Равномерный код: кодовые слова всех символов исходного алфавита имеют одинаковую длину.

Неравномерный код: кодовые слова имеют разную длину (например, Код Морзе).

Закодированное сообщение можно однозначно декодировать с начала, если выполняется *условие Фано*: *никакое кодовое слово не является началом другого кодового слова*. Закодированное сообщение можно однозначно декодировать с конца, если выполняется *обратное условие Фано*: никакое кодовое слово не является окончанием другого кодового слова

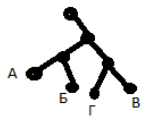
Префиксные коды: это коды со словами переменной длины, в котором ни одно кодовое слово не является началом другого кодового слова.

ПРИМЕР:

Для кодирования некоторой последовательности, состоящей из букв А, Б, В, Г и Д, решили использовать неравномерный двоичный код: А – 100 , Б – 101, В – 111, Г – 110. Укажите, каким кодовым словом из перечисленных ниже может быть закодирована буква Д. Код должен удовлетворять свойству однозначного декодирования. Если можно использовать более одного кодового слова, укажите кратчайшее из них. Варианты ответов: А) 1 Б) 0 В) 01 Г)10

РЕШЕНИЕ:

Построим дерево декодирования (0 - ветвь влево, 1 - ветвь вправо)



1 и 10 – не подходят (у этих точек есть ветви, а однозначное декодирование предполагает, что, проходя к каждой букве через узлы, мы не должны встретить другие буквы)

Подходит 0 и 01 (но в ответ пишем самое короткое 0). Ответ: Б

ФОРМУЛА ХАРТЛИ:

$$N = p^i$$

p – количество вариантов (используемых символов, сигналов), **p - МОЩНОСТЬ АЛФАВИТА**

i – количество информации (длина строки символов, длительность), **i - ДЛИНА КОДА**

N – число равновероятных событий, **N - КОЛИЧЕСТВО КОМБИНАЦИЙ**

ПРИМЕР:

Световое табло состоит из лампочек. Каждая лампочка может находиться в одном из трех состояний («включено», «выключено» или «мигает»). Какое наименьшее количество лампочек должно находится на табло, чтобы с его помощью можно было передать 18 различных сигналов?

$3^x = 18, x \approx 3$ Ответ: 3 лампочки

$$N = 2^i$$

N – мощность исходного алфавита

I – информационный вес одного символа алфавита (выражается в битах)

$$I = k \cdot i$$

k – количество символов в сообщении

i – информационный вес символа

I – информационный объем сообщения

ЕДИНИЦЫ ИЗМЕРЕНИЯ ИНФОРМАЦИИ:

- 1 байт = 2^3 (8) бит (это информационный вес символа алфавита мощностью 256)
- 1 Кбайт (Килобайт) = 2^{10} (1024) байт = 2^{13} (8192) бит
- 1 Мбайт (Мегабайт) = 2^{10} (1024) Кбайт = 2^{20} (1.048.576) байт = 2^{23}(8.388.608) бит
- 1 Гбайт (Гигабайт) = 2^{10} (1024) Мбайт = 2^{20}(1.048.576) Кбайт = 2^{30}(1.073.741.824) байт = 2^{33} (8.589.934.592) бит
- 1 Тбайт (Терабайт) = 2^{10} (1024) Гбайт = 2^{20} (1.048.576) Мбайт = 2^{30} (1.073.741.824) Кбайт = 2^{40} (1.099.511.627.776) байт = 2^{43} (8.796.093.022.208) бит
- 1 Пбайт (Петабайт) = 1024 Тбайта
- 1 Эбайт (Эксабайт) = 1024 Пбайта
- 1 Збайт (Зеттабайт) = 1024 Эбайта
- 1 Йбайт (Йоттабайт) = 1024 Збайта

Последовательность действий при переводе одних единиц измерения информации в другие приведена на схеме 1.

Схема 1

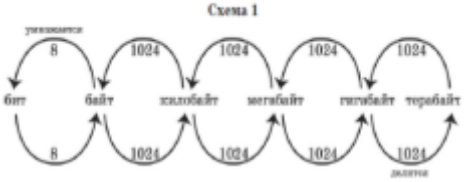


Таблица степеней двойки:

n	2^n
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192

ПРИМЕР: Для регистрации на сайте некоторой страны пользователю требуется придумать пароль. Длина пароля – ровно 11 символов. В качестве символов используются десятичные цифры и 12 различных букв местного алфавита, причём все буквы используются в двух начертаниях: как строчные, так и заглавные (регистр буквы имеет значение!). Под хранение каждого такого пароля на компьютере отводится минимально возможное и одинаковое целое количество байтов, при этом используется посимвольное кодирование и все символы кодируются одинаковым и минимально возможным количеством битов. Определите объём памяти, который занимает хранение 60 паролей. Ответы:1) 540 байт 2) 600 байт 3) 660 байт 4) 720 байт

решение см. в соседней колонке

РЕШЕНИЕ:

Мощность алфавита = 10+12+12=34 (10 цифр, 12 заглавных и 12 строчных букв).

По формуле Хартли получаем:

2^i=34, откуда i=6 бит (т.к. 2^5 = 32 <34, 2^6 = 64>34).

Тогда I=11*6=66 бит (на 1 пароль) или примерно 9 байт (на 1 пароль)

И, окончательно, т.к.60 паролей, то 60*9=540 байт

Ответ: 1.

Существуют стандартные таблицы кодов. Они могут использовать один или два байта для кодирования одного символа.

Широко используется таблица кодов, известная как **стандарт ASCII** (**American Standart Code for Information Interchange** — Американский стандартный код для обмена информацией), использующая один байт для кодирования одного символа. ASCII представляет собой кодировку для представления десятичных цифр, символов латинского и национального алфавитов, знаков препинания, символов арифметических операций и управляющих символов. Управляющие символы называют непечатаемыми символами, к ним относятся такие, как «перевод строки» (код символа 10), «возврат каретки» (код 13) и др.

Первая половина кодовой таблицы содержит **стандартные** символы ASCII (символы с кодами 0 — 127), они одинаковые во всех странах.

В системе ASCII закреплены 2 таблицы кодирования – ***базовая и расширенная***. Базовая закрепляет значения кодов от 0 до 127, а расширенная относится к символам с номерами от 128 до 255.

Первые 32 кода базовой таблицы отданы производителям аппаратных средств. Начиная с кода 32 по код 127 размещены коды символов английского алфавита, знаков препинания, цифр, арифметических действий и некоторых вспомогательных символов. Коды с номерами от 128 до 255 используются для кодирования букв национального алфавита, символов национальной валюты и т.п. Существует несколько вариантов расширенной таблицы ASCII (примеры расширений: CP-866-используется для ОС MS-DOS, CP-1251-используется для Windows, КОИ-8-используется в сети Интернет, Unicode)

ЛОГИКА

ЛОГИЧЕСКАЯ ОПЕРАЦИЯ «НЕ»

Название: логическое отрицание (инверсия):
Обозначение в булевой алгебре: ¬A $\overline{A}$
Читается: частица НЕ
Обозначение в языках программирования:
not A (Паскаль, Бейсик) !A (Си)
Логическое отрицание истинное высказывание(1) превращает в ложное (0), а ложное высказывание (0) в истинное (1).

