Лекция 13. Элементы теории кодирования. Понятие кода. Разделимость и префиксность кодов. Неравенство Крафта-Макмиллана для разделимых кодов. Алфавитное кодирование. Оптимальное кодирование. Код Фано. Коды с обнаружением и исправлением ошибок. Расстояние Хеминга. Метрическое пространство кодов. Кодовое расстояние. Виды ошибок. Код Хемминга.

Алфавитное кодирование

1. Кодом называется система условных знаков (символов) для передачи, обработки и хранения (запоминания) различной информации. Предметом исследования теории кодирования являются отображения конечных или счетных множеств объектов произвольной природы в множества последовательностей из цифр 0, 1,..., r–1, где r – некоторое целое положительное число (в частности, r = 2). Такие отображения называются **кодированиями**.

Большинство задач теории кодирования укладывается в следующую схему. Для заданного множества объектов рассматривается класс кодирований, обладающих определенными свойствами. Требуется построить кодирование из рассматриваемого класса, оптимальное в некотором заранее заданном смысле. Обычно критерий оптимальности кодирования так или иначе связан с минимизацией длин кодов, в то время как требуемые свойства кодирований могут быть весьма разнообразными. Среди таких свойств: существование однозначного обратного отображения (**декодирования**), возможность исправления при декодировании ошибок различного типа, возможность простой реализации (простота алгоритма) кодирования и декодирования и т. п.

Пусть A – произвольный алфавит. Элементы алфавита A будем называть *буквами*, а конечные последовательности (кортежи), составленные из букв, – *словами в A*. *Длина* (число букв) слова a обозначается через l(a), а слово длины 0 (**пустое слово**) обозначается символом L. *Соединение* слов a_1 и a_2 обозначим через a_1a_2, а соединение n одинаковых слов a – через a^n ($a^0 = L$).

Пусть U – произвольное множество слов в A. Через U^n (n = 0,1,...) обозначим множество всех слов в A^n, представимых в виде соединения n слов из U. В частности, через A^n (n = 0,1,...) обозначим множество всех слов длины n в алфавите A, а через A* – множество всех слов произвольной длины в A.

Множество $U^* = \bigcup_{n=0}^{\infty} U^n$ – не всевозможные слова в алфавите A; это зависит от выбранного множества U. Так, если U состоит из двух слов 00 и 11, то любое их соединение, во-первых, имеет четную длину и, во-вторых, нули и единицы расположены парами; например, слово 0110 не принадлежит множеству U^2.

Слово a_1 называют **началом** слова a, если существует слово a_2 такое, что $a = a_1a_2$; при этом слово a_1 называют **собственным началом** слова a, если $a_2 \neq L$. Слово a_2 называют **окончанием** слова a, если существует слово a_1 такое, что $a = a_1a_2$; при этом слово a2 называют **собственным окончанием** слова a, если a1 № L. В частности, пустое слово является началом и окончанием любого слова a, причем собственным,

если a ≠ L. Множество начал слов из множества U обозначим через $\vec{U}$, а множество окончаний – через $\overleftarrow{U}$.

2. Рассмотрим алфавит $B_{(r)} = \{0, 1,..., r-1\}$, где r i 2, и произвольное множество G. В частности, G может быть конечным алфавитом, множеством натуральных чисел, схем или формул определенного типа, множеством слов в некотором алфавите и т.п. Произвольное отображение множества G в множество слов в алфавите B называется **r-ичным кодированием множества G**. Мы будем рассматривать только случай r = 2, т.е. **двоичные** кодирования, и в связи с этим под log X будем понимать $\log_2 X$.

Приведем примеры кодирований.

1) Кодирование E – двоичная запись натуральных чисел. Числу n = 0 ставится в соответствие слово e(0) = 0, а числу n > 0 двоичное слово

$$e(n) = b_1^{(n)} b_2^{(n)} ... b_{l(n)}^{(n)}$$

наименьшей длины, удовлетворяющее условию

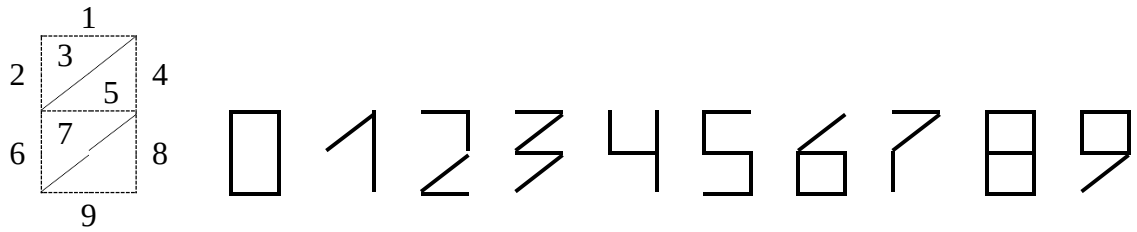
$$\sum_{j=1}^{l(n)} b_j^{(n)} \cdot 2^{l(n)-j} = n.$$

Очевидно, что запись числа n начинается с цифры 1, т.е. $b_1^{(n)} = 1$, а число знаков $l(n)$ должно удовлетворять двойному неравенству $2^{l(n)-1} \le n < 2^{l(n)}$ и, следовательно, $l(n) = [\log n] + 1 =]\log (n+1)[$ (здесь [X] обозначает целую часть числа X, а]X[– наименьшее целое, не меньшееX). Кодирование E является взаимно однозначным: при $n_1 \neq n_2$ слова $e(n_1)$ и $e(n_2)$ различны.

