

Долгое вычитание

В первой подзадаче достаточно просто реализовать предложенную последовательность действий.

```
n = int(input())
ans = 0
while n:
    s = str(n)
    if '1' in s or '3' in s or '5' in s or '7' in s or '9' in s:
        n -= 1
    else:
        n //= 2
    ans += 1
print(ans)
```

Для решения задачи на полный балл заметим, что количество операций деления не превосходит $\log_2 n \leq 60$, а вот количество выполненных подряд вычитаний единицы может быть очень большим. Например, при $n = 777$ вычитать придётся до числа 688 (почти сто раз).

Попробуем определить, до какого числа нам придётся вычитать единицу, пока не произойдёт следующая операция деления. Очевидно, что первая встретившаяся в числе нечётная цифра d должна стать чётной: ближайшей её четной цифрой, меньшей d , является $d - 1$. На последующих позициях должны стоять наибольшие чётные цифры, то есть восьмёрки.

Было: <любые чётные цифры> <нечётная цифра d > <любые цифры>

Стало: <те же чётные цифры> <чётная цифра $d - 1$ > <8...8>

Количество пропущенных операций вычитания единицы равно разности между старым и новым числом. В предложенном ниже решении функция `first_odd` возвращает индекс первой нечётной цифры, и тогда следующее число строится по описанному алгоритму. Если же нечётных цифр нет, то функция возвращает -1 и тогда выполняется операция деления на 2.

```
n = input()

def first_odd(n):
    for i in range(len(n)):
        if int(n[i]) % 2 == 1:
            return i
    return -1

ans = 0
while int(n) > 0:
    i = first_odd(n)
    if i == -1:
        n = str(int(n) // 2)
        ans += 1
    else:
        new_n = n[:i] + str(int(n[i]) - 1) + "8" * (len(n) - i - 1)
        ans += int(n) - int(new_n)
        n = new_n
print(ans)
```