A	$\overline{A}$
0	1
1	0

Таблица истинности логической операции инверсии
(таблица истинности – это таблица, в которой слева перечисляются всевозможные значения исходного высказывания 0 или 1, а в правой части в последнем столбце записывают результат выполнения логической операции для каждого из этих вариантов)

ЛОГИЧЕСКАЯ ОПЕРАЦИЯ «И»

Название: логическое умножение (конъюнкция):
Обозначение в булевой алгебре: A&B A^B
(схожесть с алгеброй – значок пересечения ∩)
Читается: союзы И (А, НО)
Обозначение в языках программирования:
A and B(Паскаль, Бейсик) A&&B (Си)
Логическая конъюнкция истинна тогда и только тогда, когда оба высказывания истинны (1) и ложна, когда ложно (0) хотя бы одно высказывание.

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1

Таблица истинности логической операции конъюнкция

ЛОГИЧЕСКАЯ ОПЕРАЦИЯ «ИЛИ»

Название: логическое сложение (дизъюнкция):
Обозначение в булевой алгебре: A∨B
(схожесть с алгеброй – значок объединения ∪)
Читается: союз ИЛИ
Обозначение в языках программирования:
A or B(Паскаль, Бейсик) A||B (Си)
Логическая дизъюнкция истинна тогда и только тогда, когда хотя бы одно высказывание истинно (1) и ложна, когда ложны (0) оба высказывания.

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Таблица истинности логической операции дизъюнкция

ЛОГИЧЕСКАЯ ОПЕРАЦИЯ «ИСКЛЮЧАЮЩЕЕ ИЛИ»

Название: сложение по модулю 2 (разделительная дизъюнкция)
Обозначение в булевой алгебре: A⊕B A⊻B
Читается: оборот «или только..., или только ...»
Обозначение в языках программирования:
A xor B(Паскаль) A^B (Си)
Логическое сложение по модулю 2 истинно (1) тогда и только тогда, когда высказывания не равны и ложно (0), когда высказывания одинаковы.

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Таблица истинности логической операции исключающее или

ЛОГИЧЕСКАЯ ФУНКЦИЯ «ИМПЛИКАЦИЯ»

Название: импликация (следование)
Обозначение в булевой алгебре: A→B A⊃B
Читается: если А, то В; из А следует В;
для того, чтобы А, необходимо, чтобы В;
для того, чтобы В, достаточно, чтобы А
Логическая импликация ложна (0) тогда и только тогда, когда из истинного высказывания следует ложное (0), в остальных случаях импликация истинна (1).

A	B	A→B
0	0	1
0	1	1
1	0	0
1	1	1

Таблица истинности логической функции импликация

ЛОГИЧЕСКАЯ ФУНКЦИЯ «ЭКВИВАЛЕНТНОСТЬ»

Название: эквивалентность (равносильность)
Обозначение в булевой алгебре:
A↔B A⊂⊃B A=B A ~ B
Читается: А равносильно В; А тождественно равно В; для того, чтобы А, необходимо и достаточно В; А тогда и только тогда, когда В.
Логическая эквивалентность истинна (1) тогда и только тогда, когда высказывания одинаковы, и ложна (0), когда высказывания не равны.

A	B	A↔B
0	0	1
0	1	0
1	0	0
1	1	1

Таблица истинности логической функции эквивалентность (эквиваленция, равнозначность)

ЛОГИЧЕСКАЯ ОПЕРАЦИЯ «ШТРИХ ШЕФФЕРА»

Название: И-НЕ (штрих Шеффера)
Обозначение в булевой алгебре: A|B
(англ. nand – not and – отрицание конъюнкции)
Штрих Шеффера ложен (0) тогда и только тогда, когда оба высказывания истинны, и истинен (1) в остальных случаях.

Штрих Шеффера

A	B	A|B
0	0	1
0	1	1
1	0	1
1	1	0

$A|B=\overline{A \cdot B}$

Таблица истинности логической операции Штрих Шеффера

ЛОГИЧЕСКАЯ ОПЕРАЦИЯ «СТРЕЛКА ПИРСА»

Название: ИЛИ-НЕ (стрелка Пирса)
Обозначение в булевой алгебре: A↓B или A↑B
(англ. nor – not or – отрицание дизъюнкции)
Стрелка Пирса истинна (1) тогда и только тогда, когда оба высказывания ложны, и ложна (0) в остальных случаях.

Стрелка Пирса

A	B	A↓B
0	0	1
0	1	0
1	0	0
1	1	0

$A \downarrow B=\overline{A+B}.$

Таблица истинности логической операции стрелка Пирса

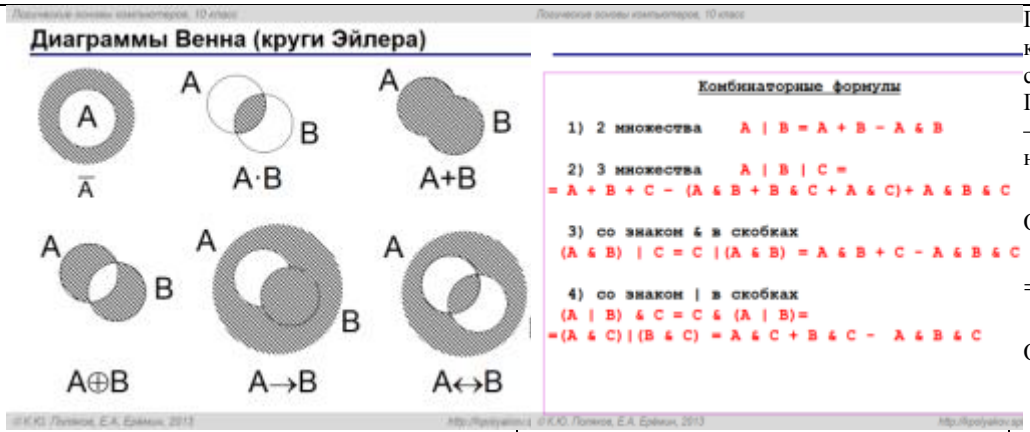
Логические переменные		Логические операции					
		отрицание	конъюнкция	дизъюнкция	исключающая дизъюнкция	импликация	эквиваленция
A	B	¬A	A&B	A∨B	A⊕B	A→B	A↔B
0	0	1	0	0	0	1	1
0	1	1	0	1	1	1	0
1	0	0	0	1	1	0	0
1	1	0	1	1	0	1	1

ПРИОРИТЕТ ВЫПОЛНЕНИЯ ЛОГИЧЕСКИХ ОПРЕЦИЙ:

- 1) В СКОБКАХ
- 2) НЕ
- 3) И
- 4) ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ
- 6) ИМПЛИКАЦИЯ
- 7) ЭКВИВАЛЕНТНОСТЬ

ЛОГИЧЕСКИЕ ЗАКОНЫ:

1) ¬¬A=A Закон двойного отрицания
2) A∨ A=A A&A=A
Законы идемпотентности
3) A ∨ 0=A A ∨ 1=1 A&0=0 A&1=A
(Законы исключения констант)
4) A ∨ (A&B) = A A& (A ∨ B) = A
(Законы поглощения)
5) A ∨ ¬A = 1 (Закон исключения третьего)
A & ¬A = 0 (Закон непротиворечия)
6) A & B = B & A A ∨ B = B ∨ A
(Законы коммутативности или переместительные законы)
7) A & (B& C) = (A & B) & C = A & B & C
A ∨(B ∨ C) = (A ∨ B) ∨ C =A ∨ B ∨ C
(Законы ассоциативности или сочетательные законы)
8) ¬ (A ∨ B) = ¬A & ¬B
¬ (A & B) = ¬A ∨ ¬B
(Законы де Моргана – названы в честь шотландского математика и логика де Моргана)
9) A & (B ∨ C) = (A & B) ∨ (A & C)
A ∨ (B & C)= (A ∨ B) & (A ∨ C)
(Законы дистрибутивности или распределительные законы)
10) A → B = ¬A ∨ B (закон снятия импликации)
11) A ↔ B = (A → B) & (B → A) =
=(A & B) ∨ (¬A & ¬B) = (A ∨¬B)&(¬A ∨B)
(закон снятия эквивалентности)
12) A ⊕ B = (A&¬B) ∨ (¬A&B)
(закон снятия сложения по модулю 2)



ПРИМЕР: Известно количество сайтов, которые находит поисковый сервер по следующим запросам: Огурцы – 100 сайтов, Помидоры – 200 сайтов, Огурцы & Помидоры – 50 сайтов. Найти, сколько сайтов будет найдено по запросу Огурцы | Помидоры?

О | П = О + П – О&П =

= 100+200-50=250

Ответ: 250.

ФОРМУЛА ДЛЯ ВЫЧИСЛЕНИЯ информационного объема звукового файла

$I = i \bullet t \bullet H \bullet k$

i – глубина кодирования (в битах)

t – время (*длительность*) звука (в секундах)

H – частота (*в Гц*)

k – режим (1 - моно, 2 - стерео, 4 - квадро)

I - информационный объем файла (в битах)

Глубина кодирования звука

- это количество информации, которое необходимо для кодирования дискретных уровней громкости цифрового звука.

Если известна глубина кодирования, то количество уровней громкости цифрового звука можно рассчитать по формуле $N = 2^i$

Пусть глубина кодирования звука составляет 16 битов, тогда количество уровней громкости звука равно: $N = 2^i = 2^{16} = 65\,536$.

ФОРМУЛА ДЛЯ ВЫЧИСЛЕНИЯ информационного объема видеофайла

Видео = (Звука + Графики) / степень сжатия

Звука = $i \bullet \omega \bullet t \bullet r$

i – глубина кодирования звука (в битах), бит

ω – частота (в ГГерцах , Гц

t – время звучания, длительность (в секундах), с

r – режим (моно - 1, стерео - 2, квадро - 4 и т.д.

Звука – информационный объем звукового файла (в битах)

Графики = $i \bullet a \bullet b \bullet v \bullet t$

i – глубина кодирования цвета (в битах), бит

a – высота картинки в пикселях

b – ширина картинки в пикселях

v – скорость кадров (количест во кадров в секунду)

t – время, длительность видеофайла (в секундах), с

Графики - информационный объем картинок

ЛИНЕЙНЫЙ АЛГОРИТМ –

- это алгоритм, в котором команды выполняются последовательно одна за другой

БЛОК-СХЕМА ЛИНЕЙНОГО АЛГОРИТМА

ЛИНЕЙНЫЙ АЛГОРИТМ НА PASCAL

```
Program <ИМЯ ПРОГРАММЫ> ;
Var <СПИСОК ПЕРЕМЕННЫХ> : <ТИП ПЕРЕМЕННЫХ> ;
Begin
<КОМАНДА ВВОДА>  (НАПРИМЕР, Readln)
<КОМАНДА ПРИСВАИВАНИЯ 1>
<КОМАНДА ПРИСВАИВАНИЯ 2>
. . . . . .
<КОМАНДА ПРИСВАИВАНИЯ n>
<КОМАНДА ВЫВОДА>  (НАПРИМЕР, Writeln)
End .
```

ВЕТВЯЩИЙСЯ АЛГОРИТМ –

- это алгоритм, в котором выполняется одна или другая серия команд в зависимости от истинности или ложности условия

ФРАГМЕНТ БЛОК-СХЕМЫ ВЕТВЯЩЕГОСЯ АЛГОРИТМА

ФРАГМЕНТ ВЕТВЯЩЕГОСЯ АЛГОРИТМА НА PASCAL

```
If <УСЛОВИЕ>  then
 begin
 <СЕРИЯ КОМАНД 1>
 end
  Else
   begin
   <СЕРИЯ КОМАНД 2>
   end;
```

АЛГОРИТМИЧЕСКАЯ СТРУКТУРА «ВЫБОР» –

- это алгоритм, в котором выполняется одна из нескольких последовательностей команд при истинности соответствующего условия (удобно использовать вместо вложенного ветвления IF)

ФРАГМЕНТ БЛОК-СХЕМЫ АЛГОРИТМА ВЫБОР

ФРАГМЕНТ АЛГОРИТМА «ВЫБОР »НА PASCAL

```
Case <ВЫРАЖЕНИЕ> Of
<ВАРИАНТ1> : <ОПЕРАТОР 1> ;
<ВАРИАНТ2> :<ОПЕРАТОР 2>  ;
. . . . . .
<ВАРИАНТ N> :<ОПЕРАТОР N-1 >;
Else  <ОПЕРАТОР  N>
End;
```

АЛГОРИТМИЧЕСКАЯ СТРУКТУРА «ЦИКЛ» –

- это алгоритм, в котором серия команд (она называется телом цикла) выполняется многократно.
- Тело цикла определяет ЧТО повторять, а заголовок цикла (вид цикла) определяет СКОЛЬКО РАЗ повторять.

ВИДЫ ЦИКЛОВ

Цикл со счетчиком (цикл n раз)

Используется, когда заранее известно, сколько раз необходимо выполнить цикл.

ФРАГМЕНТ АЛГОРИТМА ЦИКЛА n РАЗ НА АЯ (алгоритмическом языке)

```
НЦ <ПАРАМЕТР ЦИКЛА>  ОТ <НАЧАЛЬНОЕ ЗНАЧЕНИЕ> ДО <КОНЕЧНОЕ ЗНАЧЕНИЕ> ШАГ h
ПОВТОРЯТЬ
<СЕРИЯ КОМАНД – ТЕЛО ЦИКЛА >
КЦ
```

ФРАГМЕНТ АЛГОРИТМА ЦИКЛА n РАЗ НА PASCAL

```
For <СЧЕТЧИК> : = <НАЧАЛЬНОЕ ЗНАЧЕНИЕ> to (или downto) <КОНЕЧНОЕ ЗНАЧЕНИЕ> do
begin
<СЕРИЯ КОМАНД – ТЕЛО ЦИКЛА >
End;
```

Цикл с предусловием

Используется, когда заранее неизвестно, сколько раз необходимо выполнить цикл.

ФРАГМЕНТ АЛГОРИТМА ЦИКЛА с предусловием (условие впереди) НА АЯ (алгоритмическом языке)

```
НЦ ПОКА <УСЛОВИЕ>
ПОВТОРЯТЬ
<СЕРИЯ КОМАНД – ТЕЛО ЦИКЛА >
КЦ
```

Данный вид цикла может не выполниться ни разу. Ответьте, почему?