2) Кодирование E_k первых 2^k натуральных чисел. Каждому числу n ($0 \le n < 2^k$) ставится в соответствие слово $ek(n) = 0^{k-l(n)}e(n)$. Слово ek(n) называется **двоичной записью числа n с помощью k цифр**. При кодировании E_k первые 2^k натуральных чисел отображаются на множество двоичных слов длины k. В таблице 3.1 приведены слова, соответствующие числам от 0 до 15 при кодированиях E и E4.

Упражнение. Составьте коды e(25), $e_5(25)$, $e_6(25)$, e(30), $e_6(30)$, $e_7(30)$.

3) Отображение, применяемое при написании цифр почтового индекса (рис. 3.1). Каждая десятичная цифра, выделенная отправителем из девяти отрезков шаблона, кодируется для автоматического распознавания словом длины 9 в алфавите B: символами 1 отмечаются номера использованных линий (например, цифре 2 соответствует слово 100100101; цифре 5 – слово 110010011).



Упражнение. Постройте код для 6-значного почтового индекса своего почтового отделения по месту жительства, используя описанные в п.2 двоичные 9-значные коды десятичных цифр.

Отображение конечных множеств на некоторые подмножества двоичных слов часто является удобным средством для подсчета или оценки числа элементов этих множеств. Рассмотрим, например, множество $\mathcal{F}_{n,m}$ упорядоченных разбиений числа

n на m целых неотрицательных слагаемых. Каждому разбиению $n = n_1 + n_2 + ... + n_m$ поставим в соответствие слово $0^{n_1}10^{n_2}1...10^{n_m}$. Оно имеет длину (n + m – 1), т. к. содержит n нулей и (m – 1) единиц, разделяющих серии нулей. Так, разбиение числа 11 на 4 слагаемых 11 = 5 + 0 + 2 + 4 кодируется словом 00000110010000, разбиение 11 = 2 + 7 + 2 + 0 – словом 001000000010011, а разбиение 11 = 1 + 1 + 6 + 3 – словом 01010000001000. Такое кодирование является взаимно однозначным отображением множества $\mathcal{F}_{n,m}$ на множество двоичных слов длины (n + m – 1), имеющих ровно (m – 1) единиц. Следовательно, число элементов множества $\mathcal{F}_{n,m}$ равно C_{n+m-1}^{m-1}.

3.

n	e(n)	$e_4(n)$	n	e(n)	$e_4(n)$	n	e(n)	$e_4(n)$
0	0	0000	6	110	0110	12	1100	1100
1	1	0001	7	111	0111	13	1101	1101
2	10	0010	8	1000	1000	14	1110	1110
3	11	0011	9	1001	1001	15	1111	1111
4	100	0100	10	1010	1010			
5	101	0101	11	1011	1011			

Особую роль в теории кодирования играет кодирование множества G, состоящего из всех (или выделенной части) слов в некотором алфавите A. Элементы кодируемого множества (т. е. выделенные слова в A) при этом называют **сообщениями**, имея в виду возможную передачу кода по каналу связи. В общем случае не накладывается никаких условий на процесс вычисления (сам алгоритм, его эффективность) для сообщения соответствующего ему двоичного слова. Однако для многих вопросов достаточно ограничиться рассмотрением более узких классов кодирований слов, так называемых побуквенных кодирований.

Пусть $A = \{a_i\}$ – конечный алфавит, буквы которого занумерованы натуральными числами. В этом случае кодирование букв алфавита A можно задать двоичными словами $V = \{v_i\}$, где v_i есть образ буквы a_i. Слова $V = \{v_i\}$ будем называть **кодами** (алфавита A). Кодирование слов в алфавите A, при котором каждому слову (сообщению) $a_{i_1}a_{i_2}...a_{i_k}$ ставится в соответствие слово $v_{i_1}v_{i_2}...v_{i_k}$, будем называть **побуквенным кодированием** и обозначать через $K_{v_i}^{a_i}$ (или K_V^A).

Пример. Пусть A = {a,b,c,d}, двоичное кодирование его букв –

$$a \to 01,\ b \to 100,\ c \to 101,\ d \to 0.$$

Слово 0100 можно декодировать как db или add; слово 0010100 – как ddcdd, daadd или dadb.

Заметим, что хорошо известный телеграфный код Морзе – кодирование букв алфавита (латинского или русского), цифр и знаков препинания словами в двухбуквенном алфавите “точка”, “тире” – не является двоичным, поскольку коды Морзе отделены дополнительным символом – разделителем: при передаче по каналам связи – временным (паузой), а на письме – пробелом.

При переходе от взаимно однозначного кодирования букв алфавита к побуквенному кодированию слов свойство взаимной однозначности может не сохраниться. Например, при побуквенном кодировании $K_e^n(n)$ натуральных чисел кортежам из натуральных чисел (5,6, 1), (11, 2, 1) и (5, 13) соответствует одно и то же слово

1011101 (101*110*1, 1011*10*1, 101*1101). Таким образом, указанное побуквенное кодирование натуральных чисел не является взаимно однозначным.

