}(7,4) = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

порождающая матрица G имеет вид:

$$G(7,4) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Показателем качества передачи информации является k-разрядный двоичный вектор S, который называется *синдромом ошибки*.

Синдромом ошибки принятого слова $\bar{y}$ в коде Σ с проверочной матрицей H называется вектор: $\bar{s} = \bar{y} \cdot H^T$ длины $n - m$.

В соответствии с формулами, приведенными выше, если ошибок нет (t=0), то и синдром должен быть тождественно равен нулю. Таким образом если вес синдрома равен нулю, то ошибок нет. Если же произошла одиночная ошибка, то синдром равен вектор столбцу подматрицы Q.

Основные свойства линейных кодов:

1. Произведение любого кодового слова $v_i(x)$ на транспонированную проверочную матрицу дает нулевой вектор размерности $(n - m)$:
   $v_i(x)H^T(n,m) = (00\cdots)$.
2. Произведение некоторого кодового слова $v'_i(x)$, т. е. с ошибкой, на транспонированную проверочную матрицу называется синдромом и обозначается $S_i(x)$: $v'_i(x)H^T(n,m) = S_i(x)$.
3. Между порождающей и проверочной матрицами в систематическом виде существует однозначное соответствие, а именно:
   $G(n,m) \cdot H^T(n,m) = 0$
4. Кодовое расстояние $d_0(n,m)$ – кода равно минимальному числу линейно зависимых столбцов проверочной матрицы.
5. Произведение информационного слова на порождающую матрицу дает кодовое слово.

Если сообщение записывается в виде вектора $(a_1, a_2, \dots, a_m)$, то соответствующее сообщению кодовое слово вычисляется по формуле: $\bar{s} = \bar{a} \cdot G$.

6. k-разрядный исправляющий вектор (синдром) $\bar{c}$ определяется по правилу: $\bar{c} = \bar{b} \cdot H^T$.

7. Кодовое расстояние любого линейного $(n; m)$-кода удовлетворяет неравенству $d_0 \leq n - m + 1$ (граница Сингтона).

Линейный $(n; m)$-код, удовлетворяющий равенству $d_0 = n - m + 1$, называется кодом с максимальным расстоянием.

Пример 13. Линейный блочный (5,2)-код задан производящей матрицей в систематической (канонической) форме

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} g^{(1)} \\ g^{(2)} \end{pmatrix}$$

Пусть принят вектор $b = (00110)$ и известно, что возможны только одиночные ошибки.

Произвести декодирование следующими методами:

а) по минимуму расстояния;

б) вычислением синдрома.

Решение. Если производящая матрица записана в каноническом виде, это значит, что она состоит из двух блоков: диагональной матрицы размера $m \times m$, состоящей из единиц, и прямоугольной матрицы размера $m \times k$. В этом случае первые m символов в любом кодовом слове являются информационными.

Проверочная матрица H также может быть записана в каноническом виде и состоит из двух блоков. Первый блок есть прямоугольная матрица размера $k \times m$, полученная транспонированием второго блока матрицы G. В качестве второго блока матрицы H записывают диагональную матрицу размера $k \times k$, состоящую из единиц.

Для заданного кода получаем:

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

Убедимся, что полученная таким образом матрица H удовлетворяет свойству 3 $G(n,m) \cdot H^T(n,m) = 0$. Действительно,

$g_1 \cdot h_1 = (10110) \cdot (10100) = (00000);$

$g_1 \cdot h_2 = (10110) \cdot (11010) = (00000);$

……

$g_2 \cdot h_3 = (01011) \cdot (01001) = (00000)$

т.е. все 6 элементов матрицы $G(n,m) \cdot H^T(n,m)$ равны нулю.

Всего кодовая таблица содержит $2^m = 4$ вектора. Построим кодовую таблицу, воспользовавшись правилом образования кодовых слов по формуле:

$\bar{s} = h_1 g^{(1)} + h_2 g^{(2)};\ h_1, h_2 \in \{0,1\}$

$\bar{s_1} = 0 \cdot g^{(1)} + 0 \cdot g^{(2)} = (00000)$

$\bar{s_2} = 1 \cdot g^{(1)} + 0 \cdot g^{(2)} = (10110)$

$\bar{s_3} = 0 \cdot g^{(1)} + 1 \cdot g^{(2)} = (01011)$

$\bar{s_4} = 1 \cdot g^{(1)} + 1 \cdot g^{(2)} = \bar{s_2} + \bar{s_3} = (11101)$

Из кодовой таблицы определяем величину кодового расстояния $d_\Sigma = 3$. Следовательно, рассматриваемый код обнаруживает однократные и двукратные ошибки и исправляет однократные.

