Дискретная математика

Коченюк Анатолий

2 декабря 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

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

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

A, B :

- $A \cup B = \{a \mid a \in A \text{ или } a \in B\}$
- $A \cap B = \{a \mid a \in A \text{ и } a \in B\}$
- $A \setminus B = \{a | a \in A \text{ и } a \notin B\}$
- $\overline{A} = \{a | a \notin A\}$??? 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

$A = \{a | a \text{ – множество}, a \notin a\}$

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.

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

$A^0 = \{[]\} = \{\varepsilon\}$ – пустая последовательность.

Пример. $A = 2, 3, 9$ -> $A \times A = \{(2, 2), (2, 3), (2, 9), (3, 2), (3, 3), \ldots\}$

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

$f : A \to B$

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

$a \in A$

$A, B, \forall a$ (если $a \in A$, то $a \in B$) $A \subset B$

$D_4 = \{n | n$ кратно $4\}$

$E = \{n | n$ чётно$\}$

$D_4 \subset E$

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

$A \subset A$

$\emptyset \subset A$

$A \subset U$

Замечание. Необязательно все b попадают в график.

$sqr : \mathbb{N} \to \mathbb{N}$ – 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 низ равное количество элементов.

Определение 6. Два множества называется равномощными, 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. $A = B = \mathbb{N} \quad R = \{(a, b) | a < b\} \quad R =<$

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

$A =$ люди, $B =$ собаки, $R = \{(a, b) | a$ – 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 тразитивные – 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 $\implies aRb$ a и b в разных классах $\implies \neg aRb$

множество таких классов обозначается X/R

$N/\equiv_3=$

$$\{\{1, 4, 7, 10, \ldots)$$
$$\{2, 5, 8, 11, \ldots)\}$$
$$\{3, 6, 9, 12, \ldots)\}\}$$
$$.$$

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:

$T = R \circ S \quad xTy \iff \exists z : xRz$ 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

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

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

Определение 11. Транзитивное замыкание.

$R^+ = \bigcup\limits_{k=1}^{\infty} R^k$

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

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

Замечание. Транзитивное замыкание – транзитивно

Пусть $xR^+y \implies xR^iy$

Пусть $yR^+z \implies yR^jz$

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

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

Доказательство. По индукции:

База: $R^1 \subset T$ – дано

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

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

1.2 Булевы функции

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

$\{unit\}$

void – ничего, константная функция

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

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

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

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

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

$n = 1$

Таблица 1.1: n=1

x	$\mathbb{0}$	id	$\neg$	$\mathbb{1}$
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	$\mathbb{0}$	$\wedge$	$\not\to$	P_1	$\not\leftarrow$	P_2	$\oplus$	$\vee$	$\downarrow$	$=$	$\neg P_2$	$\leftarrow$	$\neg P_1$	$\to$	$\uparrow$	$\mathbb{1}$
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: $\wedge, \vee, \neg, \oplus$

   $x_1 \oplus x_2 \oplus x_3 \ldots$

$\{f_1, f_2, .., f_n\}$ – базисные.

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

Определение 12. Дерево разбора формулы. Если у функции арность – 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 $f \neq \mathbb{0}$. 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 нормальная формой ■

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

$x \vee \neg x$ – тождественный ноль

Напоминание о способах задания функций:

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

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

$\wedge, \vee, \neg$

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

Лемма 2. $\supset F$ – некоторое множество. $\overline{F} = \mathbb{BF}$

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

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

Доказательство. $G \to F \to \forall \implies G \to \forall$ – 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 люая функция выражается через $\overline{G} \implies \overline{G} = \mathbb{BF}$ ■

Пример. $\{\oplus, \wedge, \mathbb{1}\}$

$x \wedge y = x \wedge y \qquad \neg x = x \oplus 1$ – 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 $\mathbb{0}$) 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 инъеция между равномощными множествами $\implies$ это биекция. ■

1.4 Линейный функции

Полиному Жкгалкина, в которых нету $\wedge$

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

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

Утверждение 1. Если F содержит только линейный функции, то и $\overline{F}$ содержит только линейный 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 $\mathbb{1}$

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

Утверждение 2. Если F содержит только функции, сохраняющие 0, то и $\overline{F}$ тоже

аналогично для 1

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

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

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 самодвойственная, если

$f(x_1, \dots, x_n) = \neg f(\neg x_1, \dots, \neg x_n)$

Утверждение 4. Из самодвойственных функций тоже не выйти. Тоже деревом разборп

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

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

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

$a(0) = 1$

a $a(1) = 1 \implies a(x) = 1$

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

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

1. $b(1) = 0 \implies b(x) = \mathbb{0}$
2. $b(1) = 1 \implies b(x) = \neg x$

1a $\mathbb{1}\ \mathbb{0}$

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

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

2b $\neg, x$

1a $\mathbb{1}, \mathbb{0}$ $f_m(x_1, \ldots, x_n) > f_m(y_1, \ldots, y_n)$ $x_i \leqslant y_i$ Значит первое – 1, а второе – 9