Если коды всех букв алфавита имеют одинаковую длину, то код называется равномерным (пример – код En в отличие от кода E).

4. Можно считать, что в коде (множестве слов) $V = \{v_i\}$, содержатся только различные слова. Назовем код $V = \{v_i\}$ **разделимым**, если из каждого равенства в алфавите $B = \{0, 1\}$ вида $v_{i_1}v_{i_2}...v_{i_k} = v_{j_1}v_{j_2}...v_{j_l}$ следует, что $k = l$ и $i_t = j_t$, $t = 1, 2,...,k$.

Очевидно, что побуквенное кодирование является взаимно однозначным тогда и только тогда, когда оно задается с помощью разделимого кода. Из определения следует также, что все слова разделимого кода различны и непусты.

Код $V = \{v_i\}$ будем называть **префиксным**, если никакое слово v_k не является началом никакого слова v_j, $j \neq k$. Ясно, что префиксный код является разделимым, т.е. *префиксность является достаточным условием разделимости*. Заметим, что если каждое слово префиксного кода заменить наименьшим его началом, которое не является началом других кодовых слов, то полученный код также будет префиксным. Такую операцию будем называть **усечением** префиксного кода.

Простейшим примером префиксного кода является равномерный код. Он не требует разделителей, поскольку код каждой передаваемой буквы заканчивается, а код следующей буквы начинается через определенное одинаковое число кодирующих символов.

Пример. Для 7-значных номеров московских телефонов префиксность (и, значит, разделимость) обеспечивается равномерностью. Для номеров же, начинающихся с 0: 01, 02, 03 и др., требование разделимости не допускает других номеров с тем же началом. Тем самым, наличие, например, кода 01 исключает $10^5 = 100000$ возможных 7-значных номеров.

Префиксности кода можно добиться и другим способом, использованным в упомянутом коде Морзе: включением в качестве окончания любого кодового слова специального символа – разделителя. Аналогично, ввод в компьютер имени файла, имени данного, многозначного числа и т.п. должен завершаться разделителем, в частности, нажатием клавиши <ENTER>.

Префиксность не является необходимым условием разделимости кода. Так, код <a: 0, b: 01> двухбуквенного алфавита {a, b} – не префиксный, но является разделимым. В самом деле, если в некотором сообщении после символа 0 следует 0, то первый 0 кодирует букву a, так как 00 не является началом никакого кодового слова; если же после 0 следует 1, то пара символов 01 – код буквы b: 0 не может кодировать a, так как последующая единица не является началом никакого кодового слова.

Для произвольного кода V, состоящего из различных слов, рассмотрим ориентированный граф с множеством вершин $\vec{V}$, в котором вершина $\beta \in \vec{V}$ соединена с вершиной $\beta' \in \vec{V}$ дугой (β,β') тогда и только тогда, когда $\beta' = \beta 0$ или $\beta' = \beta 1$. Этот граф не содержит циклов и называется кодовым деревом кода V. Для префиксных кодов (и только для них) множество кодовых слов совпадает с множеством концевых вершин кодового дерева (т.е. вершин, из которых не исходит ребер). На рис. 6.2 изображено кодовое дерево кода, приведенного в следующей таблице ($l_i = \lambda(v_i)$).

i	l_i	v_i
0	3	000
1	3	001
2	3	010
3	4	0110
4	4	0111
5	4	1000
6	4	1001

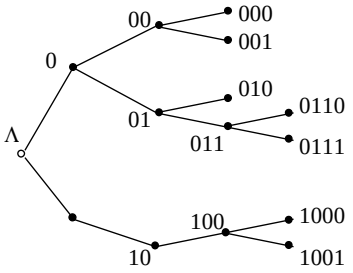


Рис. 6.2

Упражнение. Постройте кодовое дерево для кодов:

а)	б)	в)
a = 000	**a** = 0000	**a** = 00
b = 010	**b** = 0001	**b** = 0101
c = 011	**c** = 0011	**c** = 011
d = 10	**d** = 01	**d** = 1100
e = 1100	**e** = 100	**e** = 1101
f = 1110	**f** = 101	**f** = 111
g = 1111	**g** = 111	

В дальнейшем для конечного кода $V = \{v_0, v_1,..., v_{m-1}\}$ будем считать, что $m \geq 2$, и пользоваться обозначением $\lambda_{max} = \max\limits_{0<i<m-1} \lambda(v_i)$.

Теорема 1. Пусть $l_0, l_1,...,l_{m-1}$ – произвольный набор натуральных чисел ($m \geq 2$). Для того чтобы существовал разделимый код $V = \{v_0, v_1,..., v_{m-1}\}$ с длинами $\lambda(v_i) = l_i$, $i = 0, 1,..., m-1$, необходимо и достаточно выполнение **неравенства Крафта-Макмиллана для разделимых кодов**:

$$\sum_{i=0}^{m-1} \frac{1}{2^{l_i}} \leq 1.$$

Для кода, заданного в таблице выполнено: $3 \cdot \frac{1}{2^3} + 4 \cdot \frac{1}{2^4} = \frac{3}{8} + \frac{4}{16} = \frac{5}{8} < 1$.