Декодируем принятый вектор $b = 00110$.

Метод декодирования по минимуму расстояния заключается в том, что, вычислив расстояния вектора b относительно всех векторов $\bar{s_i}$ кодовой таблицы, отождествляем принятый вектор с тем, расстояние до которого минимально. Расстояния d_i приведены в таблице. По величине $d_{i\,min}$=1 решаем, что передавался вектор $\bar{s_2} = (10110)$, следовательно, ошибка в первом символе кодового слова, а информационная последовательность имеет вид $x = 10$.

Таблица кодовых слов

$\bar{s_i}$	00000	10110	01011	11101

d_i	2	1	3	4

Метод декодирования с помощью вычисления синдрома включает следующие операции:

По формуле свойства 6 заранее устанавливаем однозначную связь между векторами однократных ошибок $\bar{e}$ и соответствующими им синдромами. Все возможные векторы $\bar{c} = \bar{e} \cdot H^T$ приведены в таблице.

Таблица синдромов:

e	**c**
00001	001
00010	010
00100	100
01000	011
10000	110

Вычисляем синдром для принятого слова b по формуле $\bar{c} = \bar{b} \cdot H^T$:

c_1=(00110)(10100)=1, c_2=(00110)(11010)=1, c_3=(00110)(01001)=0, т.е. вектор $\bar{c} = (110)$. Синдром не равен нулю, следовательно, есть ошибка.

Вектору $\bar{c} = (110)$ в таблице синдромов соответствует вектор ошибки в первом символе **e**=10000. Декодируем, суммируя принятый вектор с вектором ошибки $a = b \oplus e = (00110) \oplus (10000) = 10110$.

Итак, получили тот же результат: передавался вектор $\bar{s_2} = (10110)$, соответствующий информационной последовательности $x = 10$.

Пример 14. Построить код Хэмминга, имеющий параметры: $d_\Sigma = 3, m = 3$.

Решение. Построение кода начинаем с проверочной матрицы. В качестве семи столбцов проверочной матрицы H выбираем всевозможные 3-разрядные двоичные числа, исключая число нуль. Проверочная матрица имеет вид:

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

Чтобы определить места проверочных и информационных символов, матрицу H представляем в каноническом виде. Для этого достаточно в матрице H выбрать столбцы, содержащие по одной единице, и перенести их в правую часть. Тогда получим:

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

Из формулы свойства 1 $(a_1, a_2, \dots, a_n) \cdot H^T(n, m) = (00\cdots)$ получаем следующие соотношения для символов $a_1, a_2, \dots, a_7$ кодового слова:

$a_5 = a_2 + a_3 + a_4$

$a_6 = a_1 + a_3 + a_4$

$a_7 = a_1 + a_2 + a_4$

Пользуясь этими соотношениями, составляем кодовую таблицу:

$0000 \to 0000000, \quad 0001 \to 0001111, \dots, 1111 \to 1111111.$

Для кода Хэмминга, определяемого заданной проверочной матрицей H, декодируем принятый вектор 1011001.

Решение. Для принятого вектора $b = 1011001$ по формуле вычисляем синдром: c=$(1011001) \cdot H^T = (001)$

Синдром c≠0, т.е. имеет место ошибка. В случае одиночной ошибки (и только лишь) вычисленное значение синдрома всегда совпадает с одним из столбцов матрицы H, причем номер столбца указывает номер искаженного символа. Видим, что ошибка произошла в первом символе и передавался вектор a=(0011001).

Пример 15. Построение (7, 4)-кода Хемминга.

Порождающая матрица (7, 4)-кода Хемминга в систематическом виде состоит из единичной матрицы I_4, содержащей информационные биты и прямоугольной матрицы $Q_{4\times3}$, составленной из проверочных битов:

$$G(7,4) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Кодовые слова – это 2^4 линейных комбинаций строк матрицы G.