ФРАГМЕНТ АЛГОРИТМА ЦИКЛА С ПРЕДУСЛОВИЕМ НА PASCAL

ЦИКЛ С ИСТИННЫМ ПРЕДУСЛОВИЕМ

```
While <УСЛОВИЕ>  Do
Begin
<СЕРИЯ КОМАНД – ТЕЛО ЦИКЛА >
Тело цикла выполняется пока условие истинно
End;
```

Цикл с постусловием

Используется, когда заранее неизвестно, сколько раз необходимо выполнить цикл.

ФРАГМЕНТ АЛГОРИТМА ЦИКЛА с постусловием (условие после) НА АЯ (алгоритмическом языке)

```
НЦ
ПОВТОРЯТЬ
<СЕРИЯ КОМАНД – ТЕЛО ЦИКЛА >
КЦ  ПОКА <УСЛОВИЕ>
```

Данный вид цикла выполнится хотя бы один раз. Ответьте, почему?

ФРАГМЕНТ АЛГОРИТМА ЦИКЛА С ПОСТУСЛОВИЕМ НА PASCAL

ЦИКЛ С ЛОЖНЫМ ПОСТУСЛОВИЕМ

```
Repeat
<СЕРИЯ КОМАНД – ТЕЛО ЦИКЛА >
Тело цикла выполняется пока условие
Ложно
Until <УСЛОВИЕ> ;
```

Перевод чисел с использованием 10-ой системы счисления

Перевод чисел из десятичной системы счисления в систему счисления с основанием n ИЗ 10-ОЙ	Перевод чисел из системы счисления с основанием n в десятичную систему счисления В 10-УЮ
ЦЕЛЫЕ ЧИСЛА 1 СПОСОБ (ОБЫЧНОЕ ДЕЛЕНИЕ): осуществляют последовательное деление на n до тех пор, пока частное не станет = 0, остатки записывают в обратном порядке.	ЦЕЛЫЕ ЧИСЛА 1 СПОСОБ (ПО СТЕПЕНЯМ): I) Над каждой цифрой справа налево ставим целые числа в порядке возрастания 0,1,2…; II) Составляем сумму произведений цифр числа на соответствующие степени основания (показатели степени – числа из п.I)
ЦЕЛЫЕ ЧИСЛА 2 СПОСОБ (КОРОТКОЕ ДЕЛЕНИЕ) $45_{(10)} = 101101_{(2)}$	ЦЕЛЫЕ ЧИСЛА 2 СПОСОБ (СХЕМА ГОРНЕРА)

$$0,3125_{(10)} = 0,24_{(8)}$$

$$0,3125 \cdot 8 = 2,5$$

$$0,5 \cdot 8 = 4,0$$

$$0 \cdot 8 = 0$$

$$0,36_{(10)} = 0,23432\ldots_{(7)}$$

$$= 0,(2343)_{(7)}$$

$$0,36 \cdot 7 = 2,52$$

$$0,52 \cdot 7 = 3,64$$

$$0,64 \cdot 7 = 4,48$$

$$0,48 \cdot 7 = 3,36$$

$$0,36 \cdot 7 = 2,52$$

и т.д.....

$$0,28_{(16)} = 2 \bullet 16^{-1} + 8 \bullet 16^{-2} =$$

$$= \frac{1}{8} + \frac{1}{32} = \frac{5}{32} = 0,15625_{(10)}$$

Перевод чисел с использованием 10-ой системы счисления

Перевод чисел из десятичной системы счисления в систему счисления с основанием n. ИЗ 10-ОЙ	Перевод чисел из системы счисления с основанием n в десятичную систему счисления. В 10-УЮ
ДРОБНЫЕ ЧИСЛА 1 СПОСОБ (КОГДА ЦЕЛОЙ ЧАСТИ НЕТ): осуществляем последовательное умножение на n, причем очередному умножению подвергается дробная часть результата, ответ записывается в виде целых частей в порядке их появления (дробь может оказаться и периодической).	ДРОБНЫЕ ЧИСЛА 1 СПОСОБ(КОГДА ЦЕЛОЙ ЧАСТИ НЕТ) I) Над каждой цифрой слева направо ставим целые числа в порядке убывания -1,-2…; II) Составляем сумму произведений цифр числа на соответствующие степени основания (показатели оснований – из п.I)
ДРОБНЫЕ ЧИСЛА 2 СПОСОБ (КОГДА ЦЕЛАЯ ЧАСТЬ ЕСТЬ): Переводим отдельно целую часть , отдельно дробную и отделяем их в ответе запятой.	ДРОБНЫЕ ЧИСЛА 2 СПОСОБ (КОГДА ЦЕЛАЯ ЧАСТЬ ЕСТЬ) I) Над каждой цифрой ставим 0,1,2,… (влево над целой частью) и -1,-2… (вправо над дробной частью) II) Составляем сумму произведений цифр числа на соответствующие степени основания (показатели оснований – из п.I)

$$194_{(10)} = C2_{(16)}$$

194	16
12	2
0	12=C

$$0,15625_{(10)} = 0,28_{(16)}$$

$$0,15625 \cdot 16 = 2,5$$

$$0,5 \cdot 16 = 8,0$$

$$0 \cdot 16 = 0$$

ТОГДА $194,15625_{(10)} = C2,28_{(16)}$

$$C2,28_{(16)} = 12 \bullet 16^{1} + 2 \bullet 16^{0} + 2 \bullet 16^{-1} + 8 \bullet 16^{-2} =$$

$$= 192 + 2 + \frac{1}{8} + \frac{1}{32} = 194\frac{5}{32} = 194,15625_{(10)}$$

Перевод чисел без использования 10-ой системы счисления

Перевод чисел из ДВОИЧНОЙ системы счисления в ВОСЬМЕРИЧНУЮ систему счисления. 2→8	Перевод чисел из ВОСЬМЕРИЧНОЙ системы счисления в ДВОИЧНУЮ систему счисления. 8→2
ЦЕЛЫЕ ЧИСЛА 1 СПОСОБ (ТРИАДЫ-БЕЗ 10-ОЙ С.С.) Число разбивают справа налево на группы из 3 цифр (триады) и каждую триаду переводят в 8-ую систему счисления (при необходимости слева дописывают нули).	ЦЕЛЫЕ ЧИСЛА 1 СПОСОБ (ТРИАДЫ-БЕЗ 10-ОЙ С.С.) Каждую цифру числа представляют в виде триады и записывают последовательно одна за другой (при необходимости в ответе слева незначащие нули убирают).
ЦЕЛЫЕ ЧИСЛА 2 СПОСОБ (ИСПОЛЬЗУЯ ДВА ПЕРЕХОДА: СНАЧАЛА ПЕРЕВЕСТИ ИЗ 2-ОЙ В 10-УЮ, А ЗАТЕМ ИЗ 10-ОЙ В 8-УЮ)	ЦЕЛЫЕ ЧИСЛА 2 СПОСОБ (ИСПОЛЬЗУЯ ДВА ПЕРЕХОДА: СНАЧАЛА ПЕРЕВЕСТИ ИЗ 8-ОЙ В 10-УЮ, А ЗАТЕМ ИЗ 10-ОЙ В 2-УЮ)

