**Арифметические алгоритмы**

**Количество делителей**

Подсчитайте количество натуральных делителей числа x (включая 1 и само число)

**1 вариант (за O(*n*))**

В самом простом случае переберем все числа от 1 до *n* включительно и проверим на какие из них делится число *n*.

x = **int**(**input**())

k = 0

**for** i **in range** (1, x + 1):

 **if** x % i == 0:

 k = 1

**print**(k)

**2 вариант (за O(**$\sqrt{n}$**))**

Делители обычно можно искать парами. Пусть есть число 8. Оно делится на 1 ина 8, на 2 и на 4.

Аналогично число 12 делится на 1 и 12, 2 и 6, 2 и 3.

Исключение составляют полные квадраты. Там у квадрата нет пары.

Например число 16 делится на 1 и 16, 2 и 8 и 4.

Достаточно перебирать делители до $\sqrt{n}$ и добавлять к количеству 2. В паре обязательно одно число больше $\sqrt{n}$, а второе число меньше корня из $\sqrt{n}$. Если число является полным квадратом, то у квадрата нет пары и нужно убрать 1.

x = **int**(**input**())

k = 0

**for** i **in range** (1, **int**(x \*\* 0.5) + 1):

 **if** x % i ==0:

 k += 2

**if** **int**(x \*\* 0.5) \*\* 2 == x:

 k -= 1

**print**(k)

**Проверка на простоту**

Проверьте, является ли число простым.

**Простым** называется число, которое имеет ровно два делителя.

Можно посчитать количество делителей, как это сделано выше и сравнить найденное количество с 2.

Но можно искать минимальный делитель числа, который будет меньше *n*. Т.к. делители входят в число парами (кроме квадрата числа), то в паре одно число будет меньше $\sqrt{n}$, а второе больше. Достаточно проверить все числа начиная с 2, которые не превосходят $\sqrt{n}$. Если среди них не будут найдены делители, то число простое.

n = **int**(**input**())

is\_prime = **True**

d = 2

**while** d \*\* 2 <= n:

 **if** n % d == 0:

 is\_prime = **False**

 **break**

 d += 1

**if** is\_prime:

 **print**("YES")

**else**:

 **print**("NO")

**Разложение числа на простые множители**

Найти все простые делители числа с учетом кратности.

d = 2

**while** d \* d <= n:

**if** n % d == 0:

 **print**(d);

 n //= d

**else:**

 d += 1

**print**(n)

**НОД (алгоритм Евклида)**

Даны два целых числа *a*, *b*. Найти наибольший общий делитель этих чисел.

Если числа являются натуральными, то один из вариантов написания этого алгоритма будет такой:

**while** a != b:

 **if** a > b:

 a -= b

 **else:**

 b -= a

**print**(a)

Если $a\vdots d$ и $b\vdots d, то $ $(a-b)\vdots d$ и $(b-a)\vdots d$. Значит, НОД в ходе работы алгоритма не меняется.

Этот алгоритм работает достаточно медленно. Этот алгоритм будет работать за *O*(max(*a*, *b*)). Также эта реализация не работает для отрицательных чисел.

Вычитая много раз из большего числа меньшее, будет найден остаток от деления числа *a* на число *b*.

Красивая реализация алгоритма Евклида устроена на следующей идеи

НОД(*a*, *b*) = НОД(*b*, *a* % *b*)

**while** b !=0:

 a, b = b, a% b;

**print**(a)