

Дискретная математика

Коченюк Анатолий

6 ноября 2020 г.

0.1 Введение

Связаться:

- stankev@gmail.com Собирать культуру общения: указывать Фамилию, Имя
- Телеграм @andrewzta (для немедленного ответа. Если нет, оно утонет).
- +79219034426 (для катастрофических ситуаций, ожидается, что звонить никто не будет) (ни в коем случае не писать смс)

Обращаться можно по методическим вопросам. Если проблема группы – пишет староста.

Не писать по учебно-методическим проблемам (общежитие, медосмотр, армия ..) для этого есть зам. декана Харченко (легко найти контакты в ису)

Про отчётность будет на первой практике.

Лекции есть в ютубе andrewzta

Глава 1

1 курс

1.1 Фундамент

Множество – неопределяемое понятие. Множество состоит из элементов.
 $a \in A$ а-маленькое принадлежит множеству А-большое

$$A = \{2, 3, 9\}$$

$$A = \{n \mid n \text{ чётно}, n \in \mathbb{N}\} - \text{фильтр}$$

$A, B :$

- $A \cup B = \{a \mid a \in A \text{ или } a \in B\}$
- $A \cap B = \{a \mid a \in A \text{ и } a \in B\}$
- $A \setminus B = \{a \mid a \in A \text{ и } a \notin B\}$
- $\overline{A} = \{a \mid a \notin A\}$??? U – универсум
 $\overline{A} = U \setminus A$
 $A \setminus B = A \cap \overline{B}$
- $A \triangle B = A \oplus B = (A \cup B) \setminus (A \cap B)$

Замечание. Если множество – любой набор чего-угодно возникает парадокс Рассела

$$A = \{a \mid a - \text{множество}, a \notin a\}$$

Вопрос лежит ли в себе A ?

Определение 1 (Пара). A, B – множества. Мы можем рассмотреть множество пар, где первый элемент из A , а второй из B

$$A \times B = \{(a, b) | a \in A, b \in B\}$$

$$A \times A = A^2$$

$$(A \times B) \times C = \{(x, y) | x \in A \times B, y \in C\} = \{((a, b), y) | a \in A, b \in B, y \in C\}$$

$$A \times (B \times C) = \{(a, (y, z)) | a \in A, y \in B, z \in C\}$$

$$A \times B \times C = \{(a, b, c) | a \in A, b \in B, c \in C\}$$

Для простоты, здесь и далее эта операция будет считаться ассоциативной и первые две строчки будут давать то же, что третья – множество троек.

$$A \times A \times A = A^3 \quad A^n = \begin{cases} A & , n = 1 \\ A \times A^{n-1} & , n > 1 \end{cases}$$

$$A^0 = \{\emptyset\} = \{\varepsilon\} \text{ – пустая последовательность.}$$

Пример. $A = 2, 3, 9 \rightarrow A \times A = \{(2, 2), (2, 3), (2, 9), (3, 2), (3, 3), \dots\}$

Замечание. У множества есть элемента и для любого элемента из универсума, он либо входит (1 раз) либо не входит.

Определение 2. Функция – отображение, которое каждому элементу из одного множества ставит в соответствие единственный элемент из другого множества

$$f : A \rightarrow B$$

График $\{(x, f(x))\}$.

Формально будем отождествлять функцию и её график.

$$f \subset A \times B \quad \forall a \in A \exists! b \in B \quad (a, b) \in f$$

Замечание. Не путайте принадлежность и включение

$$a \in A$$

$$A, B, \forall a \text{ (если } a \in A, \text{ то } a \in B) \quad A \subset B$$

$$D_4 = \{n | n \text{ кратно } 4\}$$

$$E = \{n | n \text{ чётно}\}$$

$$D_4 \subset E$$

$$\{2, 3, 9\} \subset \{2, 3, 4, \dots, 9\}$$

$$A \subset A$$

$$\emptyset \subset A$$

$$A \subset U$$

Замечание. Не обязательно все b попадают в график.

$sqr : \mathbb{N} \rightarrow \mathbb{N}$ – только квадраты чисел

Определение 3. $\forall b \in B \exists a \in A : b = f(a)$ – сюръекция

Определение 4. $\forall a \in A \forall b \in B \quad a \neq b \implies f(a) \neq f(b)$

Замечание. Принцип Дирихле – нет инъекции из большего в меньшее множества. Если кроликов больше, чем клеток, то какому-то кролику не хватит клетки

Определение 5. Если f – инъекция и сюръекция, то f – называется биекцией

Если между двумя конечными множествами есть биекция, то у них равное количество элементов.

Определение 6. Два множества называется равномоощными, если между ними есть дикция

B^A – множество функций из A в B

$$|A| = a, |B| = b \quad |A \times B| = a \cdot b \quad |B^A| = b^a$$

$|A^\emptyset| = 1$ эфемерная функция, которой ничего не передать

$$\emptyset^A = \emptyset, A \neq \emptyset$$

$$\emptyset^\emptyset = 1$$

Определение 7. $R \subset A \times B$ – отношение (бинарное)

Пример. $A = B = \mathbb{N} \quad R = \{(a, b) | a < b\} \quad R = <$

$$a : b \quad 6 : 2 \quad 6 \not: 5$$

A = люди, B = собаки, $R = \{(a, b) | a - \text{хозяин} b\}$

Рассмотрим 5 классов отношение на квадрате множества:

1. рефлексивные $\forall a \quad aRa$

$RC(R)$ – рефлексивное замыкание, включаем все пары (a, a)

2. антирефлексивные $\forall a \quad \neg aRa$

3. симметричные $aRb \implies bRa$

4. антисимметричные $aRb, a \neq b \implies \neg bRa$

или aRb и $bRa \implies a = b$

5. транзитивность $aRb, bRc \implies aRc$

Определение 8. 1+3+5 – рефлексивные, симметричные и транзитивные – называются отношениями эквивалентности.

Теорема 1. R – отношение эквивалентности на X , то элементы X можно разбить на классы эквивалентности так, что:

a и b в одном классе $\implies aRb$ и a и b в разных классах $\implies \neg aRb$

множество таких классов обозначается X/R

$N/\equiv_3 =$

$$\begin{aligned} &\{\{1, 4, 7, 10, \dots\} \\ &\{2, 5, 8, 11, \dots\} \\ &\{3, 6, 9, 12, \dots\}\} \end{aligned}$$

Замечание. Отношение равномощности – отношение эквивалентности.