Перевод чисел без использования 10-ой системы счисления

Перевод чисел из ДВОИЧНОЙ системы счисления в ШЕСТНАДЦАТЕРИЧНУЮ систему счисления. 2→16	Перевод чисел из ШЕСТНАДЦАТЕРИЧНОЙ системы счисления в ДВОИЧНУЮ систему счисления. 16→2
ЦЕЛЫЕ ЧИСЛА 1 СПОСОБ (ТЕТРАДЫ-БЕЗ 10-ОЙ С.С.) Число разбивают справа налево на группы из 4 цифр (тетрады) и каждую тетраду переводят в 16-ую систему счисления (при необходимости слева дописывают нули).	ЦЕЛЫЕ ЧИСЛА 1 СПОСОБ (ТЕТРАДЫ-БЕЗ 10-ОЙ С.С.) Каждую цифру числа представляют в виде тетрады и записывают последовательно одна за другой (при необходимости в ответе слева незначащие нули убирают).
ЦЕЛЫЕ ЧИСЛА 2 СПОСОБ (ИСПОЛЬЗУЯ ДВА ПЕРЕХОДА: СНАЧАЛА ПЕРЕВЕСТИ ИЗ 2-ОЙ В 10-УЮ, А ЗАТЕМ ИЗ 10-ОЙ В 16-УЮ)	ЦЕЛЫЕ ЧИСЛА 2 СПОСОБ (ИСПОЛЬЗУЯ ДВА ПЕРЕХОДА: СНАЧАЛА ПЕРЕВЕСТИ ИЗ 16-ОЙ В 10-УЮ, А ЗАТЕМ ИЗ 10-ОЙ В 2-УЮ)

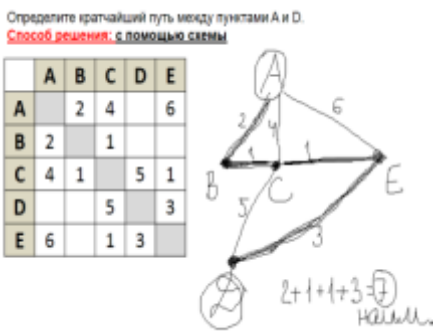
ТРИАДЫ	
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

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

Алгоритм получения дополнительного кода. Для получения дополнительного кода отрицательного числа можно использовать довольно простой алгоритм:

1. Модуль числа записать прямым кодом в n двоичных разрядах.
2. Получить обратный код числа, для этого значения всех битов инвертировать (все единицы заменить на нули и все нули заменить на единицы).
3. К полученному обратному коду прибавить единицу.

МОДЕЛИРОВАНИЕ (ПОИСК КРАТЧАЙШЕГО ПУТИ ПО ТАБЛИЦЕ)



МОДЕЛИРОВАНИЕ (ПОИСК КОЛИЧЕСТВА ПУТЕЙ НА ГРАФЕ)

МОДЕЛИРОВАНИЕ (ПОИСК КОЛИЧЕСТВА ПУТЕЙ НА ГРАФЕ)

ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ
ПРИМЕР:

Есть команды

1) +1 (обратная команда -1)
2) *2 (обратная команда :2)

Надо получить из числа 1 число 16 и записать количество команд, которые преобразуют первое число (1) во второе число (16).

Решение задачи оформляется в виде таблицы. В первой строке таблицы записываем все числа от первого (1) до последнего (16) включительно. А во второй строчке будем записывать количество команд для каждого числа. Начинаем с первого числа (под первым числом всегда пишем 1). Далее берем второе число (2) и применяем к нему все обратные команды (2-1=1, а это 1 команда, 2:2=1, а это тоже одна команда, итого получаем 1+1=2, значит под числом 2 пишем 2) и т.д. до последнего числа. Последнее число и есть ответ. В нашем случае 36.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	2	4	4	6	6	10	10	14	14	20	20	26	26	36

Основные формулы комбинаторики:

РАЗМЕЩЕНИЯ С ПОВТОРЕНИЯМИ (размещения с повторениями из m элементов по k) кортежи длины k	$\overline{A}_m^k = m^k$
РАЗМЕЩЕНИЯ БЕЗ ПОВТОРЕНИЙ (размещения без повторений из m элементов по k) кортежи длины k это упорядоченные множества	$A_m^k = \frac{m!}{(m-k)!}$
ПЕРЕСТАНОВКИ БЕЗ ПОВТОРЕНИЙ (размещения без повторений из этих элементов по m)	$P_m = m!$
СОЧЕТАНИЯ БЕЗ ПОВТОРЕНИЙ не кортежи длины k, а подмножества	$C_m^k = \frac{m!}{k!\cdot(m-k)!}$

Р-08. (http://kpolyakov.spb.ru/) Два игрока, Петя и Ваня играют в следующую игру. Задан некоторый набор символьных цепочек («слов»), в котором ни одно слово не является началом другого (выполняется условие Фано). Игра начинается с пустой строки, в конец которой игроки по очереди дописывают буквы, по одной букве за ход так, чтобы полученная цепочка на каждом шаге была началом одного из заданных слов. Первый ход делает Петя. Выигрывает тот, кто первый составит слово из заданного набора.

Пример. Пусть заданы слова {МАК, МЫЛО, РАМА, РАК}. На первом ходу Петя может написать букву М или Р. Пусть он написал букву М. В ответ Ваня может написать А или Ы. В первом случае получается МА, и Петя, дописав букву К, получает слово МАК из заданного набора и выигрывает. Во втором случае получается МЫ, Петя вынужден дописать Л и Ваня выиграет вторым ходом, дописав О и получив слово МЫЛО.

Задание 1.

а) Определите, у кого из игроков есть выигрышная стратегия для набора слов {ВАРЕНЬЕ, КОРОВА}. Опишите эту стратегию. Определите, сколько различных партий может быть сыграно при этой стратегии и какое слово будет получено в каждом случае.

б) Определите, у кого из игроков есть выигрышная стратегия для набора слов {НУБНУБ…НУБ, PUMAPUMA…PUMA}. В первом слове 55 раз повторяется слово НУБ, а во втором – 32 раза повторяется слово PUMA. Опишите эту стратегию. Определите, сколько различных партий может быть сыграно при этой стратегии и какое слово будет получено в каждом случае.

Задание 2

В наборе слов, приведённом в задании 1а, поменяйте местами две буквы в любом слове так, чтобы выигрышная стратегия была у другого игрока. Опишите эту стратегию. Определите, сколько различных партий может быть сыграно при этой стратегии и какое слово будет получено в каждом случае.

Задание 3

Дан набор слов {МОРОКА, МОРС, МОРОЗ, ПЛАХА, ПЛАТЬЕ, ПЛОМБА}. У кого из игроков есть выигрышная стратегия? Приведите в виде рисунка или таблицы дерево всех партий, возможных при этой стратегии.

Сначала предположим, что в наборе одно слово. Если игроки дописывают каждый раз по одной букве то очевидно, что первый из них (Петя) допишет все нечётные буквы, а второй (Ваня) – все чётные. Таким образом, если в слове нечётное число букв, выиграет Петя, а если чётное – Ваня.