$f_m(x_1, \ldots, x_n)$
$f_m(y_1, \ldots, x_n)$
$f_m(y_1, \ldots, x_n)$
$\vdots$
$f_m(y_1, \ldots, y_n)$

В какой-то момент единица сменилась нулём на соседних строках

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

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

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

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

$c(z) = \neg z$

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

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

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

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

Итак мы получили $\mathbb{1}, \mathbb{0}, \neg$

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 $xyu_1 \dots u_k \quad k + 2 \geqslant 2$

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

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

Если есть $\oplus 1$, напишем $\neg$

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

$xy = x \wedge y$

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

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

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

1.5 Преобразование Мёбиуса

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

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

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

$s(i) = 1 \implies x(i) = 1 \iff s \& x = s \iff s \leqslant x$ (покомпонентно)

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

Таблица истинности:

0	0	0	...	$f_{00\ldots0}$
0	0	...	1	$f_{00\ldots1}$
1	1	...	1	$f_{11\ldots1}$

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

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

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

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 ($M = M^{-1}$)

$\vec{a}_t = \bigoplus\limits_{x \leqslant t} f_x$

Доказательство. $\bigoplus\limits_{x \leqslant t} f_x = \bigoplus\limits_{x \leqslant t} \bigoplus\limits_{s \leqslant x} a_s = \bigoplus\limits_{s,x:s \leqslant x \leqslant t} a_s = \bigoplus\limits_{S} [(\#x : s \leqslant x \leqslant t) \% 2] a_s = a_t$

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

■

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 Circuts)

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

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 $x_1, x_2, \ldots, x_n$. 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

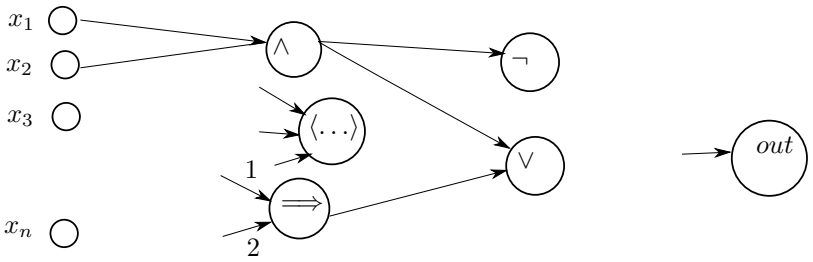


Рис. 1.1: sceme

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

Дерево разбора легко превращается в схему.

Теорема 7. Не существует формулы $len(\phi) = \widetilde{O}(n)$ 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

$\text{depth}_B(f)$ – минимальная глубина схемы для функции.

Теорема 9. B_1, B_2 – базисы.

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

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

b_i выразим через B_1

$C \leqslant \max\limits_{b_i \in B_2} size_{B_1}(b_i)$

(оптимальная схема может быть лучше, поэтому $\leqslant$) ■

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. $c_1 size_{B_2}(f) \leqslant size_{B_1}(f) \leqslant c_2 size_{B_2}(f)$

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: $x_0, \ldots, x_n, y_0, \ldots, y_n$

Побитовое И – n элементов $\wedge$ принимающие соответствующие разряды.

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

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

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

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

$size = n \quad depth = n$

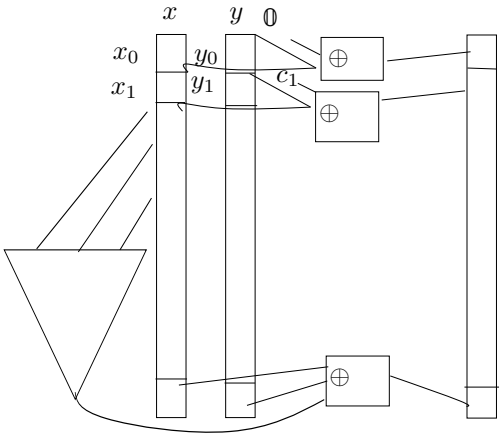


Рис. 1.2: sum

0	0	$\mathbb{0}$	k (kill)
0	1	x	p (propogate)
1	0	x	p
1	1	$\mathbb{1}$	g (generate)

1\ 2	k	g	p
k	k	g	k
g	k	g	g
p	k	g	p

$a(bc) = (ab)c = abc$ – композиция ассоциативна!

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

(Дальше жесть, 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.)

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

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

$x - y = x + (\ y) + 1$. 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	
			1	0	1	1	
		0	0	0	0		
	1	0	1	1			
1	0	1	1				

Умножать двоичные числа в стлбик 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. $x_1, x_2, \ldots, x_n$ – переменные

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

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

$$\begin{aligned} x_{n+1} &= x_2 \vee x_7 \\ x_{n+2} &= \neg x_4 \\ x_{n+3} &= x_{n+1} \wedge x_{n+2} \\ &\vdots \\ &. \end{aligned}$$

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

Пример. Сделаем $\oplus$

x_1, x_2

$$\begin{aligned} x_3 &= \neg x_1 \\ x_4 &= \neg x_2 \\ x_5 &= x_1 \wedge x_4 \\ x_6 &= x_2 \wedge x_3 \\ x_7 &= x_5 \vee x_6 \\ &. \end{aligned}$$

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.