Классы эквивалентности – порядки. Для конечного случая обозначаются числами

Определение 9. 1+4+5 – рефлексивные, антисимметричные и транзитивные – частичные порядки

Множество, на котором введён частичный порядок, то оно называется частично упорядоченным. (ч.у.м – частично упорядоченное множество, poset – partially organised set)

$R \subset X \times X$

$X, Y, Z \quad R : X \times Y \quad S : Y \times Z$

Определение 10. Композиция отношений:

$$T = R \circ S \quad xTy \iff \exists z : xRz \text{ и } zSy$$

т.е. есть z , через который можно пройти, чтобы попасть в y из x

Замечание. $R \subseteq X \times X \quad S \subseteq X \times X$

$$R \circ S \subseteq X \times X$$

$R \circ R \subseteq X \times X$ – пройти два раза по стрелкам

$$R^3 = R \circ R^2 = R^2 \circ R \text{ – пути длины ровно } 3$$

$S \circ T \circ U$ – идём по стрелке из S в T , а потом в U

Определение 11. Транзитивное замыкание.

$$R^+ = \bigcup_{k=1}^{\infty} R^k$$

$R^0 = \{(x, x) | x \in X\}$ – они не включаются по дефолту в R^+

$R^* = \bigcup_{k=0}^{\infty} R^k = R^+ \cup R^0$ – если между двумя вершинами существует какой-либо путь

Замечание. Транзитивное замыкание – транзитивно

$$\text{Пусть } xR^+y \implies xR^iy$$

$$\text{Пусть } yR^+z \implies yR^jz$$

$$\implies x(R^i \circ R^j)z \implies xR^kz$$

Замечание. $\forall T : T \text{ – транзитивно. } T \subset R \implies T^+ \subset R$

Доказательство. По индукции:

База: $R^1 \subset T$ – дано

Переход: $R^i \subset T \implies R^{i+1} \subset T$

$xR^{i+1}y \implies x(R \circ R^i)y \implies \exists z : xRz \& zR^iy \implies xRz \& zTy \implies xTy$ (по транзитивности T) ■

1.2 Булевы функции

\emptyset – пустое множество. С функциями из/в него всё достаточно грустно.

$\{unit\}$

void – ничего, константная функция

$$\mathbb{B} = \{0, 1\}$$

$f : A_1 \times A_2 \times \dots \times A_n \rightarrow B$ – функция от нескольких аргументов. Из одного, но декартового произведения

Булева функция: $f : \mathbb{B}^n \rightarrow B$

$n = 0$ – ноль аргументов $\mathbb{B}^0 = \{\emptyset\}$

$\emptyset, 1$

$n = 1$

Таблица 1.1: n=1

x	\emptyset	id	\neg	1
0	0	0	1	1
1	0	1	0	1

Замечание. Подобные таблицы называются таблицами истинности функций

$n = 2$

Таблица 1.2: n=2

x	y	\emptyset	\wedge	\nrightarrow	P_1	\neq	P_2	\oplus	\vee	\downarrow	$=$	$\neg P_2$	\leftarrow	$\neg P_1$	\rightarrow	\uparrow	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

С помощью стрелки Пирса (\downarrow) и штриха Шеффера (\uparrow) можно выразить любую другую: $\neg x = x \downarrow x$

1.3 Задания булевых функций

Самый простой способ – таблица истинности

$\oplus_n - 2^n$ значений. глупо их все отдельно описывать

1. Задание функции формулой.

Определим базисные функции, систему связок

например: $\wedge, \vee, \neg, \oplus$

$$x_1 \oplus x_2 \oplus x_3 \dots$$

$\{f_1, f_2, \dots, f_n\}$ – базисные.

строка – формула. $f_i(x_1, \dots, x_k)$ – формула

Определение 12. Дерево разбора формулы. Если у функции аргументность – k , то у ноды будет ровно k сыновей

\overline{F} – функции, которые записываются формулами, используя F (замыкание F)

Теорема 2 (Теорема о стандартном базисе). $\overline{\{\wedge, \vee, \neg\}} = \mathbb{B}$

Доказательство. Рассмотрим таблицу истинности функции f . Она принимает n аргументов и в ней 2^n строк

Пусть $f \neq 0$. Рассмотрим строчки, в которых единицы.

По аргументам запишем с не – аргументы, которые 0, и без не – те, которые 1

$\neg x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4 \wedge x^5 = 1$ на ровно одном наборе элементов. А теперь возьмём "или" по всем строкам, в которых 1

Одна такая строка называется термом.

Такая форма называется совершенной дизъюнктивной нормальной формой

■

Лемма 1. Любая функция, кроме тождественного 0 – есть СДНФ

$x \vee \neg x$ – тождественный ноль

Напоминание о способах задания функций:

$F \quad x_1, x_2, \dots, x_n f \in F$

$or(and(x, not(y)), or(0, z))$. Такие формы называются формулами. По формуле можно построить дерево разбора.

\wedge, \vee, \neg

СДНФ – дизъюнкция термов, где каждый терм – конъюнкция литералов. Совершенная – в каждом терме есть все переменные по одному разу

Лемма 2. $\sqsupset F$ – некоторое множество. $\overline{F} = \mathbb{B}F$

$\sqsupset G$ – некоторое множество функций $\forall f \in F \quad f \in \overline{G}$

Тогда с помощью G можно выразить любую функцию $\overline{G} = \mathbb{B}F$

Доказательство. $G \rightarrow F \rightarrow \forall \implies G \rightarrow \forall$ – то, что нужно доказать

фиксируем функцию $h \in \mathbb{BF}$. Она каким-то деревом разбора выражается через функции $f \in F$. Каждая функция f выражается через $g \in \overline{G}$, тогда подставим выражения функций f через g в узлах дерева и получим выражение функции h через \overline{G} , значит любая функция выражается через $\overline{G} \implies \overline{G} = \mathbb{BF}$ ■

Пример. $\{\oplus, \wedge, 1\}$

$x \wedge y = x \wedge y \quad \neg x = x \oplus 1$ – такая запись называется полиномом жегалкина

$$x \vee y = (x \wedge y) \oplus x \oplus y$$

$x \wedge y = xy \oplus y \oplus x - \wedge$ опускают

$$(x \oplus y)(y \oplus z) = xy \oplus y \oplus xz \oplus yz$$

$$(x \oplus 1)(y \oplus 1) = xy \oplus x \oplus y \oplus 1$$

$a \wedge a = a$ – идемпотентность

Теорема 3. Любая булева функция (кроме \emptyset) имеет каноничный полином, причём единственный (с точностью до коммутативности и ассоциативности)

Доказательство. булевых функций от n аргументов – 2^{2^n}

Мономов – 2^n . Каждый из них мы можем взять или не взять \implies всего $2^{2^n} - 1$, -1 из случая, где мы рассматриваем пустую сумму.