Если слов несколько, то стратегия Пети состоит в том, чтобы все время выбирать такое продолжение, при котором в итоге будет получено слово с нечётным количеством букв, а Ваня наоборот должен пытаться перескочить на слово с чётным количеством букв.

Задание 1а. В слове ВАРЕНЬЕ – 7 букв (нечётное количество, выиграет Петя), а в слове КОРОВА – 6 букв (чётное количество, выиграет Ваня). Петя ходит первый и может написать букву В. Поскольку слово КОРОВА начинается с другой буквы, Ваня будет вынужден «идти» по слову ВАРЕНЬЕ и проиграет. Этот вариант – единственный, то есть возможна только одна партия, при

которой Петя следует своей стратегии, она заканчивается словом ВАРЕНЬЕ.

Задание 1а.

Для набора слов {ВАРЕНЬЕ, КОРОВА} выигрышная стратегия есть у Пети.

Выигрышная стратегия Пети состоит в том, чтобы написать первую букву В. Далее остается только одно допустимое слово – ВАРЕНЬЕ, и Петя выиграет, так как в этом слове 7 букв и он допишет последнюю букву, имеющую нечётный номер.

При выбранной стратегии возможна только одна партия.

В результате этой партии получится слово ВАРЕНЬЕ.

Задание 1б. В первом слове набора 3*55 = 165 букв, нечётное количество. Поэтому если игра «пойдёт» по первому слову, то выиграет Петя. Во втором слове 4*32 = 128 букв, чётное количество. Поэтому если игра пойдёт по второму слову, выиграет Ваня. Слова начинаются с разных букв, поэтому Петя может выбрать, по какому слову пойдёт игра. Если он напишет букву Н, он выиграет.

Задание 1б.

Для заданного набора слов выигрышная стратегия есть у Пети.

Выигрышная стратегия Пети состоит в том, чтобы написать первую букву Н. Далее остается только одно допустимое слово – НУБНУБ…НУБ, и Петя выиграет, так как в этом слове нечётное количество букв (165) и он допишет последнюю букву, имеющую нечётный номер.

При выбранной стратегии возможна только одна партия.

В результате этой партии получится слово НУБНУБ…НУБ.

Задание 2. В наборе слов {ВАРЕНЬЕ, КОРОВА} первое слово имеет нечётное количество букв, а второе – чётное. Чтобы Ваня мог выиграть, он должен получить возможность «перескочить» на второе слово. Для этого при любом ходе Пети у Вани должен остаться выбор. Это возможно только в том случае, когда оба слова начинаются с одной и той же буквы. Поскольку разрешается переставлять буквы только в одном слове, мы не можем сделать, чтобы оба слова начинались с буквы К – в первом слове её нет. Но можно сделать так, чтобы оба слова начинались с буквы В, переставив буквы К и В в слове КОРОВА. Получается набор {ВАРЕНЬЕ, ВОРОКА}, и Ваня выигрывает, своим первым ходом дописав букву О к букве В, которую (обязательно!) напишет Петя.

Задание 2.

Для того, чтобы Ваня мог выиграть, во втором слове нужно поменять местами буквы К и В.

Так как оба слова начинаются с буквы В, Петя обязательно напишет букву В. Выигрышная стратегия Вани состоит в том, чтобы своим ходом дописать букву О. После этого игра «идёт» по слову ВОРОКА, в нём чётное количество букв и выигрывает Ваня, который допишет последнюю букву.

При выбранной стратегии возможна только одна партия.

В результате этой партии будет получено слово ВОРОКА.

Задание 3. Расположим слова в столбики по начальным буквам, на каждом шаге вниз стараясь сохранить наибольшую общую часть (маркером выделена часть слова, которая совпадает с предыдущим словом в том же столбике):

МОРОКА	+ ПЛАХА
+ МОРОЗ	ПЛАТЬЕ
МОРС	ПЛОМБА

Знаком «плюс» отмечены слова, имеющие нечётное количество букв – это выигрышные варианты для Пети.

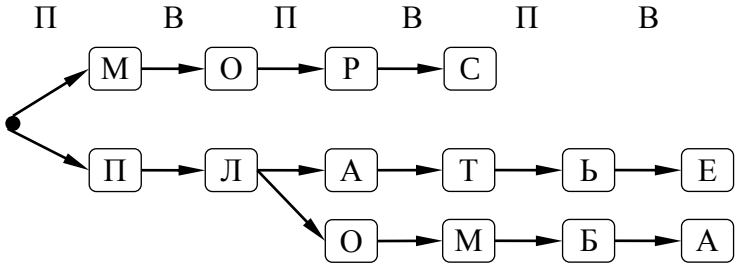
Если Петя первой напишет букву М, для выигрыша ему нужно перейти на слово МОРОЗ, но как только он составит слово МОР, Ваня тут же допишет С и выиграет, получив слово МОРС.

Если Петя напишет букву П, Ваня вынужден написать Л (это вторая буква всех оставшихся допустимых слов). Теперь Петя может написать А или О. В обоих случаях Ваня может перевести игру на слова с чётным количество букв (ПЛАТЬЕ, ПЛОМБА) и выиграть. Таким образом, при этом наборе слов выигрышную стратегию имеет Ваня.

Задание 3.

Для набора слов {МОРОКА, МОРС, МОРОЗ, ПЛАХА, ПЛАТЬЕ, ПЛОМБА} выигрышная стратегия есть у Вани.

Дерево всех возможных партий приводится на рисунке. Для Пети мы рассматриваем все возможные ходы, для Вани – только выигрышный вариант на каждом шаге. Буквами над схемой обозначены игроки (П – ход Пети, В – ход Вани).



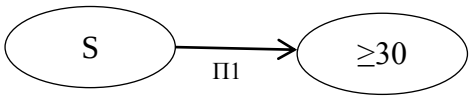
Вместо рисунка можно использовать таблицу (зелёным цветом отмечен выигрыш Вани):

Начальная позиция	П	В	П	В	П	В
_	М	МО	МОР	МОРС		
	П	ПЛ	ПЛА	ПЛАТ	ПЛАТЬ	ПЛАТЬЕ

Р-01. (Г. Сергеев, г. Москва**).** *Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу один камень, добавить в кучу три камня или увеличить количество камней в куче в два раза. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16, 18 или 30 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней. Игра завершается в тот момент, когда количество камней в куче становится не менее 30.*
В начальный момент в куче было S камней, $1 \le S \le 29$.