$n^2 \cdot (n+1)^2 \cdot \ldots \cdot (n+t-1)^2 \leqslant (n+t)^{2t}$

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

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

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

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

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

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

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

Или (доля функций $\ldots$) $\to 0, n \to \infty$

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

Рассмотрим таблица, 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

Число полос $p = \frac{2^k}{s}$

$\exists$ не более чем 2^s классов эквивалентности.

Для полосы j и маски $m \quad g_{jm}$ – 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

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

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 $s = n - 2\log_2 n$

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

■

Определение 21. Алфавит Σ – любое непустое конечное множество.

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

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

α, β – два слова.

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

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

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

Свойства конкатенации:

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

Структуру с ассоциативностью и нейстральным элементом называют моноидом

Определение 23. Σ, Π – алфавиты

Обобщённым кодом ϕ называется функция

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

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

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

png – сжатие без потерь

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

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

$\varphi(c_1c_2c_3 \dots c_n) = \varphi(c_1)\varphi(c_2) \dots \varphi(c_n)$

На время будем считать $\Sigma = \Pi$

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

Доказательство. Длины 0 меньше точно незакодировать

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?

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

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

- 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.

a	0
b	00
c	11

Это не префиксный код, потому что a префикс b

a	0
b	00
c	11

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.

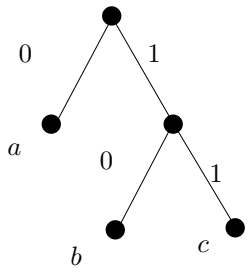


Рис. 1.3: tree

Сиволам, которые встречаются чаще, хотелось бы выдать меньший код

Задача 1. Префиксный код, $\Sigma l_i p_i \to \min$

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

Доказательство. Рассмотрим дерево, расммотрим 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 соотвествуют минимальным p_i – всё.

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

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

p_k, p_l – два самых глубоких $\implies l_i \leqslant l_k]quadl_j \leqslant l_l$

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

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

Их разность $= p_i(l_i - l_k) + p_j(l_j - l_l) - p_k(l_i - l_k) - p_l(l_j - l_l)$ ■

Пример.

a	b	c
2	2	3

$a = x0 \quad b = x1$

Пусть мы объединили a и b в один символ x

$aabbcccc = xxxxccc$

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

Пример. Код Хаффмана

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

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

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

Доказательство.

$\Longleftarrow \quad l_1 \leqslant l_2 \leqslant \dots l_k$

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

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

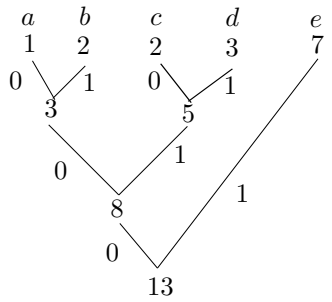


Рис. 1.4: haff

$\Longrightarrow$ 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 $\leqslant 2^0 = 1$

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

$$\begin{aligned} {}_1 &\to aba \\ c_2 &\to ab \\ \vdots &\to aa \\ c_n &\to bbb \end{aligned}$$

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

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

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

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

(слова длины 1) + (слова длины 2) + $\ldots$ + (слова длины kL)

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

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

Если сумма слева > 1, то там растущае экспонента и она обгонит линейно растущую, значит сумма $\leqslant 1$, что и требовалось.

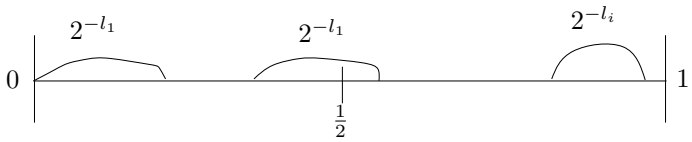


Рис. 1.5: отрезки

–

■

Замечание. Код Хаффмана – оптимальный префиксный $\Longrightarrow$

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

0 $aaaaaaaaaa$

10 a

11 b

1.9 Арифметическое кодирование

$c_1, c_2, \ldots, c_n$ – символы, которые встречаются $f_1, f_2, \ldots, f_n$ 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, расмматривая его как изначальный.

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

   $\Longleftarrow \; -q \leqslant \log_2(l - r)$

$\Longleftarrow\ q \geqslant -\log_2(l-r)$

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

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

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

$q \leqslant \Theta L, \Theta = \sum_{i=1}^{n} p_i \log_2 p_i$ – энтропия

1.9.1 RLE-кодирование

Running length encoding. $abbbbaaaaabbbbbb \quad 1a4b5a5b$

1.9.2 MTF-кодирование

Move to front

$abbbbaaaaabbbbbccccccccaa$

a	b	c
0	1	2

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

b	a	c

$b \to 0, b \to 0, \ldots, b \to 0, a \to 1, a \to 0 \ldots, b \to 1, b \to 0 \ldots, c \to 2, c \to 0$

01000100001000002000000020 – резкий переход по частотам в сторону 0

$\to bzip2$ – BWT Barrows Wheeler Transform :