Есть инъекция из булевых функций в полиному Жегалкина. Это инъекция между равномошными множествами \implies это биекция. ■

1.4 Линейный функции

Полиному Жегалкина, в которых нету \wedge

$$x \oplus y \quad x \oplus y \oplus 1$$

Определение 13. Функция называется линейной, если её канонический полином Жегалкина не содержит \wedge

Утверждение 1. Если F содержит только линейные функции, то и \overline{F} содержит только линейные функции

Доказательство. $x_1 \oplus x_2 \oplus x_3$

$x_7 \oplus x_8 = (x_1 \oplus x_2 \oplus x_3) \dots$ Заменяем и получаем всё ещё сумму переменных или 1

Если формально, строим дерево, заменяем узлы на линейные функции, заменяем повторы, раскрываем скобки (пользуемся ассоциативностью \oplus) и получаем линейную функцию. ■

Утверждение 2. Если F содержит только функции, сохраняющие 0, то и \bar{F} тоже
аналогично для 1

Определение 14. Функция f называется монотонной \iff для двух наборов x_1, x_2, \dots, x_n y_1, y_2, \dots, y_n , что $x_i \leq y_i$ $0 < 1$

$$f(x_1, x_2, \dots, x_n) \leq f(y_1, y_2, \dots, y_n).$$

Утверждение 3. Из монотонных функций не выразить немонотонную

Доказательство. Доказывается индукцией по дереву разбора. Увеличили аргумента, увеличился уровень выше, выше и корень тоже ■

Определение 15. Функция f называется самодвойственной, если
 $f(x_1, \dots, x_n) = \neg f(\neg x_1, \dots, \neg x_n)$

Утверждение 4. Из самодвойственных функций тоже не выйти. Также деревом разбор

Классы Поста:

1. F_0 – сохраняющие 0
2. F_1 – сохраняющее 1
3. F_l – линейные
4. F_m – монотонные
5. F_s – самодвойственные

Лемма 3. $F \subseteq F_i, i \in \{0, 1, l, m, s\} \implies \overline{F} \subseteq F_i$

Следствие 1. \overline{F} – не полно

Теорема 4 (критерий Поста). F – полное $\iff F \not\subseteq F_i$ для всех $i \in \{0, 1, l, m, s\}$

Доказательство. \implies Если нет, то все функции лежат внутри этого класса. Не будет включена \uparrow например, не лежащая ни в одном классе Поста

$$\iff f_0 \notin F_0, f_1 \notin F_1, f_l \notin F_l, f_m \notin F_m, f_s \notin F_s$$

$$a(x)f_0(x, x, \dots, x)$$

$$a(0) = 1$$

$$\text{a } a(1) = 1 \implies a(x) = 1$$

$$\text{b } a(1) = 0 \implies a(x) = \neg x$$

$$b(x) = f_1(x, x, \dots, x) \quad b(1) = 0$$

$$1. \quad b(1) = 0 \implies b(x) = 0$$

$$2. \quad b(1) = 1 \implies b(x) = \neg x$$

$$1\text{a } 1 \quad 0$$

$$1\text{b } 0, \neg$$

$$2\text{a } 1, \neg$$

$$2\text{b } \neg, x$$

$$1\text{a } 1, 0 \quad f_m(x_1, \dots, x_n) > f_m(y_1, \dots, y_n) \quad x_i \leq y_i \quad \text{Значит первое} - 1, \text{ а второе} - 9$$

$$f_m(x_1, \dots, x_n)$$

$$f_m(y_1, \dots, x_n)$$

$$f_m(y_1, \dots, x_n)$$

$$\vdots$$

$$f_m(y_1, \dots, y_n)$$

В какой-то момент единица сменилась нулём на соседних строках

$$f(y_1, \dots, y_{i-1}, x_i, \dots, x_n) = 1$$

$$f(y_1, \dots, y_{i-1}, y_i, \dots, x_n) = 0$$

$$x_i \leq y_i \quad x_i \neq y_i \implies x_i = 0, y_i = 1$$

$c(z) = f_m(y_1, \dots, y_{i-1}, z, x_{i+1}, \dots, x_n)$ здесь вместо x и y подставлены константы

$$c(z) = \neg z$$

$$2b \quad f_s \quad x_1, x_2, \dots, x_n : f_s(x_1, x_2, \dots, x_n) = \neg f(\neg x_1, \dots, \neg x_n) = t$$

$$d(z) = f_s(z^{x_1}, z^{x_2}, \dots, z^{x_n}) \quad x^y = \begin{cases} x & , y = 1 \\ \neg x & , y = 0 \end{cases}$$

$$d(0) = t, d(1) = t$$

$$\begin{cases} t = 1 \implies d(t) = 1 \\ t = 0 \implies d(t) = 0 \end{cases}$$

Итак мы получили $1, 0, \neg$

Воспользуемся нелинейной функцией: f_l среди нелинейных членов в полиноме Жегалкина выберем тот, в котором меньше всего переменных. Не умаляя общности скажем, что он выглядит как $xyu_1 \dots u_k \quad k + 2 \geq 2$

$h(x, y) = f_l(x, y, 1, 1, \dots, 1, 0, 0, \dots, 0)$ Вместо u_k подставляем 1 , а вместо остальных 0

$h(x, y) = xy[\oplus x][\oplus y][\oplus 1]$ – восемь вариантов.

Если есть $\oplus 1$, напомним \neg

$$xy[\oplus x][\oplus y]$$

$$xy = x \wedge y$$

$$xy \oplus x \oplus y = x \vee y$$

$$xy \oplus x \quad h(x, \neg y) = x(y \oplus 1) \oplus x = xy$$

$$xy \oplus y \quad h(\neg x, y) = (x \oplus 1)y \oplus y = xy \quad \blacksquare$$

1.5 Преобразование Мёбиуса

$$f(x_1, x_2, \dots, x_n) = x \vee y/x/y/1$$

$$a_{xy}xy \oplus a_x x \oplus a_y y \oplus a_1$$

$$f(x_1, x_2, \dots, x_n) = \bigoplus_{\vec{s} \in \mathbb{B}^n} a_{\vec{s}} \prod_{i: s(i)=1} x_i = \bigoplus_{\vec{s} \leq \vec{x}} a_{\vec{s}}$$

$$s(i) = 1 \implies x(i) = 1 \iff s \& x = s \iff s \leq x \text{ (покомпонентно)}$$