Следствие. Для любого разделимого кода $V = \{v_0, v_1,..., v_{m-1}\}$ существует префиксный код с таким же набором длин кодовых слов, что и у кода V.

Оптимальное кодирование

1. Здесь мы разберем построение 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 приходится меньшее число двоичных цифр.

Рассматривается источник, который случайным образом (т.е. в случайном порядке) последовательно порождает буквы алфавита $A = \{a_0, a_1,..., a_{m-1}\}$. Предполагается, что последовательные появления букв алфавита A статистически независимы и подчинены распределению вероятностей

$$P = \{p_0, p_1,..., p_{m-1}\} \ (p_i \geq 0, \sum_{i=0}^{m-1} p_i = 1).$$

Каждый код $V = \{v_0, v_1,..., v_{m-1}\}$ в алфавите $B = \{0, 1)$ характеризуется средним числом

$$L_V(P) = \sum_{i=0}^{m-1} p_i \cdot \lambda(v_i)$$

двоичных цифр, приходящихся на одну букву алфавита A при побуквенном кодировании $K_{v_0}^{a_0}, K_{v_1}^{a_1}, ..., K_{v_{m-1}}^{a_{m-1}}$. Величину $L_V(P)$ называют **стоимостью** кода V при распределении P. Пусть $L(P) = \inf L_V(P)$, где нижняя грань берется по (счетному) множеству префиксных кодов V, состоящих из m слов. Префиксный код $V = \{v_0, v_1, ..., v_{m-1}\}$будем называть **оптимальным** при распределении $P = \{p_0, p_1, ..., p_{m-1}\}$, если $L_V(P) = L(P)$. Основная задача, которая здесь возникает, состоит в отыскании методов построения оптимальных или близких к ним по стоимости кодов и в оценке величины $L(P)$.

Известны два метода построения кодов, близких к оптимальному, принадлежащие К. Шеннону и Р. Фано. Метод Фано, отличающийся чрезвычайной простотой конструкции, заключается в следующем. Упорядоченный (в порядке убывания 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 приписывается символ 0, а буквам из второй части – символ 1. Далее точно так же поступают с каждой из полученных частей, если она содержит по крайней мере 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. Легко видеть, что полученный код является префиксным. Пример кода, построенного по методу Фано, приводится в следующей таблице.

сообщение	вероятность									коды	вес
a	0,30	0,48	0,30			0	0			00	2
b	0,18		0,18				1			01	2
c	0,14	0,52	0,28	0,14		1	0	0		100	3
d	0,14			0,14				1		101	3
e	0,11			0,11			1	0		110	3
f	0,06		0,24	0,13	0,06			1	0	1110	4
g	0,05				0,07	0,05		1	1	11110	5
h	0,02					0,02				11111	5

Стоимость кода:

$$2 \cdot (0{,}30 + 0{,}18) + 3 \cdot (0{,}14 + 0{,}14 + 0{,}11) + 4 \cdot 0{,}06 + 5 \cdot (0{,}05 + 0{,}02) = 2{,}72.$$

Следует заметить, что деление пополам списка букв может при определенном соотношении частот осуществляться неоднозначно.

Упражнения. Постройте самостоятельно код Фано для списка сообщений с заданным распределением частот. Определите стоимость кода.

	S	T	U	V	W	X	Y	Z
1.	0,15	0,1	0,15	0,15	0,1	0,1	0,15	0,1
2.	0,15	0,02	0,25	0,15	0,08	0,15	0,2	-
3.	0,02	0,25	0,04	0,01	0,4	0,1	0,03	0,15
4.	0,15	0,2	0,08	0,12	0,15	0,1	0,2	-
5.	0,07	0,04	0,3	0,1	0,07	0,12	0,3	-

2. В распространенном в телеграфии коде Морзе каждой букве или цифре, а также знакам препинания и некоторым другим символам сопоставляется некоторая последовательность кратковременных посылок тока (“точек”) и в 3 раза более длинных посылок тока (“тире”: 3 кратких посылки с малыми промежутками), разделяемых кратковременными паузами той же длительности, что и “точки”. Пробел между буквами изображается разделителем-паузой (выключением тока на 3 единицы времени), длительность которой равна длительности “тире”, а пробел между словами – двойной паузой.

Код Морзе создан с учетом частоты употребления разных букв латинского алфавита в европейских языках. В таблице ниже приведены коды латинских букв и их частоты в англоязычных текстах. Более употребительные буквы, в основном, кодируются более короткими словами. Если кодировать английский текст по буквам, не учитывая указанных частот (и частоты пробела), то потребуется в среднем $\log_2 26 \approx 4.70$ двоичных знаков на одну букву (или $4.75 \approx \log_2 27$ – с учетом пробела). При использовании кода Морзе – 4.14 знака (экономия около 13%).