$abacaba\$ \quad \$$

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

последний столбец – результат $abcb\$aaa$

Text $\to$ BWT $\to$ MTF $\to$ AE/Haff

1.9.3 LZ77-78

$abacaba$

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

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

a	0
b	1
c	2
ab	3
ba	4
ac	5
ca	6
aba	7
$a\$$	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 кодирование 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. $d(x,y) \geqslant 0$
2. $d(x,y) = 0 \iff x == y$
3. $d(x,y) = d(y,x)$
4. $d(x,y) + d(y,z) \geqslant d(x,z)$

количество $\geqslant 0$

ноль, только если нет различий

Определение не учитывает порядок x, y

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

Всё выполняется, и правда расстояние

Пример. $ex : \sum \to \mathscr{B}^n$

$\lhd \min\limits_{x,y\in\sum} H(c(x),c(y))$ – 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 $\min\limits_{x,y\in\Sigma} H(c(x),c(y)) = d(c) > k$

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

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

$d(c) = 2$ 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 $a \leqslant k$ ошибок. Получим a'. Кодировщик может восстановить символ, найдя код, отличающийся от a' не более, чем в k битах

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

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

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

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

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

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

Повторим $k+1$ раз каждый бит

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

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

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

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

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

То $S(c(x),k) \cap S(c(y),k) \neq 0$

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

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

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

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

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

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

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

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

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

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

Доказательство. $\mathbb{B}^n \quad x_1 = c(1) \quad$ 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 $m \cdot V(n, 2k) \leqslant 2^n$, то $\exists$ код $c : \Sigma \to \mathbb{B}^n$, исправляющий $2k$ ошибок $|\Sigma| = m$

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

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

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

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

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 $|\Sigma| \leqslant 128$

Код постоянной длинны $\to \mathbb{B}^7$

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

Различается $\geqslant 3$ инф бита – ОК

различается 1 инф бит $\implies \geqslant 2$ 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 $12 \ldots {}^{s-1}_{2} \ldots |\not{2}^s$

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

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

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

Глава 2

Комбинаторика

Раздел математики, изучающий комбинаторные объекты. Объекты, которые наделены какой-то внутренней структурой.

Основные задачи:

1. подсчёт и перечисление комбинаторных объектов
2. эффективная нумерация комбинаторных объектов. (мы можем хотеть эффективно их закодировать)

2.1 Вектора фиксированной длины

Последовательность из элементов какого-то алфавита Σ фиксированной длины n

$\Sigma^n \quad \mathscr{B}^n$

Выполним задачу подсчёта. Сколько у нас двоичных векторов? 2^n

Для произвольного алфавита k^n

Задача перечисления: лексикографический порядок

2.2 Лексикографический порядок

Пусть есть множество A

$X^* = \bigcup\limits_{k=0}^{\infty} X^k$ – все последовательности

пусть X линейно упорядочено, т.е. есть $\leqslant$ и любые два элемента можно сравнить

$A \subseteq X^*$

Перенесём порядок с X на A. Этот перенос называется лексикографическим порядком на A

$a = x_1 x_2 \dots x_l$

$b = y_1 y_2 \dots y_t$

$x_i, y_i \in X$

Будем говорить $a \leqslant_{lex} b$, если $\exists i \leqslant \min(l, t)$, что:

1. $i \leqslant j \implies x_i = y_j$
2. $\begin{cases} i = l \\ i < t, x_{i+1} < y_{i+1} \end{cases}$

Для $n = 3$

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

$\Sigma = \{a, b, c\} \quad n = 2$

aa
ab
ac
ba
bb
bc
ca
cb
cc

Пример. Двоичные вектора без двух единиц подряд.

000 - 0
001 - 1
010 - 2
100 - 4
101 - 5

Посчитаем их.

n		f_n
0	ε	1
1	0,1	2
>1	0.. или 10...	$F_n = 0F_{n-1} + 10F_{n-2}$ – Фибоначчи

2.3 Перестановки

(permutations)

$a_i \in \{1, 2, ..., n\} \quad i \neq j \implies a_i \neq a_j$

123
132
213
231
312
321

$\Sigma = \{\triangle, \circ, \star\}$

$\triangle \circ \star$
$\triangle \star \circ$
$\circ \triangle \star$
$\circ \star \triangle$
$\star \triangle \circ$
$\star \circ \triangle$

Подсчёт: на первую позицию можно поставить n элементов, на вторую $n-1$ и так далее получается $n(n-1)(n-2)\ldots 2 \cdot 1 = n!$

Пусть те же условия, но позиций не n, а k

$$\begin{cases} n < k & \emptyset \\ n = k & \text{перестановки} \\ n > k & \ldots \end{cases}$$

$n = 4, k = 2$: 12, 13, 14, 21, 23, 24, ...