Определение 16 (Доминирование). $\vec{a} \leq \vec{b} \iff \forall i \quad a_i \leq b_i$

$$\begin{array}{cccc|c} & 0 & 0 & 0 & \dots & f_{00\dots 0} \\ & 0 & 0 & \dots & 1 & f_{00\dots 1} \\ \text{Таблица истинности:} & & & & & \\ & 1 & 1 & \dots & 1 & f_{11\dots 1} \end{array}$$

$$f \in \mathbb{B}^{2^n}$$

$$\vec{a} = M\vec{f} \quad \vec{f} = M\vec{a}$$

$$M_{xs} = [s \leq x]$$

$$\text{Преобразование Мёбиуса – матрица } M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Теорема 5. Преобразование матрицы – инволюция ($M = M^{-1}$)

$$\vec{a}_t = \bigoplus_{x \leq t} f_x$$

$$\text{Доказательство. } \bigoplus_{x \leq t} f_x = \bigoplus_{x \leq t} \bigoplus_{s \leq x} a_s = \bigoplus_{s, x: s \leq x \leq t} a_s = \bigoplus_S [(\#x : s \leq x \leq t) \% 2] a_s = a_t$$

1. $s \not\leq t \implies \#x = 0$
2. $s = t \implies \#x = 1, s = x = t$
3. $s \leq t_1 \quad s \neq ts$ – нечётное число раз ксориться. z различных разрядов,
 $z \leq 1 \quad 2^z$

■

$$\text{Пример. } \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} a_{11} = 1, a_{01} = 0, a_{10} = 0, a_{00} = 1$$

$xy \oplus 1$ – штрих Шефера

1.6 Схемы из функциональных элементов (Boolean Circuits)

Определение 17. Топологической сортировкой называется отображение $\varphi : V \rightarrow \{1, \dots, n\}$ $u \neq v \implies \varphi(u) \neq \varphi(v)$ $uv \in E \implies \varphi(u) < \varphi(v)$

Теорема 6. Ациклический ориентированный граф имеет топологическую сортировку.

Лемма 4. Если G ациклический граф, то существует вершина, из которой не выходит рёбер

Доказательство леммы. Возьмём вершину: если

■

Доказательство теоремы. $n = 1$ дадим единственной вершине номер 1

$n > 1$ — возьмём вершину из которой нет рёбер, дадим ей номер n и удалим её из графа. Граф от этого не стал иметь циклов, поэтому по индукционному предположению мы можем занумеровать оставшиеся $n - 1$ элементов ■

Вершины, в которых нет рёбер называются x_1, x_2, \dots, x_n . Дальше идут внутренние вершины, обозначаемые функциями. Например, если обозначена \wedge , то в неё входит два ребра. Если некоммутативная функция, то указывается порядок. Исходящая степень может быть любой. Завершает всё вершина выхода

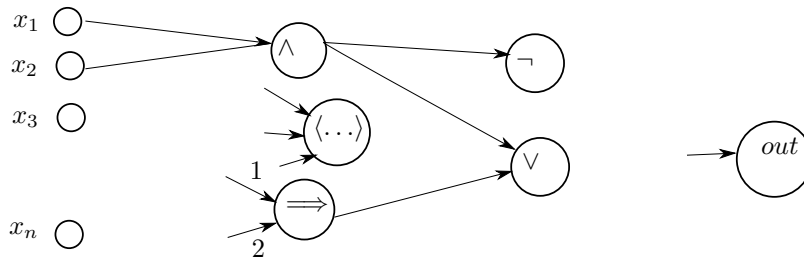


Рис. 1.1: sceme

$$x \oplus y = (x \wedge \neg y) \vee (\neg x \wedge y)$$

Дерево разбора легко превращается в схему.

Теорема 7. Не существует формулы $len(\phi) = \tilde{O}(n)$ для \oplus_n в $\{\wedge, \vee, \neg\}$

В схеме мы можем пересипользовать то, что в формуле пришлось бы повторять.

B – базис

Теорема 8. Функцию f можно задать формулой в базе $B \iff f$ можно представить схемой

Определение 18. Сложностью функции f в базисе B $size_B(f) = \min$ число функциональных элементов в схеме.

Определение 19. Глубина схемы определяется рекурсивно: глубина входов – 0, глубина вершины – максимум из глубины входящих + 1 $depth_B(f)$ – минимальная глубина схемы для функции.

Теорема 9. B_1, B_2 – базисы.

$\exists c \quad \forall f \quad size_{B_1}(f) \leq c \cdot size_{B_2}(f)$

Доказательство. $B_2 = \{b_1, b_2, \dots, b_n\}$

b_i выразим через B_1

$C \leq \max_{b_i \in B_2} size_{B_1}(b_i)$

(оптимальная схема может быть лучше, поэтому \leq) ■

Теорема 10. То же самое про глубину

Следствие 2. $size(f)$ без базиса – асимптотическое поведение не зависящее от базиса (по теоремам при переходе к другому базису всё отличается в константу)

Следствие 3. $c_1 size_{B_2}(f) \leq size_{B_1}(f) \leq c_2 size_{B_2}(f)$

Размер функции с точностью до константы не зависит от базиса

1.7 Конкретные схемы для логических операций

Числа храниться в виде двоичного кода. Занумеруем в двух числах биты:
 $x_0, \dots, x_n, y_0, \dots, y_n$

Побитовое И – n элементов \wedge принимающие соответствующие разряды.

$$z_0 = x_0 \wedge y_0 \dots z_n = x_n \wedge y_n$$

Размер схемы: n глубина: 1 $size = n$ $depth = 1$

Побитовое ИЛИ – так же. Любая побитовая операция – так же.

Арифметические операции – не так же. Биты начинают зависеть друг от друга.