1. *При каких S: 1а) Петя выигрывает первым ходом; 1б) Ваня выигрывает первым ходом?*
2. *Назовите три значения S, при которых Петя может выиграть своим вторым ходом?*
3. *При каких S Ваня выигрывает своим первым или вторым ходом?*

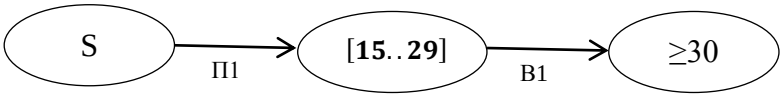
Решение (математический способ, Г. Сергеев, г. Москва**):**
Вопрос 1а. Петя выигрывает первым ходом:



Петя должен правильно выбрать один из трёх возможных вариантов действий (+1 **ИЛИ** +3 **ИЛИ** *2), которое переведет кучу камней к состоянию ≥30. Таким образом, получаем совокупность неравенств:

$$\begin{bmatrix} S+1 \ge 30 \\ S+3 \ge 30 \\ S*2 \ge 30 \end{bmatrix} \Rightarrow \begin{bmatrix} S \ge 29 \\ S \ge 27 \\ S \ge 15 \end{bmatrix} \Rightarrow S \in [15..29]$$

Вопрос 1б. Ваня выигрывает первым ходом



Любое действие Пети (**И** +1 **И** +3 **И** *2) должно привести кучу камней к состоянию $\boldsymbol{S \in [15..29]}$. Только это может обеспечить выигрыш Вани на следующем ходу. Таким образом, получаем **систему**:

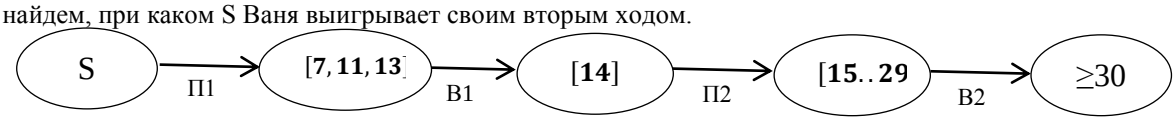
$$\begin{cases} S+1 \in [15..29] \\ S+3 \in [15..29] \\ S*2 \in [15..29] \end{cases} \Rightarrow \begin{cases} S \in [14..28] \\ S \in [12..26] \\ S \in [8..14] \end{cases} \Rightarrow S \in [14]$$

Назовите три значения S, при которых Петя может выиграть своим вторым ходом?

Петя должен выиграть, а это значит, он должен правильно выбрать один из трёх возможных вариантов действий (+1 **ИЛИ** +3 **ИЛИ** *2), которое переведет кучу камней к состоянию $\boldsymbol{S \in [14]}$. Только это может обеспечить ему выигрыш при любом действии его противника Вани. Таким образом, получаем совокупность:

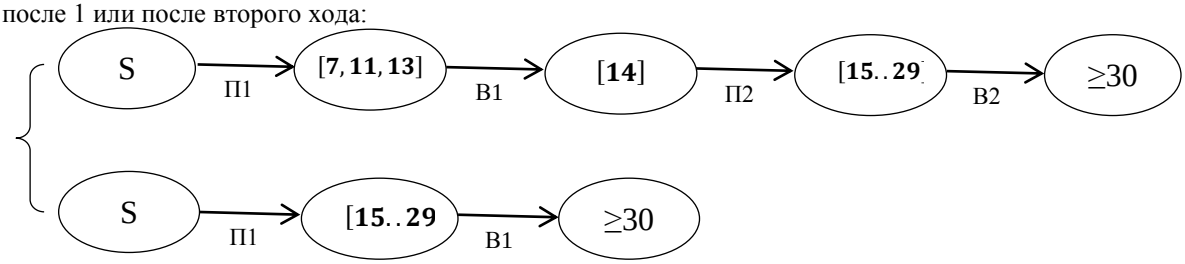
$$\begin{bmatrix} S+1 \in [14] \\ S+3 \in [14] \\ S*2 \in [14] \end{bmatrix} \Rightarrow \begin{bmatrix} S \in [13] \\ S \in [11] \\ S \in [7] \end{bmatrix} \Rightarrow S \in [7,11,13]$$

Вопрос 3. При каком S Ваня выигрывает своим первым или вторым ходом?
Сначала найдем, при каком S Ваня выигрывает своим вторым ходом.



$$\begin{cases} S+1 \in [7,11,13] \\ S+3 \in [7,11,13] \\ S*2 \in [7,11,13] \end{cases} \Rightarrow \begin{cases} S \in [6,10,12] \\ S \in [4,8,10] \\ S \in \emptyset \end{cases} \Rightarrow \boldsymbol{S \in \emptyset}$$

Таким образом, получаем, что нет такого количества камней S, которые гарантировали бы выигрыш Вани именно после его второго хода при любых действиях Пети.
Найдем, при каких значениях S любое действие Пети приведет кучу камней к такому состоянию, при котором Ваня сможет выиграть после 1 или после второго хода:



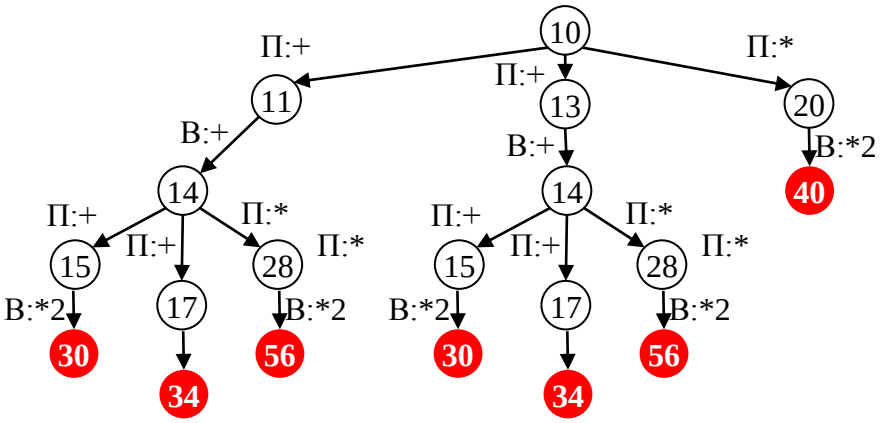
Составим систему на основе следующих условий:
любой ход Пети ведет в позицию выигрыша в два хода ($[7,11,13]$) или в один ход ($[15..29]$)

текущая позиция не совпадает с проигрышной позицией в один ход ($\underline{S \notin [14]}$), иначе, кроме нужных значений S, мы здесь получим ещё ответ на вопрос 1б

$$\begin{cases}\left[\begin{array}{l}S+1\in[7,11,13]\\S+1\in[15..29]\end{array}\right.\\\left[\begin{array}{l}S+3\in[7,11,13]\\S+3\in[15..29]\end{array}\right.\\\left[\begin{array}{l}S*2\in[7,11,13]\\S*2\in[15..29]\end{array}\right.\\S\notin[14]\end{cases}\Rightarrow\begin{cases}\left[\begin{array}{l}S\in[6,10,12]\\S\in[14..28]\end{array}\right.\\\left[\begin{array}{l}S\in[4,8,10]\\S\in[12..26]\end{array}\right.\\\left[\begin{array}{l}S\in\emptyset\\S\in[8..14]\end{array}\right.\\S\notin[14]\end{cases}\Rightarrow\begin{cases}S\in[6,10,12,14..28]\\S\in[4,8,10,12..26]\\S\in[8..14]\\S\notin[14]\end{cases}\Rightarrow\begin{cases}S\in[10,12,14]\\S\notin[14]\end{cases}\Rightarrow \boldsymbol{S\in[10,12]}$$

Итак, **ответ на вопрос 3**: S = 10 или 12.