Это так называемые размещения (arrangements)

$A_n^k = n(n-1)(n-2)\ldots(n-k+1) = n^{\underline{k}} = (n-k+1)^{\overline{k}} = \frac{n!}{(n-k)!}$

$n! = n^{\underline{n}}$

2.4 Сочетания

$\{1, 2, \ldots, n\}$

Выбираем из этих элементов k и составляем из них множество $\{a_1, a_2, \ldots, a_k\}$ $a_i \neq a_j$

$n = 3, k = 2$ $\{1, 2\}$ $\{1, 3\}$ $\{2, 3\}$

12 И 21 это разные размещения, но одно сочетание $\{1, 2\}$

Чтобы подсчитать воспользуемся канонизацией:

из 7 по 4 $\{2, 3, 5, 6\}$ $\{3, 6, 2, 5\} \ldots$ 24 способа записать одно и то же сочетание

Один из этих способов будем использовать как канонический. Путь это будет естественное возрастающее представление $\{2, 3, 5, 6\}$

n=5, k=3

123
124
125
134
135
145
234
235
245
345

Перечислили. Теперь займёмся подсчётом

$A_{n,k}$ – множество. $a \sim b$, если они задают одно и то же сочетания ($sort(a) = sort(b)$ $set(a) = set(b)$)

$C_{n,k} = A_{n,k}|_{\sim}$

У каждого класса эквивалентности получается одинаковый размер

$|C_{n,k}| = \frac{|A_{n,k}|}{k!} = \frac{n!}{k!(n-k)!} = C_n^k = \binom{n}{k}$ = биномиальный коэффициент = = n choose k

$C_{n,k} = C_{n-1,k-1} \cup C_{n-1,k}$ $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ – треугольник Паскаля

1
1 1
1 2 1
1 4 4 1
1 4 6 4 1
1 5 10 10 5 1

2.5 Код Грея

Определение 32. Код Грея на двоичных веторах:

перечисление $g_1 g_2 \dots g_n \quad H(h_i, g_{i+1}) = 1$

Пример. $n = 3$

000
001
011
010
110
111
101
100

концы тоже на расстоянии 1. Такой код называется циклическим

Теорема 17. Для любого n существует циклический код Грея $\mathscr{B}^n$

$n = 1: \quad 0\ 1$

$n \to n + 1$

Запишем два раза n один сверху вниз, другой наоборот. Затем припием к первому 0, а к другому 1

Пример. Целые числа от 1 до n кратные 3 или 5

$C = C_3 \cup C_5$

$C_3 = \left\lfloor \frac{n}{3} \right\rfloor \quad C_5 = \left\lfloor \frac{n}{5} \right\rfloor$

Некоторые считаются два раза

2.6 Формула включений-исключений

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

$C = C_3 + C_5 - C_{15}$

$|A \cup B \cup C| = |A \cup B| + |C| - |(A \cup B) \cap C| = |A| + |B| + |C| - |A \cap B| - |(A \cap C) \cup (B \cap C)| = |A| + |B| + |C| - |A \cap B| - |A \cap C| -$

Теорема 18 (Формула включений-исключений)**.** $A_1, A_2, \dots, A_n$

$\left| \bigcup_{i=1}^{n} A_i \right| = \sum_{\emptyset \neq I \subseteq \{1,2,\dots,n\}} (-1)^{|I|+1} \left| \bigcap_{i \in I} A_i \right|$

Доказательство. $\left|\bigcup\limits_{i=1}^{n} A_i\right| = \left|\bigcup\limits_{i=1}^{n-1} A_i \cup A_n\right| = \left|\bigcup\limits_{i=1}^{n-1} A_i\right| + |A_n| - \left|\bigcup\limits_{i=1}^{n-1} (A_i \cap A_n)\right| =$
$\sum_{I\subseteq\{1,2,\ldots,n-1\}} (-1)^{|I|+1} \left|\bigcap\limits_{i\in I} A_i\right| + |A_n| + \sum_{J\subseteq\{1,2,\ldots,n-1\}} (-1)^{|I|+1} \left|\bigcap\limits_{i\in J} A_i \cap A_n\right| =$
$(*)$ ■

2.7 Алгоритмы ..

Двоичные вектора фиксированной длины $\mathscr{B}^n$

Как нам сгенерировать все эти вектора?

```
gen:
    gen all, starting with 0
    gen all, starting with 1
```

Это рекурсия, мы каждый раз делим на два.

Другой взгляд: обход в глубину двоичного дерева.

```
gen(p) # p - prefix
    if len(p) == n:
        print(p)
        return
    gen(p+[0])
    gen(p+[1])

main:
    gen([])
```

Отметим, что эта программа печатает (перечисляет) вектора в лексигографическом порядке.

Перестановкаи

$n = 3123, 132, 213, 231, 312, 321$

Когда мы выбираем первый элемент, остаётся выбор только из двух, а на следующей итерации нет выбора – один возможный вариант.

```
gen(p):
    if len(p) = n:
        print(p)
        return
    for i = 1 .. n:
        if i not in p:
            gen(p+[i])
```

Строчки из 0 и 1 длинны $\leqslant n$

$\bigcup_{k=0}^{n} \mathscr{B}^n$

ε
0
00
000
001
01
010
011

Печатаем теперь не только в листе, а везде

```
gen(p) # p - prefix
    print(p)
    if len(p) = n:
        return
    gen(p+[0])
    gen(p+[1])

main:
    gen([])
```

В общем случает. Мы хотим сгенерировать объект

```
gen(p) # p - префикс комбинаторного обекта
    if (p - к.о.):
        print(p)
    for (c in Sigma) # перебор в возрастающем порядке
        if (p+[c] -- префикс к.о.)
            gen(p+[c])
```

Сочетания $\Sigma = \{1, 2, \ldots, n\}$. Характеризуются двумя числами: n, k

```
if len(p) = n:
    print(p)
    return
for (c = 1 или (p[-1]+1) .. n-k+len(p)+1)
    if p != [] and c <= p[-1]
        continue
    if n-c < k-len(p)-1
        continue # break
    gen(p+[c])
```

Правильные скобочные последовательности.

