Пусть имеется рекуррентное соотношения:

$$T(n) = \begin{cases} a\,T\left(\frac{n}{b}\right) + O(n^c), & n > 1 \\ O(1), & n = 1 \end{cases}$$

где $a \in \mathbb{N}, b \in \mathbb{R}, b > 1, c \in R^+$.

Тогда асимптотическое решение имеет вид:

3. Если $c > \log_b a$, то $\boxed{T(n) = O(n^c)}$
2. Если $c = \log_b a$, то $T(n) = O(n^c \log n)$
1. Если $c < \log_b a$, то $T(n) = O\left(n^{\log_b a}\right)$

Найти асимптотическую оценку функции $T(n)$, воспользовавшись основной теоремой о рекурсии:

(а) $T(n) = 3T\left(\frac{n}{3}\right) + dn$;

(б) $T(n) = 8T\left(\frac{n}{2}\right) + dn^2$;

(в) $T(n) = 8T\left(\frac{n}{2}\right) + dn^4$.

а) $a = 3$ $b = 3$ $c = 1 \Rightarrow \log_3 3 = 1 = c$

$\boxed{T(n) = O(n \log n)}$

б) $a = 8$ $b = 2$ $\log_b a = \log_2 8 = 3 > c$

$\boxed{T(n) = O(n^3)}$

в) $a = 8$, $b = 2$, $c = 4$ $\log_2 8 = 3 < 4$

$\boxed{T(n) = O(n^4)}$

Определим $f(n)$ как количество выводов строки $HW!$, осуществляемых при выполнении функции HW(n). Оцените асимптотику роста функции $f(n)$.

```
Function HW(n):
    if n < 2021 then
        for i = 1 to n do
            print("HW!");
        end
    else
        HW(⌊n/3⌋);   (1)
        print("HW!");   + (1)
        HW(⌊n/3⌋);   (2)
        for i = 1 to 2021 do
            print("HW!");   (2021)
        end
    end
end
```

$f(n) = n$, $n < 2021$

$f(n) = 2 f\left(\frac{n}{3}\right) + O(n^0)$ — a, b, c

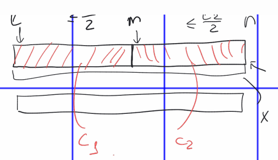
$\log_b a = \log_3 2 > 0$

$\boxed{f(n) = O\left(n^{\log_3 2}\right)}$

| 0 0 | 1 2 |

Пусть $A[1 \dots n]$ — массив. Элемент массива $A[1 \dots n]$ называется *majority element*, если он встречается в массиве больше чем $n/2$ раз. Постройте алгоритм (*разделяй и властвуй*), который находит majority element в массиве за $O(n \log n)$, если он есть. Операции сравнения элементов как чисел запрещены (представьте, что вы имеете дело с массивом картинок); вы можете только проверять условия вида $A[i] = A[j]$.

ответ



x maj

```
def f(l, r):
    if l == r:
        return A[l]
    m = ⌊(l+r)/2⌋
    c1 = f(l, m)
    c2 = f(m+1, r)
    count[c1] = 0
    count[c2] = 0
    for i = l ... r:
        if A[i] == c1:
            count[c1] += 1
        if A[i] == c2:
            count[c2] += 1
    if count[c1] > (r-l+1)/2:
        return c1
    if count[c2] > (r-l+1)/2:
        return c2
    return None
```

$O(r - l + 1)$

00 | 1 2

$c_1 = 0$ $c_2 = None$

$count[c_1] = 2$

$count[c_2] = 0$

$2 > \frac{4}{2}$

00 | 0 2

$c_1 = 0$ $c_2 = None$

$count[c_1] = 3$

$3 > \frac{4}{2}$

return 0

$a = 2$

$b = 2$

$c = 1$

$\log_b a = 1 = c$

$\boxed{T(n) = n \log n}$

freq[A[i]] += 1

Пусть $A[1 \dots n]$ — массив. Элемент массива $A[1 \dots n]$ называется *majority element*, если он встречается в массиве больше чем $n/2$ раз. Постройте алгоритм (*разделяй и властвуй*), который находит majority element в массиве за $O(n \log n)$, если он есть. Операции сравнения элементов как чисел запрещены (представьте, что вы имеете дело с массивом картинок); вы можете только проверять условия вида $A[i] = A[j]$.