Таблица кодовых слов

Номер по пор.	Дес №	Информационные				Проверочные			Вес
		i_1	i_2	i_3	i_4	P_1	P_2	P_3	W_R
0	0	0	0	0	0	0	0	0	0
1	15	0	0	0	1	1	1	1	4
2	22	0	0	1	0	1	1	0	3
3	25	0	0	1	1	0	0	1	3
4	37	0	1	0	0	1	0	1	3
5	42	0	1	0	1	0	1	0	3
6	51	0	1	1	0	0	1	1	4
7	60	0	1	1	1	1	0	0	4
8	67	1	0	0	0	0	1	1	3
9	76	1	0	0	1	1	0	0	3
10	85	1	0	1	0	1	0	1	4
11	90	1	0	1	1	0	1	0	4
12	102	1	1	0	0	1	1	0	4
13	105	1	1	0	1	0	0	1	4
14	112	1	1	1	0	0	0	0	3
15	127	1	1	1	1	1	1	1	7

Описание таблицы: 16 строк – кодовые слова; 10 колонок: порядковый номер, десятичное представление кодового слова, 4 информационных символа, 3 проверочных символа, W – вес кодового слова равен числу ненулевых разрядов ($\neq 0$). Заливкой выделены 4 кодовых слова-строки – это базис векторного подпространства.

Построим проверочную матрицу в систематическом виде $H(7,4) = [Q_{4\times3}^T, I_3]$):

$$\mathrm{H}(7,4) = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Построим проверочную матрицу $H(7,4)$ в упорядоченном виде (здесь столбцы представляют упорядоченную запись десятичных чисел в двоичной форме):

$$\mathrm{H}(7,4) = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Проверочная матрица в упорядоченном виде представляет совокупность проверочных уравнений, в которых проверочные символы занимают позиции с номерами 2^i (i=0,1,2...). Для (7,4)-кода Хэмминга проверочными уравнениями будут $v_1=v_3+v_5+v_7$; $v_2=v_3+v_6+v_7$; $v_4=v_5+v_6+v_7$; где v_1, v_2, v_7; – проверочные символы. Элементы синдрома определяются из выражений $S_0=v_1+v_3+v_5+v_7$; $S_1=v_2+v_3+v_6+v_7$; $S_2=v_4+v_5+v_6+v_7$.

Проверочные уравнения используются для построения кодера, а синдромные – декодера кода Хэмминга.

Основной задачей декодера является проверка того, является ли полученное слово (7 разрядов) тем, которое было отправлено на передающей стороне, не содержит ли слово ошибок. Для решения этой задачи для каждого полученного слова декодером путем умножения его на проверочную матрицу H[n-m, n] вычисляется короткий вектор-синдром S (3 разряда).

В проверочной матрице кода декодер находит столбец, совпадающий со значением синдрома, и порядковый номер этого столбца принимает равным искаженному ошибкой разряду. После этого для двоичных кодов декодером выполняется изменение этого разряда – просто замена на противоположное значение, т. е. единицу заменяют нулем, а нуль – единицей.

Предположим, что получен вектор: $\bar{y} = (0110100)$.

Синдром вектора равент $\bar{s} = \bar{y} \cdot H^T$ =(111), что совпадает с четвертым столбцом матрицы H, следовательно вектор ошибок равен $\bar{e} = (0001000)$.

Тогда $\bar{y} + \bar{e} = (0110100) + (0001000) = (0111100)$ – получили переданный вектор.

Пример 16. При кодировании текста с использованием стандартной клавиатуры ASCII-кодов каждому символу (букве алфавита) соответствует в этой кодировке октет восемь разрядов. Для (7,4)-кода Хемминга, в словах которого только 4 информационных символа, при кодировании символа клавиатуры на букву требуется два кодовых слова, т.е. восемь разрядов разбивается на два информационных слова естественного языка (ЕЯ) вида (a_1, a_2, a_3, a_4).

Например, необходимо передать слово «цифра» в ЕЯ. Входим в таблицу ASCII-кодов, буквам соответствуют: ц –11110110, и –11101000, ф – 11110100, р – 11110000, а – 11100000 октеты. Или иначе в ASCII-кодах слово «цифра» = 1111 0110 1110 1000 1111 0100 1111 0000 1110 0000 с разбивкой на тетрады (по 4 разряда). Таким образом, кодирование слова «цифра» ЕЯ требует 10 кодовых слов (7, 4)-кода Хемминга. Тетрады представляют информационные разряды слов сообщения. Эти информационные слова (тетрады) преобразуются в слова кода (по 7 разрядов) перед отправкой в канал сети связи. Выполняется это путем векторно-матричного умножения: информационного слова на порождающую матрицу.