·	E	0.13105	– – ·	D	0.03788	· – –	W	0.01539
–	T	0.10468	· – · ·	L	0.03389	– · · ·	B	0.01440
· –	A	0.08151	· · – ·	F	0.02924	· · · –	V	0.00919
– – –	O	0.07995	– · – ·	C	0.02758	– · –	K	0.00420
– ·	N	0.07098	– –	M	0.02536	– · · –	X	0.00166
· – ·	R	0.06882	· · –	U	0.02459	· – – –	J	0.00132
· ·	I	0.06345	– – ·	G	0.01994	– – · –	Q	0.00121
· · ·	S	0.06101	– · – –	Y	0.01982	– – · ·	Z	0.00077
· · · ·	H	0.05259	· – – ·	P	0.01982			

Код Морзе для кириллицы составлен с учетом совпадения с латинским оригиналом для букв, изображающих одинаковые звуки (ср. русские Б и Н с латинскими B и N). Поэтому коды не соответствуют частотам встречаемости букв в русскоязычных текстах. В таблице ниже приведены коды и соответствующие частоты (буквы Ь и Ъ, а также Е и Ё кодируются одинаково).

– – –	О	0.110	– –	М	0.031	– – – ·	Ч	0.015
·	Е	0.087	– · ·	Д	0.030	· – – –	Й	0.012
· –	А	0.075	· – – ·	П	0.028	· · · ·	Х	0.011
· ·	И	0.075	· · –	У	0.025	· · · –	Ж	0.009
–	Т	0.065	· – · –	Я	0.022	· · – –	Ю	0.007
– ·	Н	0.065	– · – –	Ы	0.019	– – – –	Ш	0.007
· · ·	С	0.055	– – · ·	З	0.018	– · – ·	Ц	0.005
· – ·	Р	0.048	– · · –	Ь	0.017	– – · –	Щ	0.004
· – –	В	0.046	– · · –	Ъ	0.017	· · – · ·	Э	0.003
· – · ·	Л	0.042	– · · ·	Б	0.017	· · – ·	Ф	0.002
– · –	К	0.034	– – ·	Г	0.016			

Без учета частот различных букв для 32-буквенного алфавита требуются в среднем $\log_2 32 = 5.00$ двоичных знаков на одну букву (для 31 букв – $\log_2 31 \approx 4.95$). С учетом реальных частот – 4.45 (экономия около 10%), а при использовании кода Морзе – несколько больше. Интересно, что аналогичный подсчет для частот букв в

тексте романа Л. Толстого “Война и мир” с учетом пробелов между словами дает более низкое значение: 4.35.

Стоит заметить, однако, что неравномерный код Морзе с разделителем был позднее заменен в телеграфии равномерным 5-значным двоичным кодом Бодо (со сменой регистра, когда одно и то же 5-значное число кодирует два или три разных символа) из-за малой пригодности для буквопечатающего приема. В коде Бодо не требуется разделителя, т.к. на приемном конце канала известно, что через каждые 5 элементарных сигналов кончается одна буква и начинается следующая. Применяются также 6-значные коды без смены регистра.

Коды с обнаружением и исправлением ошибок

1. Рассмотрим множество B^n слов длины n в алфавите B = {0, 1}. Каждое слово $x_1x_2...x_n$ в алфавите B отождествим с n-мерным вектором (x1,x2,...,xn). Суммой векторов $(x_1x_2...x_n)$ и $(y_1,y_2,...,y_n)$ из B^n назовем вектор $(x_1 \oplus y_1, x_2 \oplus y_2,..., x_n \oplus y_n)$. Напомним, что знак $\oplus$ означает сумму по модулю 2. Если ввести еще умножение произвольного вектора $X = (x_1,x_2,...,x_n)$ на элемент $b \in B$ следующим образом: $bX = (bx_1,bx_2,...,bx_n)$, то множество B^n можно рассматривать как n-мерное векторное пространство.

В векторном пространстве B^n можно определить **расстояние Хемминга d(X,Y)** между векторами X и Y как число несовпадающих компонент векторов X и Y и **норму** $\|X\|$ вектора X как расстояние между X и нулевым n-мерным вектором (0,0,...,0). Очевидно, что для любого слова $X = x_1x_2...x_n$ из B^n:

$$\|X\| = \sum_{j=1}^{n} x_j$$

и для любых X и Y из B^n:

$$d(X,Y) = \|X \oplus Y\|.$$

Метрическое пространство B^n допускает уже рассматривавшуюся нами простую геометрическую интерпретацию: множеству B^n соответствует множество вершин n-мерного единичного куба, а расстояние между двумя элементами из B^n равно минимальному числу ребер в цепи, соединяющей соответствующие вершины куба.

Для произвольного слова $X = x_1x_2...x_n \in B^n$ определим его *числовое значение* N(X):

$$N(X) = \sum_{j=1}^{n} x_j \cdot 2^{n-j}.$$

Примеры. Пусть X = 0110011 и Y = 0001101. Тогда = 4, = 3, X ⊕ Y = 0111110, d(X, Y) = = 5, N(X) = 1 + 2 + 16 + 32 = 51, N(Y) = 1 + 4 + 8 = 13. Заметим также, что, например, для числа 19 e(19) = 10011, $e_7(19)$ = 0010011.