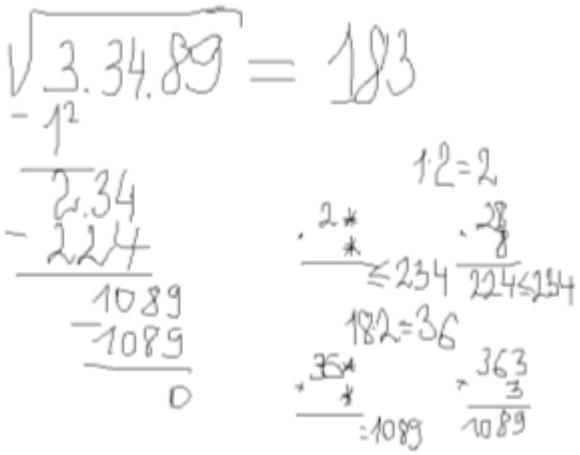
Построим дерево игры для S = 10. Обратите внимание, что после ходов Пети +1 и +3 Ваня своим следующим ходом сводит игру к одной и той же проигрышной (для Пети) позиции S = 14.



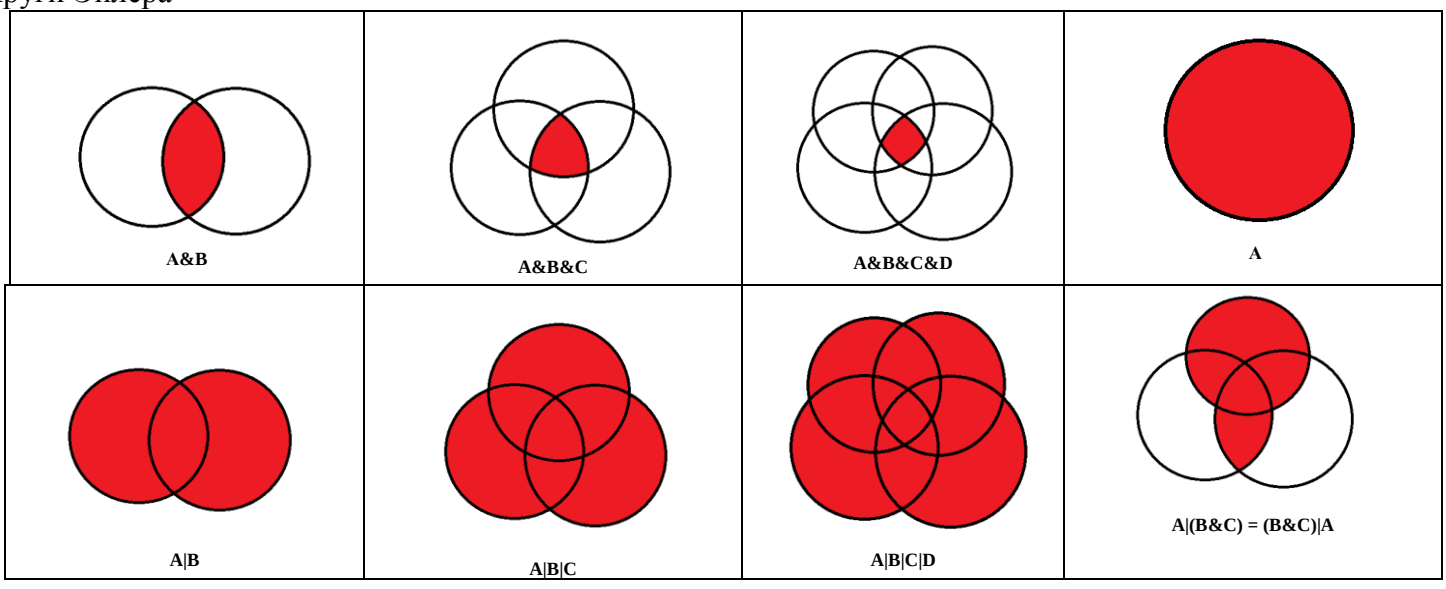
ДОПОЛНЕНИЕ: АЛГОРИТМ ИЗВЛЕЧЕНИЯ КВАДРАТНОГО КОРНЯ ИЗ НАТУРАЛЬНОГО ЧИСЛА БЕЗ ПОМОЩИ КАЛЬКУЛЯТОРА

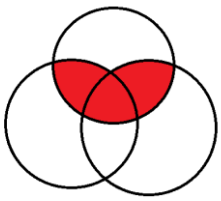
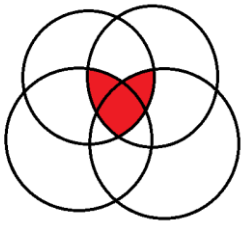
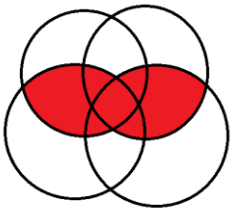
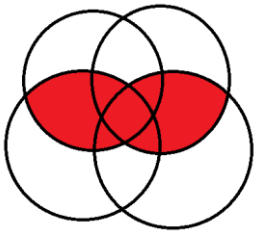
Алгоритм рассмотрим на примере конкретного числа, например, 33489

1) Разбиваем число на группы цифр (в каждой группе 2 цифры), двигаясь справа налево, т.е. 3.34.89. Кстати, сколько получилось групп цифр, столко и будет цифр в ответе.
2) Ищем наибольшее число, квалрат которого не превосходит числа, стоящего в первой группе цифр и записываем его в ответ. В нашем случае, это 1, т.к. $1^2 <= 3$.
3) Возводим это число в квадрат и вычитаем из числа, стоящего в первой группе цифр. К полученной разности приписываем вторую группу цифр (как бы «сносим»).
4) Удваиваем число, которое было записано в ответ и приписываем к полученному чису справа такую наибольшую цифру (находим ее подбором), чтобы произведение полученного числа на эту цифру не превосходило числа, полученного в пункте 3. Записываем эту цифру в ответ.
5) Из числа, полученного в пункте 3, вычитаем число, которое получилось при произведении в пункте 4, и приписываем справа следующую группу цифр (как бы «сносим»).
6) Удваиваем число, которое было записано в ответ и приписываем справа к нему снова такую наибольшую цифру (находим ее подбором), чтобы произведение полученного числа на эту цифру уже совпадало (т.к. в числе уже снесли все цифры) с числом, полученном в пункте 4. Записываем эту цифру в ответ.
7) Корень извлекли и получили ответ 183.



Круги Эйлера



 A&(B|C) = (B|C)&A	 A&B&(C|D)	 (A&B)|(C&D)	 (A|B)&(C|D)

Адрес документа в Интернете (URL = *Uniform Resource Locator*) состоит из следующих частей:

- протокол, чаще всего **`http`** (для Web-страниц) или **`ftp`** (для файловых архивов)
- знаки **`://`**, отделяющие протокол от остальной части адреса
- доменное имя (или IP-адрес) сайта
- каталог на сервере, где находится файл
- имя файла

ТАКИМ ОБРАЗОМ:

Адрес файла в сети Интернет:

протокол **`://`** доменное имя (или IP-адрес) сайта (сервер) / каталог на сервере, где находится файл / имя файла

Пример: http://www.vasya.ru/home/user/vasya/qu-qu.zip - здесь задан протокол http, доменное имя сайта (сервер) www.vasya.ru, каталог на сайте home/user/vasya и имя файла qu-qu.zip.

Каждый компьютер, подключенный к сети Интернет, должен иметь собственный адрес, который называют IP-адресом (IP = *Internet Protocol*). IP-адрес состоит из четырех чисел, разделенных точками; каждое из этих чисел находится в интервале 0…255, например: **192.168.85.210.**