

Главное из уроков по темам: единицы и формулы объёма информации, системы счисления, логика, IP-адреса, коды, теория игр и приёмы Python для заданий с программой. Обозначения такие же, как в заданиях ЕГЭ 2027.

Единицы информации

Единица	Сколько
1 байт	8 бит
1 Кбайт	1024 байт
1 Мбайт	1024 Кбайт
1 Гбайт	1024 Мбайт

- В ЕГЭ Кбайт и Мбайт связаны с байтом степенью двойки: 1 Кбайт = 2^{10} байт, 1 Мбайт = 2^{20} байт.
- Из меньшей единицы в большую дели на 1024, из большей в меньшую умножай.

Объём информации

- Алфавит из N символов: на символ нужно i бит, где $2^i \geq N$ (наименьшее такое i).
- Сообщение из K символов: $I = K \cdot i$.
- Изображение из $W \cdot H$ точек, N цветов: $I = W \cdot H \cdot i$, где $2^i \geq N$.
- Звук: $I = f \cdot i \cdot k \cdot t$: частота дискретизации f (Гц), глубина кодирования i (бит), число каналов k (моно 1, стерео 2), время t (с).
- Передача данных: $I = v \cdot t$, скорость v в бит/с.
- Пароль или номер из K символов, если запись занимает целое число байт: байт на запись = $\frac{K \cdot i}{8}$, округлённое вверх.

Степени двойки

n	2^n	n	2^n
1	2	9	512
2	4	10	1024
3	8	11	2048
4	16	12	4096
5	32	13	8192
6	64	14	16384
7	128	15	32768
8	256	16	65536

Системы счисления

- Из системы с основанием b в десятичную: каждую цифру умножить на b в степени её номера справа (с нуля) и сложить. Например, $1011_2 = 8 + 2 + 1 = 11$.
- Из десятичной в систему с основанием b : делить на b с остатком, пока не получится 0, остатки записать в обратном порядке.
- Двоичная и восьмеричная: каждой восьмеричной цифре соответствуют три двоичные. Двоичная и шестнадцатеричная: четыре двоичные.

- Шестнадцатеричные цифры: A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.
- Число b^k в системе с основанием b записывается единицей и k нулями, а число $b^k - 1$ записывается k цифрами, равными $b - 1$.
- Python: `int('1A3', 16)` переводит в десятичную, `bin(x)`, `oct(x)`, `hex(x)` в двоичную, восьмеричную и шестнадцатеричную. В любую систему: цикл `while x > 0`, цифра `x % b`, дальше `x //= b`.

Логика

Операция	Запись	Python	Истинна, когда
отрицание	$\neg A$	<code>not A</code>	A ложно
конъюнкция (И)	$A \wedge B$	<code>A and B</code>	оба истинны
дизъюнкция (ИЛИ)	$A \vee B$	<code>A or B</code>	хотя бы одно истинно
импликация	$A \rightarrow B$	<code>A <= B</code>	везде, кроме $1 \rightarrow 0$
тождество	$A \equiv B$	<code>A == B</code>	значения совпадают

- В заданиях конъюнкцию пишут ещё как $\wedge$ или `&`, дизъюнкцию как $\vee$ или вертикальную черту.
- Порядок действий без скобок: отрицание, конъюнкция, дизъюнкция, импликация, тождество.
- $A \rightarrow B$ равносильно $\neg A \vee B$.
- Законы де Моргана: $\neg(A \wedge B) = \neg A \vee \neg B$, $\neg(A \vee B) = \neg A \wedge \neg B$.
- Все наборы значений в Python: `for x, y, z in itertools.product([0, 1], repeat=3):`.

IP-адреса и маски

- IP-адрес: четыре байта, записываются числами от 0 до 255 через точку.
- Маска в двоичной записи: сначала только единицы, потом только нули.
- Адрес сети: поразрядная конъюнкция адреса узла и маски. В Python для чисел `ip & mask`.
- Если в маске k нулей, в сети 2^k адресов. Адрес из одних нулей в части узла обозначает саму сеть, из одних единиц служит широковещательным, поэтому узлам остаётся $2^k - 2$.
- Модуль `ipaddress`:
`ipaddress.ip_network('192.168.32.0/255.255.224.0')` перебирает адреса сети.

Коды и условие Фано

- Условие Фано: ни одно кодовое слово не является началом другого кодового слова. Тогда сообщение раскодируется однозначно.
- Коды удобно строить двоичным деревом: слова стоят в листьях, у каждого узла не больше двух ветвей.

- Чтобы найти самое короткое свободное слово, ищи в дереве незанятую ветку поближе к корню.

Теория игр: № 19–21

- Позиция выигрышная, если из неё есть ход в проигрышную позицию для соперника.
- Позиция проигрышная, если любой ход из неё ведёт в выигрышную позицию для соперника.
- Программой: рекурсивная функция «может ли ходящий выиграть не больше чем за m ходов» и перебор всех S из условия.

Python: частые приёмы

- Чтение чисел из файла: `nums = list(map(int, open('17.txt').read().split()))`.

- Делители n : перебор d до $\sqrt{n}$, каждый делитель d даёт пару d и $n // d$.
- Простое число: больше 1 и не делится ни на одно число от 2 до $\sqrt{n}$.
- Все пары соседних элементов: `for i in range(len(a) - 1):`, все пары вообще: два вложенных цикла, j от $i + 1$.
- Перебор слов: `itertools.product('ABB', repeat=5)`, перестановки: `itertools.permutations(s)`.
- Рекурсия с запоминанием: `from functools import lru_cache` и строка `@lru_cache(None)` над функцией. Глубину рекурсии поднимает `sys.setrecursionlimit(10000)`.
- Целочисленное деление и остаток: `//` и `%`, сразу оба: `divmod(a, b)`.