Например, чтобы закодировать букву ц $= (1111\ 0110)$ нужно каждую половинку буквы-сообщения умножить на порождающую матрицу (7,4)-кода Хемминга.

$$(1111) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} = (1111111)$$

$$(0110) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} = (0110011)$$

7. Циклические коды

7.1 Основные понятия

Циклическим кодом называется линейный блоковый (n, m)-код, который характеризуется свойством цикличности, т. е. сдвиг влево на один шаг любого разрешенного кодового слова дает также разрешенное кодовое слово, принадлежащее этому же коду. Множество кодовых слов представляется совокупностью многочленов степени $(n-1)$ и менее, как следует из определения, в циклическом коде кодовые слова представляются в виде многочленов $v(x) = v_{n-1}x^{n-1} + \cdots + v_0x^0$, где n – длина кода; v_i– коэффициенты. Если коэффициенты принимают значения *0, 1*, то код называется двоичным.

Пример 17. Если кодовое слово циклического кода x =*1011101001*, то соответствующий ему многочлен $v(x) = x^9 + x^7 + x^6 + x^5 + x^3 + 1$.

Общее свойство кодовых слов циклического кода – это их делимость без остатка на некоторый многочлен $g(x)$, называемый порождающим. Результатом деления двучлена $x^n + 1$ на многочлен $g(x)$ является проверочный многочлен $h(x)$. При декодировании циклических кодов используются многочлен ошибок $e(x)$ и синдромный многочлен $S(x)$.

Многочлен ошибок степени не более $(n-1)$ определяется из выражения $e(x) = v'(x) + v(x)$; где $v'(x)$, $v(x)$ – многочлены, отображающие соответственно принятое (с ошибкой) и переданное кодовые слова. Ненулевые коэффициенты в $e(x)$ занимают позиции, которые соответствуют ошибкам.

Пример 18. $v(x) = x^6 + x^4 + x^3 + x + 1$ (1011011);
$v'(x) = x^6 + x^4 + x^2 + x + 1$ (1010111); $e(x) = x^3 + x^2$(0001100).

Синдромный многочлен, используемый при декодировании циклического кода, определяется как остаток от деления принятого кодового слова на порождающий многочлен, т.е. $S(x) = R_{g(x)}[v'(x)]$=$R_{g(x)}[v'(x) + e(x)] = R_{g(x)}[e(x)]$. Следовательно, синдромный многочлен зависит непосредственно

от многочлена ошибок $e(x)$. Это свойство используется при построении таблицы синдромов, применяемой в процессе декодирования.

Таблица многочленов ошибок

многочлен $e(x)$	синдром $S(x)$	многочлен $e(x)$	синдром $S(x)$
1	$R_{g(x)}[1]$	$x+1$	$R_{g(x)}[x+1]$
x	$R_{g(x)}[x]$	x^2+1	$R_{g(x)}[x^2+1]$
x^2	$R_{g(x)}[x^2]$	…	…

В процессе декодирования по принятому кодовому слову вычисляется синдром, затем в таблице находится соответствующий многочлен $e(x)$, суммирование которого с принятым кодовым словом дает исправленное кодовое слово, т. е. $v_i(x) = v_i'(x) + e_i(x)$.

Перечисленные многочлены $v(x), v'(x), g(x), h(x), e(x), S(x)$ можно складывать, умножать, делить, используя правила алгебры, но с приведением результата по $mod\,2$, а затем по $mod\,x^n+1$, если степень результата превышает степень $(n-1)$.

Пример 19. Допустим, что длина кода *n=7*, то результат приводим по $mod(x^7+1)$: $x^8+x^5+x^4+x^2+x+1\ mod(x^7+1) = x^5+x^4+x^2+1$.

При построении и декодировании циклических кодов в результате деления многочленов обычно необходимо иметь не частное, а остаток от деления. Поэтому рекомендуется более простой способ деления, используя не многочлены, а только его коэффициенты.