Есть последовательность открывающихся и закрывающихся скобочек. И выполнены условия:

1. На любом префиксе $\#(-\#) \geqslant 0$

2. В конце баланс – 0

() (())

((())) (()()) (()) () ()(()) () () ()

$\Sigma = \{(,)\}$ $(<)$

```
gen(p, bal):
    if len(p) = 2n:
        print(p)
        return
    if bal+1<= 2n-len(p)-1:
        gen(p+[(], bal+1)
    if bal>0:
        gen(p+[)], bal-1)

main:
gen([], 0)
```

Оптимизация: храним префикс в общем массиве a и будет брать его срезы нужной длины. Кроме того создадим массив used – маска какие мы элементы взяли. а какие ещё нет

```
gen(p) # p -- длина префикса
    if len(p) == n:
        print(a)
        return
    for i = 1 .. n:
        if !used[i]:
            used[i] = true
            a[p]=i
            gen(p+1)
            used(p[i]) = false
```

Задача 2. Хотим получить k-ый (в лексикографическом порядке) объект (нумерация с 0)

```
    gen(p)
        if p == n
            if k = 0
                print(a)
            k--
            return
        for i = 1 .. n:
            if !used[i]
                if k >= (n-p-1)!:
                    k-=(n-p-1)!
```

```
                else:
                    used[i] = true
                    a[p] = i
                    gen(p+1)
                    used[i]=false
                    return

    gen(p) # p - префикс комбинаторного обекта
        if (p - к.о.):
            if (k=0):
                print(p)
                return
            k--
        for (c in Sigma) # перебор в возрастающем порядке
            if (p+[c] -- префикс к.о.)
                t = количество к.о. с префиксом p+[c]
                if k >= t:
                    k-=t
                else:
                    gen(p+[c])
                    return
                gen(p+[c])
```

В перестановках мы можем оптимизировать ещё дальше и делать $c = \left\lfloor \frac{k}{(n-p-1)!} \right\rfloor +$ 1 $k = k\%(n-p-1)!$

```
Из n берём m
k

gen(p):
    if p == m:
        print(a)
        return
    for c = 1(a[p-1] ... n-m+p+1:
        t = Из n-c выбираем m-p-1
        if k >= t:
            k-=t
        else:
            gen(p+1)
```

Теперь будем выполнять обратную задачу: искать номер по объекту.

```
gen(p):
    if p -- к.о.:
        if p == z:
            res = k
        else:
            k++
```

```
for (c in Sigma):
    if (p+[c] -- префикс к.о):
        gen(p+[c])
```

Применим “изоморфный” алгоритм предыдущему:

```
gen(p) # p - префикс комбинаторного обекта
    if (p - к.о.):
        if p = z:
            res = k
            return
        k++
    for (c in Sigma)
        if (p+[c] -- префикс к.о.)
            t = количество к.о. с префиксом p+[c]
            if (p+[c] -- префикс z):
                gen(p+[c])
                return
            else:
                k += t
```

Следующий к.о.

к.о. лежит в упорядоченном множестве к.о. Следующий имеет с этим какой-то общий префикс

1. максимальный общий префикс, который можно сохранить, увеличив след. p
2. Увеличить следующий элемент минимальным образом, заменив a на b
3. Дальше к этому элементу $p + b$ нужно приписать минимальную возрастающую последовательность

3576421

2.8 Лекция

Перестановки $n(n-1)(n-2)\dots$

Сочетания $\frac{n(n-1)\dots(n-k+1)}{k!} = \frac{n!}{k!(n-k)!}$

Скобочные последовательности

((())) (()()) (())() ()(()) ()()()

2.8.1 Рекуррентные соотношения

Сочетания

$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ – Делим на две группы

Перестановки $P_n = P_{n-1} \cdot n$

Правильная скобочная последовательность:

1. У любого префикса неотрицательный баланс.

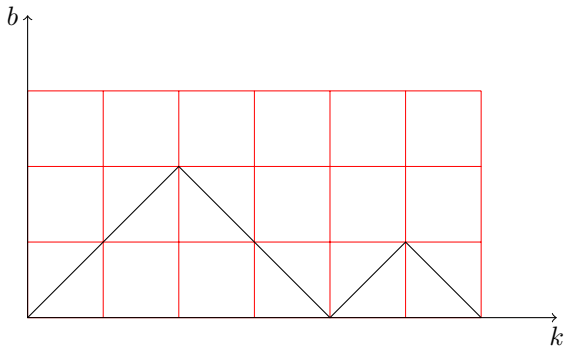
2. В конце баланс 0

Скобочная последовательность из n открывающихся скобок: длина $2n$

Если мы на месте k, у нас есть какой-то неотрицательный баланс $b \geqslant 0$

$A_{k,b}$ – количество префиксов правильных скобочных последовательностей, длины k, баланс в конце b