Сложение двух битов: заведём два выходных бита: $low = a \oplus b$ $high = a \wedge b$. Такая схема называется неполным сумматором. Неполным, потому что из него не собрать сумматор для целых чисел. Для второго бита понадобится сложить биты чисел и ещё бит переноса. Но сумма трёх битов, к счастью, все ещё помещается в два бита $1 + 1 + 1 = 3 = 11_2$

a, b, c $low = \oplus_3(a, b, c)$ $high = med_3(a, b, c)$ – полный сумматор. Первому биту на перенос подаётся 0, а для остальных будут складываться соответствующие биты и перенос с предыдущих битов. Другое название – линейный сумматор.

$$size = n \quad depth = n$$

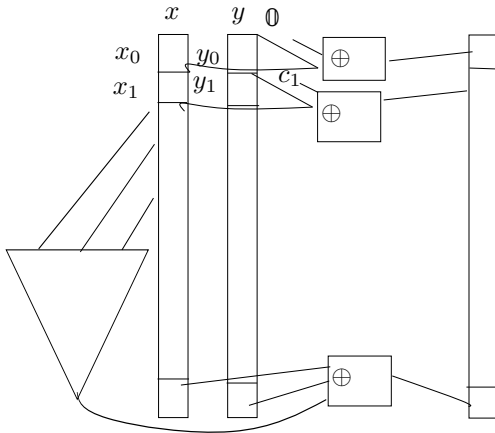


Рис. 1.2: sum

0	0	0	k (kill)
0	1	x	p (propagate)
1	0	x	p
1	1	1	g (generate)

1 \ 2	k	g	p	
k	k	g	k	$a(bc) = (ab)c = abc$ – композиция ассоциативна!
g	k	g	g	
p	k	g	p	

Схема композиции: принимает четыре значения, выдаёт два. Имеет константную глубину.

(Дальше жёсть, которую я не могу нарисовать, но суть в том, что раз оно ассоциативное, то мы можем запилить двоичное дерево и делать всё за радостный логарифм.)

$size = O(n)$ $depth = O(\log n)$ – Двоичный каскадный сумматор. Лучше сделать нельзя.

$-y = (\sim y) + 1$ отрицательные числа хранятся как дополнение +1

$x - y = x + (\sim y) + 1$. Отрицание y сделать легко, но как добавить ещё 1? Но у нас есть нулевой перенос в нулевой разряд. Давайте сделаем его $c_0 = 1$

		1	0	1	1
		1	1	0	1
		<hr/>			
		1	0	1	1
	0	0	0	0	
1	0	1	1		
1	0	1	1		
<hr/>					

Умножать двоичные числа в столбик просто. Схема даже имеет название Матричный умножитель

Дерево Уоллиса: Во-первых превратим сумму трёх чисел в сумму двух. Для трёх чисел поразрядно сделаем сумматор, который будет возвращать сумму и перенос побитого. Здесь мы не передаём перенос никуда. Дальше из переносов сделаем число и из сумм сделаем число. Получим два числа и нам нужно сложить уже их.

1.8 Линейные программы

Определение 20. x_1, x_2, \dots, x_n – переменные

$x_{n+1}, x_{n+2}, \dots, x_{n+t}$ – дополнительные t переменных.

Для базиса (например \vee, \wedge, \neg):

$$x_{n+1} = x_2 \vee x_7$$

$$x_{n+2} = \neg x_4$$

$$x_{n+3} = x_{n+1} \wedge x_{n+2}$$

$$\vdots$$
$$\cdot$$

В дополнительных переменных разрешается одна функция из базиса применённая к предыдущим переменным.

Пример. Сделаем \oplus

x_1, x_2

$$x_3 = \neg x_1$$

$$x_4 = \neg x_2$$

$$x_5 = x_1 \wedge x_4$$

$$x_6 = x_2 \wedge x_3$$

$$x_7 = x_5 \vee x_6$$

$$\cdot$$

Теорема 11. \exists схема из функциональных элементов длины $t \iff \exists$ линейная программа длины t

Доказательство. Если на схеме задать топологическую сортировку (пронумеровать так, чтобы стрелки были из меньшего числа к большему, то можно идти по полученным номерам: сначала сделать доп. переменные от входов, потом уже зависящие не только от них, но от уже заведённых согласно схеме.

Обратно: каждой доп. переменной соответствует применение функции (функционального элемента) к уже полученным. В этот элемент идут аргументы из определения доп. переменной, а из неё, соответственно её значение. ■

Замечание. Линейных программ больше, чем схем из функциональных переменных:

$\begin{cases} x_3 = \neg x_1 \\ x_4 = \neg x_2 \end{cases}$ и $\begin{cases} x_3 = \neg x_2 \\ x_4 = \neg x_1 \end{cases}$ приводят к одному результату и одной схеме,
 но это различные линейные программы

$\{\downarrow\}$ – базис.

$$n^2 \cdot (n+1)^2 \cdot \dots \cdot (n+t-1)^2 \leq (n+t)^{2t}$$

Лемма 5. Схем из t функциональных переменных $\leq (n+t)^{2t}$

$$\frac{2^n}{3^n} \text{ Схем } c \leq \left(n + \frac{2^n}{3^n}\right)^{\frac{2 \cdot 2^n}{3^n}}$$

$$\alpha \leq \frac{\left(n + \frac{2^n}{3^n}\right)^{\frac{2 \cdot 2^n}{3^n}}}{2^{2^n}} - \text{ для функций, которые можно реализовать за } \frac{2^n}{3^n} \text{ элементов}$$

$$\log_2 \alpha \leq \frac{2^{2^n}}{3^n} \log_2 \left(n + \frac{2^n}{3^n}\right) - 2^n = 2^n \left(\frac{2}{3^n} \log_2 \left(n + \frac{2^n}{3^n}\right) - 1\right) \leq 2^n \left(\frac{2}{3^n} \cdot n - 1\right) \leq -\frac{1}{3} 2^n$$

$$\alpha \leq 2^{-\frac{1}{3} 2^n} \leq \left(\frac{1}{\sqrt[3]{2}}\right)^{2^n} \rightarrow 0, n \rightarrow \infty$$

$$\exists n_0 : n > n_0 \implies n + \frac{2^n}{3^n} \leq 2^n$$

Теорема 12. $\forall c > 0 \quad g(n) \leq \frac{2^n}{3^n} \quad \exists n_0 : n > n_0$, то (доля функций от n аргументов, которые можно реализовать с помощью $g(n)$) $\leq c$

Или (доля функций ...) $\rightarrow 0, n \rightarrow \infty$

Доказательство. $f(x_1 x_2 \dots x_k y_{k+1} \dots y_n)$

Рассмотрим таблицу, где по горизонтали указывается набор x -ов, а по вертикали – y

$$x_1 \oplus y_2 \oplus y_3$$

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

Разобьём таблицы на горизонтальные полосы длины s

Столбцы $a \sim_j b$ – равны в j полосе – отношение эквивалентности

$$\text{Число полос } p = \frac{2^k}{s}$$

\exists не более чем 2^s классов эквивалентности.