В дальнейшем в этом разделе будут рассматриваться лишь коды, состоящие из $m \geq 2$ двоичных слов фиксированной длины, причем в тех случаях, когда нумерация слов кода не существенна, мы не будем ее указывать. Для произвольного кода $V = \{v_0,v_1,...,v_{m-1}\} \subseteq B^n$ положим:

$$d(V) = \min_{i \neq j} d(v_i, v_j).$$

Величина d(V) называется **кодовым расстоянием**.

Упражнение. Для кода V = {a_1: 0100, a_2: 1101, a_3: 1110, a_4: 0010} вычислить попарные расстояния $d(a_i, a_j)$, $i \neq j$ и кодовое расстояние d(V).

2. Рассмотрим некоторые виды преобразований двоичных слов, называемых **ошибками**.

Одиночной ошибкой вида 0 $\to$1 (1 $\to$0) в слове X называют результат замены одного из символов 0 (соответственно 1) символом 1 (соответственно 0). Одиночные ошибки этого вида называют также **замещениями** символов, или **аддитивными ошибками**.

Одиночной ошибкой вида 0 $\to$L (1 $\to$L) в слове X называют результат удаления одного из символов 0 (соответственно 1); при этом длина слова уменьшается на единицу. Одиночные ошибки этого вида называются **выпадениями символов**.

Одиночной ошибкой вида L $\to$0 (L $\to$1) называют результат *вставки символов* перед некоторым символом слова или после его последнего символа; при этом длина слова увеличивается на единицу.

Одиночной ошибкой вида $+2^i$ (-2^i) в слове X $\in B^n$, где $0 \le i < n$, называют преобразование слова X в слово Y $\in B^n$, числовое значение которого на 2^i больше (соответственно меньше) числового значения слова X. Одиночные ошибки вида $+2^i$ и -2^i называются **арифметическими ошибками**.

Для иллюстрации в таблице 3.6 приведены слова, полученные из слова 0001101 (двоичная запись числа 13) в результате ошибок рассматриваемых видов.

Типом ошибки будем называть некоторое множество видов одиночных ошибок. Например, ошибка типа {0 $\to$L, 1 $\to$L, L $\to$0, L $\to$1} есть выпадение или вставка произвольного символа. Число одиночных ошибок в некоторой их последовательности будем называть **кратностью** ошибки. Так, в результате ошибки кратности 3 (вставка 1 перед первым символом, затем выпадение 0 перед пятым и замещение последнего символа) слово 0001101 переводится в слово 1001100.

В дальнейшем особое внимание будет уделено ошибкам типа {(0 $\to$1), (1 $\to$0)}, т.е. замещениям символов.

При постановке различных задач обнаружения и исправления ошибок заданного типа исходят из того или иного допущения о законе образования ошибок. Эти законы могут иметь как вероятностный, так и комбинаторный характер. Говорят, что тот или иной закон образования ошибок задается каналом передачи (или хранения) двоичных слов. Наиболее часто рассматриваются следующие два типа каналов:

а) в любом символе с вероятностью p ($0 < p < 1/2$) происходит ошибка заданного типа, причем ошибки в различных символах статистически независимы;

б) в каждом слове длины n может произойти любая ошибка заданного типа кратности не более s.

Основная идея помехоустойчивого кодирования состоит в следующем. Вместо сообщений X, Y, Z,... по каналу связи посылают их определенным образом закодированные образы X', Y', Z',..., которые обеспечивают обнаружение ошибок или правильное декодирование даже при наличии ошибок допустимого типа.

Возможность исправления ошибок при использовании кода $V = \{v_0, v_1, ..., v_{m-1}\} \in B^n$ можно пояснить следующим образом. Обозначим через T(X) множество слов, которые можно получить из слова X в результате ошибок, *допустимых в рассматриваемом канале*, в частности $T(v_i)$ – множество слов, в которые может

превратиться в результате ошибок кодовое слово v_i. Произвольное однозначное отображение D множества $\bigcup_{i=0}^{m-1} T(v_i)$ на V будем называть **декодированием**. Задание декодирования D равносильно разбиению множества $\bigcup_{i=0}^{m-1} T(v_i)$ на непересекающиеся подмножества (**окрестности**) $D^{-1}(v_i)$, i = 0, 1,..., m–1, где каждая окрестность $D^{-1}(v_i)$ состоит из прообразов слова v_i при отображении D. Естественно считать, что при декодировании D в произвольном слове $v_i \in V$ исправляются те и только те ошибки, которые преобразуют слово v_i в некоторое слово из $T(v_i) \cap D^{-1}(v_i)$. В частности, для существования декодирования D, при котором исправляются все ошибки, допустимые в рассматриваемом канале, необходимо и достаточно, чтобы множества $T(v_i)$, i = 0, 1,..., m–1, попарно не пересекались. Тогда, получая по каналу связи некоторое сообщение Y, мы определяем, окрестности какого кодового слова X оно принадлежит, и делаем вывод, что послано именно слово X. Код, обладающий таким свойством, называется кодом, исправляющим ошибки данного типа.