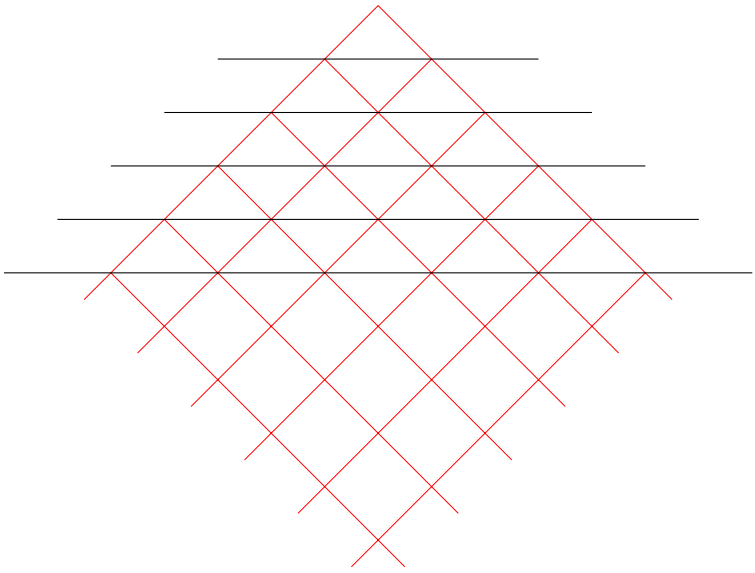
Тогда нужное нам количество $C_n = A_{2n,0}$



Так можно изобразить последовательность (())(). Такие графики называются путями Дика (Dyck Path)

$A_{k,b} = A_{k-1,b+1} + A_{k-1,b-1}$

$A_{0,0} = 1$



Сочетания тоже можно представить сеткой (рисунок выше)

$A_{k,b} = A_{k-1,b+1} + \underbrace{A_{k-1,b+1}}_{b>0}$

$A_{0,0} = 1$

Если убрать внешние скобки, то у оставшейся последовательности понизиться график Дика по балансу. Найдём индекс (первый), в котором баланс стал -1 – i. Тогда можно представить последовательность в виде (псп)псп, где псп – правильная скобочная последовательность. Эти две последовательности никак не связаны друг другом, значит их можно отдельно посчитать и перемножить $C_{i-1}C_{n-i}$

$C_n = \sum_{i=1}^{n} C_{i-1}C_{n-1} = \sum_{i=0}^{n-1} C_i C_{n-i-1}$

C_n – число Каталана: $1, 1, 2, 5, 14, 42, \ldots$

Деревья:

1. Двоичные деревья. ... Числа Каталан
2. Разбиения

```
5=1+1+1+1+1
5=5
5=4+1 (=1+4)
5=3+2
5=3+1+1
5=2+2+1
5=2+1+1+1
```

$P_{n,k}$ – количество разбиения числа n на слагаемые $\leqslant k$

$P_n = Pn, n$

$P_{0,k} = 1$

$P_{n,k} = P_{n-k,k} + P_{n,k-1}$

Можно лучше: Пентагональная теорема Эйлера

3. Разбиения с различными элементами. Числа Белла B_n

   $S_2(n,k) = \left\{ {n \atop k} \right\}$ – количество разбиений n-элементного множества на k непустых множеств, Числа Стирлинга 2 рода

   $\left\{ {n \atop k} \right\} = \left\{ {n-1 \atop k-1} \right\} + k \cdot \left\{ {n-1 \atop k} \right\}$ – один в отдельном множестве + один в множестве с друзьями.

4. Разбиение на циклы. Для перестановки можно нарисовать граф циклов. Число Стирлинга 1 рода $S_1(n,k) = \left[{n \atop k} \right]$

   $\left[{n \atop k} \right] = \left[{n-1 \atop k-1} \right] + (n-1) \left[{n-1 \atop k} \right]$

2.9 Группы действий

Размещение – список из k элементов из первых n чисел (в списке различные)

$n^{\underline{k}}$

Сочетание – облачко из k разных элементов от 1 до n

$\binom{n}{k} = \frac{n!}{k!(n-k)!}$

Введём отношение: равенство с точностью до порядка. $4317 \sim 3714$, потому что можно переставить – поменять порядок.

$A_{n,k}/_\sim = C_{n,k}$

$\{1, 2, \ldots, n\}^k$

$[1,2,3,4]$ – $4! = k!$ в классе эквивалентности

$[1,1,1,1]$ – $1! \neq k!$

Определение 33. Группоид – Множество X с бинарной операцией $\cdot : X \times X \to \quad a \cdot b \to ab$

Определение 34. Полугруппа – группоид с ассоциативностью. $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

Определение 35. Моноид – полугруппа с единицей: $\exists e : \forall a \cdot e = e \cdot a = a$

e – нейтральный элемент, нейтральное действие.