Для полосы j и маски m g_{jm} – значения маски в полосе, за её пределами – 0

Теперь возьмём мультиплексор (n входов, 2^n выходов, 1 на выходе с числом $(x_1 \dots x_n)_2$). Выделим в нём полосу j , в неё проогим те значения, которые могут быть 1

$$f(x_1 \dots x_k, y_{k+1} \dots y_n) = \bigvee_{j=1}^p g_{jm_j}$$

$$\text{Суммарно: } 2^k + 2^{k+s} + 2^{n-k} + 2^{n-k} \cdot \frac{2^k}{s} + 2^{n-k} + 2^{n-k} = O\left(2^{k+s} + \frac{2^n}{s}\right)$$

Теперь возьмём $k = \log_2 s$, а $s = n - 2 \log_2 n$

$$2^{k+s} + \frac{2^n}{s} = 2^{n-\log_2 n} + \frac{2^n}{n-2 \log_2 n} = O\left(\frac{2^n}{n}\right)$$

■

Определение 21. Алфавит Σ – любое непустое конечное множество.

Последовательность символов: $\Sigma^2 \ \Sigma^3 \dots \bigcup_{k=0}^{\infty} \Sigma^k =: \Sigma^*$ – множество всех слов (или подстрочек) над алфавитом Σ

$$\Sigma^0 = \{\varepsilon\}$$

α, β – два слова.

Определение 22. $\alpha\beta$ – конкатенация $\Sigma^* \times \Sigma^* \rightarrow \Sigma^*$

$$\alpha \in \Sigma^k \quad \beta \in \Sigma^l \quad \gamma = \alpha\beta \in \Sigma^{k+l}$$

$$\gamma[i] = \begin{cases} \alpha[i] & , i \leq k \\ \beta[i - k] & , i > k \end{cases}$$

Свойства конкатенации:

1. $(\alpha\beta)\gamma = \alpha(\beta\gamma)$
2. $\alpha\varepsilon = \varepsilon\alpha = \alpha$

Структуру с ассоциативностью и нейтральным элементом называют моноидом

Определение 23. Σ, Π – алфавиты

Обобщённым кодом ϕ называется функция

$$\varphi : \Sigma^* \rightarrow \Pi^*.$$

Определение 24. Код называется декодируемым (или однозначным), если $\alpha \neq \beta \implies \varphi(\alpha) \neq \varphi(\beta)$

Или, что то же самое, φ – инъективная функция.

Замечание. $zip : \Sigma^* \rightarrow \Sigma^*$ – однозначное декодируемый. Не требует, чтобы любая последовательность символов была валидным кодом, в который могло что-то зашифроваться.

$jpeg : \Sigma^* \rightarrow \Sigma^*$ – сжатие с потерями. Когда декодируем, получаем другой файл. Несколько файлов могут сжаться в один код.

png – сжатие без потерь

Транслитерация фамилий в паспорте $A \rightarrow A \quad C \rightarrow S \quad Ч \rightarrow CH$

Определение 25. Разделяемый код: каждый символ кодирует отдельно $\varphi : \Sigma \rightarrow \Pi^*$

$$\varphi(c_1 c_2 c_3 \dots c_n) = \varphi(c_1) \varphi(c_2) \dots \varphi(c_n)$$

На время будем считать $\Sigma = \Pi$

Утверждение 5. Не существует кода $\Sigma^* \rightarrow \Sigma^*$, который не увеличивает любой текст, а некоторые уменьшает

Доказательство. Длины 0 меньше точно закодировать

Длины 1 не можем опять.

Длины 2, опять та же проблемы, все тексты меньше уже заняты. ■

Замечание. Но zip то всё сжимает..

(zip архив точно не сожмёт дальше)

S – строка. Хотим построить для неё оптимальный код. Какой?

$\Sigma = \{c_1, c_2, \dots, c_n\}$ p_i – количество вхождений c_i в S

$\varphi : \Sigma \rightarrow \mathbb{B}^*$ – двоичный код. $l_i = \text{len}(\varphi(c_i)) \quad \text{len}(\varphi(s)) = \sum_{i=1}^k l_i p_i$

- Префиксный код
- код Хаффмана
- неравенство Крафта-МакМиллана

Определение 26. φ – префиксный код, если

$$\forall a, b \in \Sigma \quad \varphi(a) \text{ не префикс } \varphi(b).$$

Пример. $\begin{matrix} a & 0 \\ b & 00 \\ c & 11 \end{matrix}$ Это не префиксный код, потому что a префикс b

$\begin{matrix} a & 0 \\ b & 00 \\ c & 11 \end{matrix}$

Это уже префиксный код

0	0	10	10	11	11	11	10	0
a	a	b	b	c	c	c	b	a

Лемма 6. Префиксный код однозначно декодируемый

Можно строить дерево двоичного кода.

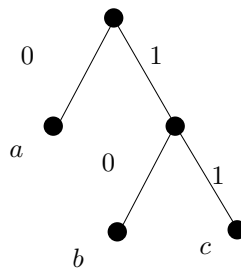


Рис. 1.3: tree

Символам, которые встречаются чаще, хотелось бы выдать меньший код

Задача 1. Префиксный код, $\sum l_i p_i \rightarrow \min$

Лемма 7 (1). \exists дерево оптимального, когда два символа с минимальным p_i являются братьями на максимальной глубине.

Доказательство. Рассмотрим дерево, рассмотрим две минимальные вершины. Не может быть, чтобы брата не было (иначе у минимальной вершины можно было бы отрезать последний символ, оставив код префиксным).

Если два брата соответствуют минимальным p_i – всё.

Если нет, p_i, p_j – минимальные p_k, p_l – самые глубокие

p_i, p_j – два самых минимальных $\implies p_j \leq p_k, p_j \leq p_l$

p_k, p_l – два самых глубоких $\implies l_i \leq l_k, l_j \leq l_l$

$$\sum_t l_t p_t = \sum_{t \neq i, j, k, l} l_t p_t + p_i l_i + p_j l_j + p_k l_k + p_l l_l$$

$$\sum_t l'_t p_t = \sum_{t \neq i, j, k, l} l_t p_i + p_j l_k + p_j l_l + p_k l_i + p_l l_j$$

$$\text{Их разность} = p_i(l_i - l_k) + p_j(l_j - l_l) - p_k(l_i - l_k) - p_l(l_j - l_l) \quad \blacksquare$$

Пример.

a	b	c
2	2	3

$$a = x0 \quad b = x1$$

Пусть мы объединили a и b в один символ x

$$aabbcccc = xxxcccc$$

$$\sum_{a, b \rightarrow x} p_i l_i = \sum_{i \neq x} p_i l_i + p_x l_x = \sum_{i \neq x} p_i l_i + p_a(l_a - 1) + p_b(l_b - 1) = \sum_{i(a, b) \text{отдельно}} p_i l_i - p_a - p_b$$