Для кода $V \subseteq B^n$ с исправлением s ошибок типа замещения множество T(X) совпадает с *метрической окрестностью радиуса s слова X*, т.е. множеством точек, удаленных от X на расстояние не более s. Поэтому условие непересечения множеств $T(v_i)$ равносильно тому, что кодовое расстояние $d(V) > 2s$. Таким образом, *код V является* **кодом с исправлением s замещений** *тогда и только тогда, когда d(V) > 2s*, т.е. $d(V) \geq 2s + 1$, поскольку d(V) – целое число.

Наряду с задачами исправления ошибок можно рассматривать и более слабые задачи обнаружения ошибок. Очевидно, при использовании кода V нельзя обнаружить лишь те ошибки, которые преобразуют кодовые слова в другие кодовые слова. Отсюда, в частности, следует, что *код V с кодовым расстоянием d всегда позволяет обнаружить (d – 1) или менее одиночных ошибок типа замещения.*

3. Из сказанного выше можно заключить, что для обеспечения помехоустойчивости кодирования приходится применять более длинные коды, в которых кроме информации о самом сообщении содержится дополнительная информация, позволяющая при возникновении ошибок в процессе передачи (или хранения) сообщения обнаруживать или восстанавливать переданное сообщение. Этот факт можно сформулировать таким образом: за надежность приходится платить избыточностью кодирования.

Так, для кодирования 10 арабских цифр достаточно использовать не 9 двоичных знаков, как описано в примере 3 , а 4 ($2^4 = 16 > 10$), и соответственно минимизировать визуальный способ написания. Однако при таком способе кодирования среди $C_{10}^2 = 45$ пар кодов имеется 14 пар, различающихся ровно в одном разряде, и даже при одиночном замещении можно принять один из кодов за другой. В реально используемом кодировании десятичных цифр 9-значными двоичными числами только в одной паре (цифры 0 и 8) различие в одном разряде (что, по-видимому, приводит иногда к ошибкам, связанным с неясным заполнением или сбоем при прочтении); в остальных парах – не менее, чем в трех разрядах. Кроме того, принятый метод кодирования предпочтителен по другой причине: он удобнее, поскольку имеет сходство с обычным начертанием арабских цифр.

1 0 1 2 3 4 5 6 7 8 9

Простейший способ построения помехоустойчивых кодов – дублирование. Так, если каждую букву передаваемого слова повторить (s + 1) раз, то такой код способен обнаружить до s ошибок типа замещения: при отсутствии ошибок сообщение разбивается на однородные серии из (s + 1) одинаковых символов; если же произошли ошибки, то, даже если все s ошибок – внутри одной серии, можно обнаружить символ или несколько символов, которые, возможно, переданы с ошибкой, – те, в которых нарушена однородность.

Пример. Слово 01101 для возможности обнаружения до 2 ошибок можно закодировать как 000111111000111. Тогда при получении по каналу связи сообщения

000010111000111 (*)

следует разделить его на тройки: 000 010 111 000 111, после чего видно, что во втором символе – ошибка. Однако мы не можем восстановить передаваемое сообщение (т.е. достоверно утверждать, что посланное слово именно 01101 и произошла двойная ошибка во втором символе), поскольку такое же сообщение (*) могло быть получено из слова 00101 при одиночной ошибке во втором символе.

Аналогично, повторение каждого символа передаваемого слова (2s + 1) раз позволяет не только обнаружить, но и исправить до s ошибок типа замещения. Для предыдущего примера слово 01101 должно быть передано как 000001111111111000001111 1. При не более чем двух ошибках, даже если они внутри одной серии, большинство букв серии – не замещены и определяют правильное декодирование. Если, например, получено слово 000000110111111000001111 1, то после разбиения его на пятерки символов 00000 01011 11111 00000 11111 видно, во-первых, что ошибка произошла во втором символе, и, во-вторых, что исходная буква – 1 (число замещенных символов меньше половины длины серии).

4. Далее мы будем рассматривать только случай одиночной ошибки типа замещения. При использовании описанных простейших приемов для возможности обнаружения одиночной ошибки длина передаваемого слова удваивается, а для ее исправления – утраивается.

Однако возможно значительно более экономное помехоустойчивое кодирование. Идея таких методов заключается в следующем. На множестве двоичных слов рассматривается некоторая функция. Искомый код определяется как множество слов из B^n, на которых эта функция принимает некоторое фиксированное значение. Функция подбирается таким образом, чтобы в результате любой одиночной ошибки значение функции изменялось и чтобы по этому изменению и, быть может, самому полученному слову можно было однозначно определить вид и место ошибки.

Мы рассмотрим один пример такого кодирования – код Хэмминга.

Зафиксируем число n и найдем число l такое, что $2^{l-1} \le n < 2^l$. В этом случае $l = [\log n] + 1$. Например, $l(5) = l(7) = 3$, $l(8) = l(10) = l(13) = 4$.

Для произвольного слова $X = x_1x_2...x_n \in B^n$ положим

$$H(X) = x_1e_l(1) \oplus x_2e_l(2) \oplus ... \oplus x_ne_l(n).$$

H(X) представляет собой вектор длины l, полученный в результате сложения векторов, являющихся двоичными записями (с помощью l цифр) номеров единичных символов слова X.

Пример. Пусть n = 6, X = 010101 и Y = 110100. Тогда $l = l(6) = 3$,

H(X) = (0, 1, 0) E (1, 0, 0) E (1, 1, 0) = (0, 0, 0),

H(Y) = (0, 0, 1) E (0, 1, 0) E (1, 0, 0) = (1, 1, 1).