7.2 Матричное задание кодов

Циклический код может быть задан порождающей и проверочной матрицами. Для их построения достаточно знать порождающий $g(x)$ и проверочный $h(x)$ многочлены. Для несистематического циклического кода матрицы строятся циклическим сдвигом порождающего и проверочного многочленов, т.е. путем их умножения на *x*: $G(n,m) = \|g(x); xg(x); \dots; x^{m-1}g(x)\|^T$, $H(n,m) = \left\|h(x); xh(x); \dots; x^{k-1}h(x)\right\|^T$. При

построении матрицы $H(n,m)$ старший коэффициент многочлена $h(x)$ располагается справа.

Пример 20. Для циклического (7,4)-кода с порождающим многочленом $g(x) = x^3 + x + 1$ матрицы $G(n,m)$ и $H(n,m)$ имеют вид:

	0	*0*	*0*	*1*	*0*	*1*	*1*
$G(7,4)$	*0*	*0*	*1*	*0*	*1*	*1*	*0*
	0	*1*	*0*	*1*	*1*	*0*	*0*
	1	*0*	*1*	*1*	*0*	*0*	*0*

и

	1	*1*	*1*	*0*	*1*	*0*	*0*
$H(7,4)$	*0*	*1*	*1*	*1*	*0*	*1*	*0*
	0	*0*	*1*	*1*	*1*	*0*	*1*

где $h(x) = (x^7 + 1)/g(x) = x^4 + x^2 + x + 1$.

Для систематического циклического кода матрица $G(n,m)$ определяется из выражения $G(n,m) = [I_m, R_{m\times k}]$ где I_m – единичная матрица; $R_{m\times k}$ – прямоугольная матрица. Строки матрицы $R_{m\times k}$ определяются из выражений $r_i(x) = R_{g(x)}[a_i(x)x^k] = R_{g(x)}\left[x^{n-i}\right]$, где $a_i(x)$ – значение i-той строки матрицы I_m; i – номер строки матрицы $R_{m\times k}$.

Пример 21. Матрица $G(n,m)$ для (7,4)-кода на основе порождающего многочлена $g(x) = x^3 + x + 1$, строится в следующей последовательности $G(7,4) = [I_4, R_{4\times 3}]|$. Определяется $R(4,3)$, используя $r_i(x) = R_{g(x)}[a_i(x)x^k]$:

$r_1(x) = R_{g(x)}[x^3 \cdot x^3] = x^6 mod(x^3 + x + 1) = x^2 + 1\ (101)$;

$r_2(x) = x^2 + x + 1\ (111)$;

$r_3(x) = x^2 + x + (110)$;

$r_4(x) = x + 1\ (011)$.

В результате получаем:

	1	*0*	*0*	*0*		*1*	*0*	*1*
$G(7,4)$	*0*	*1*	*0*	*0*		*1*	*1*	*1*
	0	*0*	*1*	*0*		*1*	*1*	*0*
	0	*0*	*0*	*1*		*0*	*1*	*1*

Используя выражение $r_i(x) = R_{g(x)}\left[x^{n-i}\right]$, получим тот же результат. Строки матрицы $G(n,m)$ можно определить непосредственно из выражения $v_i(x) = a_i(x)x^k + r_i(x)$, где $r_i(x) = R_{g(x)}[a_i(x)x^k]$. Проверочная матрица в систематическом виде строится на основе матрицы $G(n,m)$, а именно: $H(n,m) = \left[{R_{m\times k}}^T; I_k\right]$; где I_k – единичная матрица; ${R_{m\times k}}^T$ – матрица из $G(n,m)$ в транспонированном виде.

Пример 22. Для (7,4)-кода матрица $H(n,m)$ будет иметь вид:

	1	*1*	*1*	*0*		*1*	*0*	*0*
$H(7,4)$	*0*	*1*	*1*	*1*		*0*	*1*	*0*
	1	*1*	*0*	*1*		*0*	*0*	*1*

Одна из основных задач, стоящих перед разработчиками устройств защиты от ошибок при передаче дискретных сообщений по каналам связи является выбор порождающего многочлена $g(x)$ для построения циклического кода, обеспечивающего требуемое минимальное кодовое расстояние для гарантийного обнаружения и исправления *t*-кратных ошибок. Существуют специальные таблицы по выбору $g(x)$ в зависимости от предъявляемых требований к корректирующим возможностям кода. Однако у каждого циклического кода имеются свои особенности формирования $g(x)$. Поэтому при изучении конкретных циклических кодов будут рассматриваться соответствующие способы построения $g(x)$.

Задания

1. Подсчитать расстояние Хэмминга для слов 001011 и 011001; 01011010 и 00001000.

