

Использовать стандартную библиотеку (`priorityqueue`, `set`, `TreeSet`, и т. п.) или структуры данных, отличные от СНМ и приоритетных очередей (например, дерево отрезков или двоичное дерево поиска), не разрешается.

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 $1 \leq i \leq n$ выполняются следующие условия:

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

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. Система непересекающихся множеств

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

Реализуйте систему непересекающихся множеств. Вместе с каждым множеством храните минимальный, максимальный элемент в этом множестве и их количество.

Формат входного файла

Первая строка входного файла содержит n — количество элементов в носителе ($1 \leq n \leq 100000$). Далее операций с множеством. Операция `get` должна возвращать минимальный, максимальный элемент в соответствующем множестве, а также их количество.

Формат выходного файла

Выведите последовательно результат выполнения всех операций `get`.

Пример

<code>dsu.in</code>	<code>dsu.out</code>
5	3 3 1
union 1 2	1 2 2
get 3	1 3 3
get 2	5 5 1
union 2 3	4 5 2
get 2	1 5 5
union 1 3	
get 5	
union 4 5	
get 5	
union 4 1	
get 5	

Задача С. 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 D. Парковка

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

На кольцевой парковке есть n мест пронумерованных от 1 до n . Всего на парковку приезжает n машин в порядке нумерации. У i -й машины известно место p_i , которое она хочет занять. Если машина приезжает на парковку, а её место занято, то она едет далее по кругу и встаёт на первое свободное место.

Формат входного файла

В первой строке входного файла находится число n ($1 \leq n \leq 300\,000$) — размер парковки и число машин. Во второй строке записаны n чисел, i -е из которых p_i ($1 \leq p_i \leq n$) — место, которое хочет занять машина с номером i .

Формат выходного файла

Выведите n чисел: i -е число — номер парковочного места, которое было занято машиной с номером i .

Пример

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