Пример. $a \to 3a$ Можем ли мы "отменить" это действие? Иногда можем

$3a \to a$ (умножаем на $\frac{1}{3}$)

$a \cdot 0 = 0$ Узнать что было до этого мы не можем. Нет обратного действия.

Определение 36. К Моноиду добавляется существование обратного элемента к любому элементу. $\forall a \exists a^{-1} \quad a \cdot a^{-1} = a^{-1} \cdot a = e$

Пример. $(\mathbb{Z}, +)$

$(\mathbb{R}, +)$

$(\mathbb{R}, \cdot)$ – НЕ группа

$(\mathbb{R} \setminus \{0\})$ – группа

$(\mathbb{Z}, \cdot)$ – НЕ группа

$(\{1, -1\}, \cdot)$ – группа

Определение 37. G – группа.

1. $g(hk) = (h)k$
2. $\exists e \quad eg = ge = g$
3. $\exists g^{-1} : \quad gg^{-1} = g^{-1}g = e$

X – объекты.

$gx = y \in X \quad G \times X \to X$

1. $h(gx) = (hg)x$
2. $ex = x$

В таком случае говорят G действует на X

Пример. $X = \mathbb{R}^2 \quad G = \mathbb{R} \setminus \{0\}, \cdot \qquad g(x, y) \mapsto (gx, gy)$

Пример. $X = \mathbb{R}^2 \quad G = [0, 2\pi), +_{mod 2\pi}$ – действие группы поворотов на множество точек

Пример. $X = \mathscr{B}^n \quad G = S_n$ – группа перестановок n элементов.

$x = 11001$ и применим к нему перестановку $p = 31245$

Получим $y = px = 10101$ (первый идёт на позицию 3, второй на позицию 1, третий на позицию 2, 4 и5 остаются на месте)

Определение 38. Группа перестановок:

ab

Хотим $(ab)x = a(bx)$

$x[i] \to b[i] \to a[b[i]$

$c = ab \quad c[i] = a[b[i]$ – это перестановка. Все $b[i]$ различны, а значит и все $a[b[i]$ различны

Даёт ли это нам группу

$a[(b[c[i]])] \quad a(bc)$

$a[b[(c[i])]] \quad (ab)c$

Ассоциативность есть.

$e_i = i \quad (ae)i = a_{e_i} = a_i$

$ea = ae = a$

$b = a^{-1}$

$ba = e \quad b[a[i] = i for i = 1 \dots n$

Замечание. Умножение перестановок – действие S_n на S_n

Пример. $X = T^n$

$G = \{0, \dots, n-1\}, +_{mod\ n}\}$

$[1, 2, 3, 4]$ свинутая 3 раза $[2, 3, 4, 1$

$X \quad G$ – группа, которая действует на X

$\sim_G$ – эквивалентны с точностью до G

$x \sim_G y \quad \exists g \in G \quad y = gx$

1. $x \sim_G x \quad x = ex$
2. $x \sim_G y \quad y = gx \quad x = g^{-1}y \quad y \sim_G x$
3. $x \sim_G y \quad y \sim_G z \quad y = gx \quad z = hy = hgx$

X/G – классы эквивалентности, орбиты

Пример. 1. $g(x,y) = (gx, gy)$. Коллинеарные вектора – орбиты

2. Повороты: Круги – орбиты

Определение 39. Неподвижные точки:

$g \quad I_g = \{x | gx = x\}$ – элементы, на которые g действует никак.

$g \neq 1 \quad I_g = \{0,0\} \quad g = 1 \quad I_g = \mathbb{R}^2$

$T^n, S_n \quad I_\pi =$ все элементы внутри цикла равны.

$n = 6$ сдвиг на 3. – три цикла.

$n = 7$ сдвиг на 3 – один большой цикл.

В общем случае для сдвигов число циклов это НОД(n,k)

Теорема 19 (Лемма Бернсайда)**.** $|X/G| = \frac{\sum_{g\in G}|I_g|}{|G|}$

Доказательство. Построим таблицу $G \times X$ и выкинем все столбцы, кроме одного для каждой орбиты.

$z = g_1 x \quad x = g_1^{-1} z$

$z = z \quad z = g_2 g_1^{-1} z$

...

$z = g_k g_1^{-1} z$

$\{h | hz = z\} = St\ z$ – действия, которые не меняют объект. Именно столько раз будет повторятся объект в столбце.

$|G| \cdot |X/G| = \sum_{x\in X} |St\ x| = \sum_{x\in X} \sum_{h\in G} [gx = x] = \sum_{g\in G} \sum_{x\in X} [gx = x] = \sum_{g\in G} |I_g|$

$$[hx = x] = \begin{cases} 1 & , hx = x \\ 0 & , \text{ иначе} \end{cases}$$

■

Пример. X – размещения, S_k – перестановки

X/S_k – сочетания

$|X/S_k| = \frac{|I_k|}{k!} = \frac{|X|}{k!}$

Пример. $n = 3 \quad \mathscr{B}^3$

123	8
132	4
213	4
231	2
312	2
321	4

$\frac{8+4+4+2+2+4}{6} = 4 \quad 000 \quad 001 \quad 011 \quad 111$

Для ожерелий $2^{\text{НОД(n,k)}}$