Теорема. **Код Хэмминга** H_n, состоящий из всех слов $X = x_1x_2...x_n \in B^n$ таких, что

$$H(X) = (0, 0,..., 0), \qquad (*)$$

является кодом с исправлением одного замещения.

В рассмотренном выше примере $X \in H_6$, $Y \notin H_6$.

Пример. Пусть в слове $X = 010101 \in H_6$ произошло замещение пятого символа, в результате чего получилось слово T = 010111. Так как H(T) = (0, 1, 0) ⊕ (1, 0, 0) ⊕ (1, 0, 1) ⊕ (1, 1, 0) = (1, 0, 1), то числовое значение слова H(Y) равняется 5 и определяет номер замещенного символа.

Заметим, что $|H_n|$ – число элементов кода H_n – равно 2^{n-l}. Это следует из того, что при произвольном задании значений (n – l) компонент, номера которых отличны от $2^0, 2^1,..., 2^{l-1}$ (**информационные** позиции), значения компонент с номерами $2^0, 2^1,..., 2^{l-1}$ (**проверочные** позиции) однозначно определяются из условия(*).

Так как $l = [\log n] + 1 =]\log (n + 1)[$, то

$$\frac{2^{n-1}}{n} \le |H| = 2^{n-]\log(n+1)[} \le \frac{2^n}{n+1}.$$

В частности, $|H_6| = 2^{6-]\log 7[} = 2^{6-3} = 2^3 = 8$. Код H_6 приведен ниже:

H_6
000000
111000
110011
001011
101101
010101
011110
100110

Упражнение. Проверьте, что любые два элемента кода H_6 различаются не менее, чем в трех разрядах.

Таким кодовым множеством можно кодировать все слова длины 3 – их тоже 8, сопоставив любым образом взаимно однозначно. Р.Хэммингом предложен способ кодирования, обеспечивающий простое и удобное декодирование. Для этого кодируемое слово X длины m дополняется l проверочными разрядами ($l =]\log (m + 1)[$), которые определенным образом рассчитываются при кодировании, и полученное сообщение X состоит из m информационных и l проверочных позиций. Для проверочных разрядов отводятся 1-й, 2-й, 4-й, 8-й и т.д., номера которых являются целыми степенями числа 2: их двоичные представления содержат ровно одну единицу. На остальные места: 3, 5, 6, 7, 9, 10,... помещают символы кодируемого слова X.

Покажем на двух примерах, как проводится кодирование по Хэммингу. Пусть кодируемое слово X = 1100: m = 4, l =]log 5[= 3. Закодированное сообщение длины m + l = 7 будет иметь вид $p_1p_2 1 p_4 100$. 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, т.е. в i-ом разряде справа: p_1 – в 1-ом разряде, p_2 – во 2-ом, p_4 – в 3-ем (таблица 3.7). $p_1 = p_3 \oplus p_5 = 0$; $p_2 = p_3 = 1$; $p_4 = p_5 = 1$. В результате получаем X' = 0111100.

i	i в двоичн. представлении	Информац. позиции	Проверочные позиции	Расчет проверочных символов	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. Кодируемое слово X длины m = 10: 1001110110; l = 4, вычисление проверочных разрядов X′ = 01110010110110.

i	i в двоичн. представ.	Информ. позиции	Проверочн. позиции	Расчет проверочных символов
1	0001		0	$p_3 \oplus p_7 \oplus p_9 \oplus p_{13}$
2	0010		1	$p_3 \oplus p_7 \oplus p_{10}$
3	0011	1		
4	0100		1	$p_7 \oplus p_{12} \oplus p_{13}$
5	0101	0		

i	i в двоичн. представ.	Информ. позиции	Проверочн. позиции	Расчет проверочных символов
6	0110	0		
7	0111	1		
8	1000		0	$p_9 \oplus p_{10} \oplus p_{12} \oplus p_{13}$
9	1001	1		
10	1010	1		
11	1011	0		
12	1100	1		
13	1101	1		
14	1110	0		

Докажем теперь, 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) r(X, Y) ≥ 3, т.е. X, Y различаются в трех или больше разрядах: при кодировании эти различия сохраняются; могут добавиться различия в проверочных разрядах, т.е. r(X', Y') ≥ 3.

2) r(X, Y) = 2, т.е. X, Y различаются в двух разрядах. Пусть в закодированных словах это позиции a ≠ b, числа a, b в двоичной записи различаются хотя бы в одном (i-ом) разряде, который для a равен 0, а для b 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 a, b, слова X и Y совпадают. Значит, в 2^i-ой проверочной позиции слова X' и Y' также различаются: в итоге $r(X', Y') \geq 3$.

3) $r(X, Y) = 1$, т.е. X, Y различаются ровно в одном разряде. Номер этого информационного разряда в словах X и 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 и Y различаются: снова $r(X', Y') \geq 3$.

Если при передаче кодированного слова X произошла одна ошибка, то при декодировании полученного слова T вектор H(T), прочитанный как двоичная запись, дает номер поврежденного разряда.