

## Задача А. Хип ли?

Имя входного файла: `isheap.in`  
Имя выходного файла: `isheap.out`

Структуру данных Heap можно реализовать на основе массива.

Для этого должно выполняться *основное свойство Heap'a*, которое заключается в следующем. Для каждого  $0 \leq i < n$  выполняются следующие условия:

- Если  $2i + 1 < n$ , то  $a[i] \leq a[2i + 1]$
- Если  $2i + 2 < n$ , то  $a[i] \leq a[2i + 2]$

Дан массив целых чисел. Определите является ли он Heap'ом.

### Формат входного файла

Первая строка входного файла содержит целое число  $n$  ( $1 \leq n \leq 10^5$ ). Вторая строка содержит  $n$  целых чисел по модулю не превосходящих  $2 \cdot 10^9$ .

### Формат выходного файла

Выведите «YES», если массив является Heap'ом и «NO» в противном случае.

### Пример

<code>isheap.in</code>	<code>isheap.out</code>
5 1 0 1 2 0	NO
5 1 3 2 5 4	YES

## Задача В. Приоритетная очередь

Имя входного файла: `priorityqueue.in`  
Имя выходного файла: `priorityqueue.out`

Реализуйте приоритетную очередь. Ваша очередь должна поддерживать следующие операции: добавить элемент, извлечь минимальный элемент, уменьшить элемент, добавленный во время одной из операций.

Все операции нумеруются по порядку, начиная с 1.

### Формат входного файла

Входной файл содержит описание операций со очередью. В очередь помещаются и извлекаются только целые числа, не превышающие по модулю  $10^9$ .

### Формат выходного файла

Выведите последовательно результат выполнения всех операций `extract-min`. Если перед очередной операцией `extract-min` очередь пуста, выведите вместо числа звездочку.

### Пример

<code>priorityqueue.in</code>	<code>priorityqueue.out</code>
<code>push 3</code>	<code>2</code>
<code>push 4</code>	<code>1</code>
<code>push 2</code>	<code>3</code>
<code>extract-min</code>	<code>*</code>
<code>decrease-key 2 1</code>	
<code>extract-min</code>	
<code>extract-min</code>	
<code>extract-min</code>	

## Задача С. Сортировка

Имя входного файла: `sort.in`  
Имя выходного файла: `sort.out`

Дан массив целых чисел. Ваша задача — отсортировать его в порядке неубывания.

Для того, чтобы узнать, какую сортировку вам нужно реализовать, возьмите остаток от деления вашего номера на 3.

- 0 — Сортировка слиянием.
- 1 — Сортировка кучей.
- 2 — Быстрая сортировка.

### Формат входного файла

В первой строке входного файла содержится число  $n$  ( $1 \leq n \leq 100000$ ) — количество элементов в массиве. Во второй строке находятся  $n$  целых чисел, по модулю не превосходящих  $10^9$ .

### Формат выходного файла

В выходной файл надо вывести этот же массив в порядке неубывания, между любыми двумя числами должен стоять ровно один пробел.

### Пример

sort.in	sort.out
10 1 8 2 1 4 7 3 2 3 6	1 1 2 2 3 3 4 6 7 8

## Задача D. К-ая порядковая статистика

Имя входного файла: `kth.in`  
Имя выходного файла: `kth.out`

Дан массив из  $n$  элементов. Какое число  $k$ -ое в порядке возрастания в этом массиве.

### Формат входного файла

В первую строке входного файла содержится два числа  $n$  — размер массива и  $k$ . ( $1 \leq k \leq n \leq 3 \cdot 10^7$ ). Во второй строке находятся числа  $A$ ,  $B$ ,  $C$ ,  $a_1$ ,  $a_2$  по модулю не превосходящие  $10^9$ . Вы должны получить элементы массива начиная с третьего по формуле:  $a_i = A * a_{i-2} + B * a_{i-1} + C$ . Все вычисления должны производиться в 32 битном знаковом типе, переполнения должны игнорироваться.

### Формат выходного файла

Выведите значение  $k$ -ое в порядке возрастания число в массиве  $a$ .

### Пример

<code>kth.in</code>	<code>kth.out</code>
5 3 2 3 5 1 2	13
5 3 200000 300000 5 1 2	2

Во втором примере элементы массива  $a$  равны: (1, 2, 800005, -516268571, 1331571109).

## Задача Е. Цифровая сортировка

Имя входного файла: `radixsort.in`  
Имя выходного файла: `radixsort.out`

Дано  $n$  строк, выведите их порядок после  $k$  фаз цифровой сортировки.

### Формат входного файла

В первой строке входного файла содержится число  $n$  — количество строк,  $m$  — их длина и  $k$  — число фаз цифровой сортировки ( $1 \leq n \leq 1000$ ,  $1 \leq k \leq m \leq 1000$ ). В следующих  $n$  строках находятся сами строки.

### Формат выходного файла

Выведите строки в порядке в котором они будут после  $k$  фаз цифровой сортировки.

### Пример

radixsort.in	radixsort.out
3 3 1 bbb aba baa	aba baa bbb
3 3 2 bbb aba baa	baa aba bbb
3 3 3 bbb aba baa	aba baa bbb

## Задача F. Анти-QuickSort

Имя входного файла:        `antiqs.in`  
Имя выходного файла:      `antiqs.out`

Для сортировки последовательности чисел широко используется быстрая сортировка — QuickSort. Далее приведена программа, которая сортирует массив `a`, используя этот алгоритм.

```
def qsort(left , right):  
    key = a[(left + right) // 2]  
    i = left  
    j = right  
    while i <= j:  
        while a[i] < key:      # первый while  
            i += 1  
        while a[j] > key:      # второй while  
            j -= 1  
        if i <= j:  
            a[i], a[j] = a[j], a[i]  
            i += 1  
            j -= 1  
    if left < j:  
        qsort(left , j)  
    if i < right:  
        qsort(i , right)
```

```
qsort(0, n - 1)
```

Хотя QuickSort является самой быстрой сортировкой в среднем, существуют тесты, на которых она работает очень долго. Оценивать время работы алгоритма будем количеством сравнений с элементами массива (то есть суммарным количеством сравнений в первом и втором `while`). Требуется написать программу, генерирующую тест, на котором быстрая сортировка сделает наибольшее число таких сравнений.

### Формат входного файла

В первой строке находится единственное число  $n$  ( $1 \leq n \leq 70000$ ).

### Формат выходного файла

Вывести перестановку чисел от 1 до  $n$ , на которой быстрая сортировка выполнит максимальное число сравнений. Если таких перестановок несколько, вывести любую из них.

### Пример

<code>antiqs.in</code>	<code>antiqs.out</code>
3	1 3 2