Пример. Код Хаффмана

Теорема 13 (Неравенство Крафта-МакМиллана). $S = c_1 \dots c_k$

Можно построить однозначно декодируемый двоичный код слов l_i тогда и только тогда, когда

$$\sum_{i=1}^k s^{-l_i} \leq 1.$$

Доказательство.

$$\Leftarrow l_1 \leq l_2 \leq \dots \leq l_k$$

$$2^{-l_1} \geq 2^{-l_2} \geq \dots \geq 2^{-l_k}$$

$$2^{-l_1} + \dots + 2^{-l_{i-1}} < \frac{1}{2} \times 2^{l_i}$$

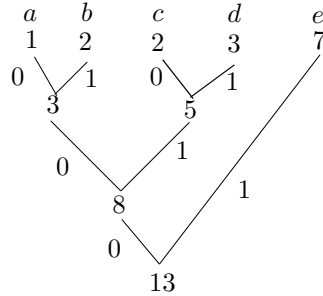


Рис. 1.4: haff

\Rightarrow Пусть есть префиксный код. Запишем в листья его дерева $2^{-d}, d$ – глубина. После этого запишем в узлах сумму детей. Тогда в корне будет число $\leq 2^0 = 1$

Теперь пусть его однозначно декодируемый код. 0 и 1 заменим на а и в, чтобы они не интерпретировались как числа.

$$\begin{aligned} c_1 &\rightarrow aba \\ c_2 &\rightarrow ab \\ &\vdots \\ c_n &\rightarrow bbb \end{aligned}$$

Сложим их $(aba + ab + aa + bbb)^k = abaaba \dots aba + abab \dots ab + \dots + bbbbbb \dots bbb$ n^k слагаемых.

$L = \max l_i$ максимальная длина слова в сумме – kL

Подставим $a = \frac{1}{2}$ $b = \frac{1}{2}$ в равенство

$$\left(\sum 2^{-li}\right)^k = \text{длины от 1 до } kL \text{ и все слова различны}$$

$$(\text{слова длины 1}) + (\text{слова длины 2}) + \dots + (\text{слова длины } kL)$$

Для i каждое слово вычислится в $\left(\frac{1}{2}\right)^i$ и они все различны, т.е. их максимум 2^i , значит скобка не превышает 1

$$\text{Значит } \left(\sum 2^{-li}\right)^k \leq kL \forall k$$

Если сумма слева > 1 , то там растущая экспонента и она обгонит линейно растущую, значит сумма ≤ 1 , что и требовалось.

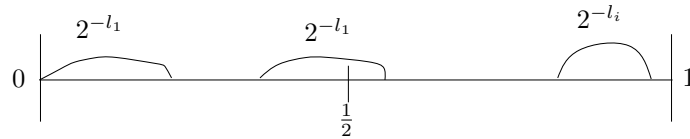


Рис. 1.5: отрезки

Замечание. Код Хаффмана – оптимальный префиксный \implies

Пусть есть код с буквами a, b и букв a очень много

0 aaaaaaaaaa

10 a

11 b

1.9 Арифметическое кодирование

c_1, c_2, \dots, c_n – символы, которые встречаются f_1, f_2, \dots, f_n раз

$\sum f_i = L$ – длина текста, который мы хотим закодировать

$$p_i = \frac{f_i}{L}$$

$$abacaba \quad p_a = \frac{4}{7} \quad p_b = \frac{2}{7} \quad p_c = \frac{1}{7}$$

Алгоритм: $l = 0, r = 1$ Делим в пропорциях p_i -ых (порядок не важен, но должен быть одинаков у кодировщика и декодировщика) Далее берём следующую букву, выделяем соответствующий отрезок, зумимся в него и повторяем операцию, рассматривая его как изначальный.

$$1. \ 2^{-q} \leq r - l$$

$$\iff -q \leq \log_2(l - r)$$

$$\Leftarrow q \geq -\log_2(l-r)$$

Но также $q \leq \lceil -\log_2(l-r) \rceil$

$$l-r = \prod_{i=1}^L p_i^{c_i} = \left(\prod_{i=1}^L p_i^{p_i} \right)^L$$

$$-\log_2(l-r) = -L \log_2 \left(\prod_{i=1}^L p_i^{p_i} \right) = -L \sum_{i=1}^n p_i \log_2 p_i$$

$$q \leq \Theta L, \Theta = \sum_{i=1}^n p_i \log_2 p_i - \text{энтропия}$$

1.9.1 RLE-кодирование

Running length encoding. *abbbbbaaaaabbbbbbb 1a4b5a5b*

1.9.2 MTF-кодирование

Move to front

abbbbbaaaaabbbbbcccccccaaa

<i>a</i>	<i>b</i>	<i>c</i>
0	1	2

$a \rightarrow 0, b \rightarrow 1$ но мы перемещаем b в начало

<i>b</i>	<i>a</i>	<i>c</i>
----------	----------	----------

$b \rightarrow 0, b \rightarrow 0, \dots, b \rightarrow 0, a \rightarrow 1, a \rightarrow 0 \dots, b \rightarrow 1, b \rightarrow 0 \dots, c \rightarrow 2, c \rightarrow 0$

01000100001000020000000020 – резкий переход по частотам в сторону 0

$\rightarrow bzip2$ – BWT Barrows Wheeler Transform :

abacaba\$ \$

\$avacaba
a\$abacab
aba\$abac
acaba\$ab
abacaba\$
ba\$abaca
bacaba\$a
caba\$aba

последний столбец – результат *abcb\$aaa*

Text \rightarrow BWT \rightarrow MTF \rightarrow AE/Haff

1.9.3 LZ77-78

abacaba

abac(4, 3) – отступи на 4 назад и повтори 3.

$ababababc \quad ab(2, 2)(4, 4)c \quad ab(2, 6)c$

$a \quad 0$
 $b \quad 1$
 $c \quad 2$

$ab \quad 3$
 $ba \quad 4$
 $ac \quad 5$
 $ca \quad 6$
 $aba \quad 7$
 $a\$ \quad 8$

010230

1.10 Избыточное кодирование

1.10.1 бит чётности

Например перед каждыми 8 битами будет стоять бит чётности и если его значение не совпадает с чётностью 8-ю битов, то будет понятно, что память повредилась.. После этого

Определение 27 (расстояние Хэмминга). Пусть мы пользуемся кодированием постоянной длины.