Хочу: $O(n)$

Пусть

$\boxed{1 \le A[i] \le n}$

3 2 0 0 1 1 2 2 2 2 2 3 3 3 3 3

freq[i] – частота i в A

freq[0] = 2

freq[1] = 2

f

→ freq = [2 2 6 6 0 0 0 ..]

max(freq) > $\frac{n}{2}$?

cand => A = [1 2 2 3 3 3]

cand: 1 1 2 2 3 3

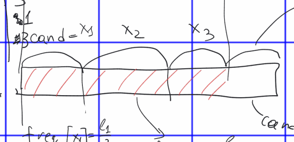
c: 1 0 1 0 1 2

A: 1 1 4 2 3

c: 0 1 2 1 0 1

cand: 1 1 1 1 3

```
def find_cand(A[]):
    c = 1
    cand = A[1]
    for i = 2 .. n:
        if c == 0:
            cand = A[i]
            c = 1
        else:
            if A[i] == cand:
                c += 1
            else:
                c -= 1
    if c > 0:
        return cand
```



$freq[x] \le \frac{l_i}{2}$

$freq[x] > \frac{l_4}{2}$

посл. кусочек

$freq[x_1] = \frac{l_1}{2}$

$freq[x] \le \frac{l_2}{2}$

$c > 0$

$freq[x] \le \frac{l_1}{2}$ if $x_1 \ne x$

$freq[x] = \frac{l_1}{2} \le \frac{l_1}{2}$ $x_1 = x$

Пусть x – maj в A

Тогда x = find_cand(A)

~~1 1 2 3 3 3~~

1 2 1 2 3 3 3

find_cand(A) = 3

Дан двумерный массив целых чисел `squares`. Каждое `squares[i]` $= [x_i, y_i, l_i]$ задаёт координаты нижнего левого угла и длину стороны квадрата, стороны которого параллельны осям.

Найдите минимальное значение координаты y горизонтальной прямой такое, что суммарная площадь частей квадратов, находящихся выше этой прямой, равна суммарной площади частей квадратов, находящихся ниже неё.

Ответы в пределах 10^{-5} от правильного считаются верными.

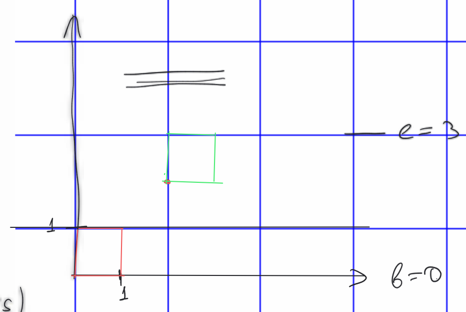
Замечание: квадраты могут перекрываться. Перекрывающиеся области учитываются несколько раз.

Пример 1:

- Ввод: `squares = [[0,0,1],[2,2,1]]`
- Вывод: `1.00000` x y l
- Пояснение: Любая горизонтальная прямая между $y = 1$ и $y = 2$ оставляет по 1 единице площади выше и ниже; наименьшее значение — 1.

Пример 2:

- Ввод: `squares = [[0,0,2],[1,1,1]]`
- Вывод: `1.16667`



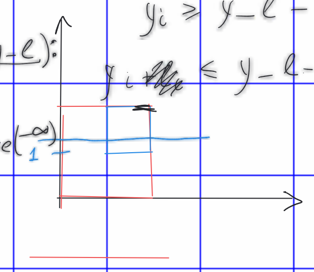
n – len(squares)

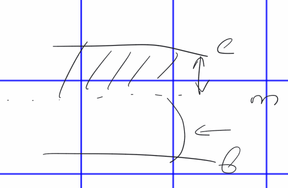
$y_i \ge y\_l$ – выше.

$y_i \le y\_l - l_i$ – ниже

```
def find_above(y_l):
    //exercise

total = find_above(-∞)
target = total / 2
```






```
b, e = min(y_i), max(y_i + l_i)

while e - b > 10^-5:
    m = (b+e)/2
    above = find_above(m)
    if above > target:
        b = m
    else:
        e = m

return (e+b)/2
```