2. Подсчитать минимальное кодовое расстояние для следующего кода 01101000; 00101111; 01010101; 11111111. Сколько ошибок может обнаружить код и сколько исправить?

3. Пусть дана матрица

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

Напишите коды сообщений: 001; 111; 101; 010.

Напишите систему линейных уравнений для кодов.

4. Для заданного сообщения нужно построить код Хэмминга. Внесите ошибку в i-й разряд и, произведя декодирование, подтвердите место ошибки:

1) A=1010011 (i=5)

2) B=11001010 (i=6)

5. Закодировать двухзначное десятичное число методом Хэмминга, ввести одиночную ошибку и исправить ее.

6. Найти ошибку и восстановить сообщение $a_4a_3a_2a_1$, закодированное по Хэммингу: 0110011.

7. Определить положение одиночной ошибки в искаженном слове 11110111000101111 0001 кода Хэмминга длины 7.

8. Для кодирующих матриц $G_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$, $G_2 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ построить соответственно (5,2)-код и (4,3)-код.

   a) Найти основные характеристики полученных кодов: скорость кода; избыточность; минимальное расстояние между словами кода; максимальную кратность ошибок, до которой включительно они все исправляются или обнаруживаются.

   b) Построить таблицы декодирования.

с) Во что будут декодированы слова: 10001, 01110, 10101, 1001, 0110, 1101?

9. Составить кодовую таблицу, определить кодовое расстояние и вычислить минимальное значение избыточности 3-разрядного двоичного кода, удовлетворяющего требованиям:

а) код содержит максимальное количество кодовых слов;

б) код обнаруживает все однократные ошибки;

в) код исправляет все однократные ошибки.

10. Найти все кодовые векторы для линейного блочного кода, заданного порождающей матрицей $G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$

11. Для кода, заданного порождающей матрицей $G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$ построить проверочную матрицу H.

12. Какими корректирующими свойствами обладает код заданный кодовой схемой

$$\Sigma: \begin{cases} a_1 \to 000000 \\ a_2 \to 10101 \\ a_3 \to 01011 \\ a_4 \to 11110 \end{cases}$$

Является ли этот код линейным блочным?

13. Систематический код задан производящей матрицей

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

Закодировать информационную последовательность, заданную в виде десятичного числа x = 7.

14. Систематический (n, m)-код задан производящей матрицей

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

Декодировать принятый вектор 0111011 методами минимального расстояния и вычисления синдрома.

15. Построить код Хэмминга, заданный параметрами $d_{\Sigma} = 3$, $m = 4$, Декодировать принимаемый вектор b=110101010101010.

16. Кодирующее устройство отображает последовательность информационных символов простого равномерного кода 00, 01, 10, 11 в последовательность кодовых векторов 00000, 01101, 10111, 11010, принадлежащих корректирующему коду:

а) показать, что полученный код является систематическим. Выразить каждый символ кодового слова в виде линейной комбинации информационных символов;

б) найти для этого кода производящую и проверочную матрицы;

в) привести таблицу декодирования (вычисления синдрома) для случая q = 1.

Список литературы

1. Березкин, Е. Ф. Основы теории информации и кодирования : учебное пособие / Е. Ф. Березкин. – 3-е изд., стер. – Санкт-Петербург : Лань, 2022. – 320 с. – ISBN 978-5-8114-4119-8.

2. Кудряшов, Б.Д. Основы теории кодирования : учебное пособие / Б. Д. Кудряшов – СПб.: БХВ-Петербург, 2016. – 400 с. – ISBN 978-5-9775-3527-4.

3. Ляшева, С. А. Теория информации и кодирования : учебно-методическое пособие / С. А. Ляшева. – Казань : КНИТУ-КАИ, 2020. – 120 с. – ISBN 978-5-7579-2493-1.

4. Осокин, А. Н. Теория информации : учебное пособие для вузов / А. Н. Осокин, А. Н. Мальчуков. – Москва : Издательство Юрайт, 2022. – 205 с. – (Высшее образование). – ISBN 978-5-9916-7064-7.

Прокопенко Наталья Юрьевна

ТЕОРИЯ КОДИРОВАНИЯ

Учебное пособие

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Нижегородский государственный архитектурно-строительный университет»
603950, Нижний Новгород, ул. Ильинская, 65.
http://www. nngasu.ru, srec@nngasu.ru