$H(x, y)$ – количество позиций $i : x[i] \neq y[i]$

$H(001101, 101000) = 3$

Замечание. А расстояние ли оно? Вспомним матан

1. $d(x, y) \geq 0$
2. $d(x, y) = 0 \iff x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, y) + d(y, z) \geq d(x, z)$

количество ≥ 0

ноль, только если нет различий

Определение не учитывает порядок x, y

Если входит в $d(x, z)$, то входит в $d(x, y) + d(y, z)$

Всё выполняется, и правда расстояние

Пример. $ex : \Sigma \rightarrow \mathcal{B}^n$

$\triangleleft \min_{x,y \in \Sigma} H(c(x), c(y))$ – эта величина характеризует насколько далеко находятся кодовые слова, т.е. насколько много нужно повредить, чтобы получить другой символ

Определение 28. Код обнаруживает d ошибок, если $\min_{x,y \in \Sigma} H(c(x), c(y)) = d(c) > k$

Если повредить $\leq k$ битов в c мы получим c' , но оно не может быть корректным, значит мы точно поймём, что слово повредилось.

Пример. $c : \mathbb{B}^8 \rightarrow \mathbb{B}^9$

$d(c) = 2$ Хотя бы два бита будут различаться. А значит такой код обнаруживает одну ошибку, согласно определения

Определение 29. Код c исправляет k ошибок, если $d(c) > 2k$

Внесём в слово $a \leq k$ ошибок. Получим a' . Кодировщик может восстановить символ, найдя код, отличающийся от a' не более, чем в k битах

$$H(a, a') \leq k \quad H(a, b) \leq k \quad H(a, b) \leq 2k?!!$$

Утверждение 6. Если код c исправляет k ошибок, тогда он обнаруживает $2k$ ошибок

Утверждение 7. Если код c обнаруживает k ошибок, тогда он исправляет $\lfloor \frac{k}{2} \rfloor$ ошибок

Утверждение 8. $\forall k \exists$ код, который обнаруживает k ошибок

$$\Sigma \quad |\Sigma| \leq 2^n$$

Сделаем код $c_i = i_2$ (в двоичной записи)

Повторим $k + 1$ раз каждый бит

$c : \Sigma \rightarrow \mathcal{B}^{(k+1)n}$ – он может обнаружить k ошибок

Определение 30. $\mathbb{B}^n, x \in \mathbb{B}^n, r \in \mathbb{Z}^+$

Шаром $S(x, r) = \{y | H(x, y) \leq r\}$

$$S(0000, 1) = \{0000, 0001, 0010, 0100, 1000\}$$

Определение 31. $V(n, r)$ – объём шара с радиусом r
 объём не зависит от центра – n

Замечание. $V(n, r) = |S(x, r)| \forall x \in \mathbb{B}^n$

$$S(x, r) = \{z | z = t \oplus x \oplus y, t \in S(y, r)\}$$

$$H(x, z) = |\{i | x[i] \neq z[i]\}|$$

$$y = x \oplus (x \oplus y)$$

$$t = z \oplus (x \oplus y)$$

$$H(y, t) = |\{i | y[i] \neq t[i]\}| = |x[i] \oplus (x[i] \oplus y[i]) \neq z[i] \oplus (x[i] \oplus y[i])| = \dots$$

Лемма 8. $x \neq y \in \Sigma$ c – код, исправляющий k ошибок

$$\text{То } S(c(x), k) \cap S(c(y), k) \neq \emptyset$$

$$z \in S(c(x), k) \cap S(c(y), k)$$

$$H(c(x), z) \leq k \quad H(c(y), z) \leq k \implies H(c(x), c(y)) \leq 2k$$

Теорема 14 (граница Хемминга). c – код для m -символьного алфавита, исправляющий k ошибок

$$c : \Sigma \rightarrow \mathbb{B}^n, \text{ то } m \cdot V(n, k) \leq 2^n$$

Пример. 3 символа. испр. 1 ошибку. $m = 3$?

$$3 \cdot V(3, 1) \leq 2^3$$

$$3 \cdot 4 \not\leq 2^3$$

$$\log_2 n + \log_2 V(n, k) \leq n$$

$$\frac{\log_2 m}{n} \leq 1 - \frac{\log_2 V(n, k)}{n}$$

первое – скорость передачи, которая уменьшилась на $\frac{\log_2 V(n, k)}{n}$

Доказательство. \mathbb{B}^n $x_1 = c(1)$ испр. k ошибок

$S(x_1, 2k)$ запрещены, т.к. слишком близко к x_1

$$x_2 \in \mathbb{B}^n \setminus S(x_1, 2k) \quad x_2 = c(2)$$

$$x_3 \in \mathbb{B}^n \setminus \left(\bigcup_{i=1}^2 S(x_i, 2k) \right)$$

■

Теорема 15 (граница Гильберта). Если $m \cdot V(n, 2k) \leq 2^n$, то \exists код $c : \Sigma \rightarrow \mathbb{B}^n$, исправляющий $2k$ ошибок $|\Sigma| = m$

$$V(n, r) = 1 + n + \frac{n(n-1)}{2} + \dots + C_n^r$$

$$r = 1 \quad V(n, 1) = 1 + n$$

$$m(n+2) \leq 2^n$$

$$m \leq \frac{2^n}{n+1}$$

1.10.2 Код Хемминга

Занумеруем биты от 1 до n

Биты в коде Хемминга делятся на информационные и контрольные. Контрольные – те, у которых номер позиции это степень двойки.

Пусть $|\Sigma| \leq 128$

Код постоянной длины $\rightarrow \mathbb{B}^7$

Контрольному биту соответствуют позиции $P_i = \{j | j \& 2^i = 2^i\}$. контрольный бит равен ксору всех битов его множества.

Теорема 16. Код Хемминга исправляет одну ошибку

Доказательство. $\forall x, y \quad H(x, y) \geq 3$

Различается ≥ 3 инф бита – ОК

различается 1 инф бит $\implies \geq 2$ Различных контрольных бит. Суммарно расстояние хотя бы 3.

Различаются 2 инф бита. Пусть у них номера j_1, j_2 У них есть хотя бы одна позиция, в которой они различаются ($j_1 \oplus j_2 \neq 0$), значит есть контрольный бит, учитывающий одно и не учитывающий другой. Хотя бы s -ый контрольный бит различаются \implies ■

Замечание. В коде $12 \dots 2^{s-1} \dots |2^s$

$2^s - 1 - s$ – инф. бит s контрольных

$2^s - 1 - s + s \leq 2^s - 1$ – здесь неравенство выполняется как равенство.

$$\log(m) + \log(n+1) \leq n$$