

## Задача А. Хипуй!

Имя входного файла: `heap.in`  
Имя выходного файла: `heap.out`  
Ограничение по времени: 4 секунды  
Ограничение по памяти: 64 мегабайта

В этой задаче вам необходимо организовать структуру данных **Heap** для хранения целых чисел, над которой определены следующие операции:

- **Insert( $X$ )** — добавить в **Heap** число  $X$ ;
- **Extract** — достать из **Heap** наибольшее число (удалив его при этом).

### Формат входных данных

Во входном файле записано количество команд  $N$  ( $1 \leq N \leq 100\,000$ ), потом последовательность из  $N$  команд, каждая в своей строке.

Каждая команда имеет такой формат: "**0** <число>" или "**1**" что означает соответственно операции **Insert**(<число>) и **Extract**. Добавляемые числа находятся в интервале от 1 до  $10^7$  включительно.

Гарантируется, что при выполнении команды **Extract** в структуре находится по крайней мере один элемент.

### Формат выходных данных

В выходной файл для каждой команды извлечения необходимо вывести число, полученное при выполнении команды **Extract**.

### Примеры

| heap.in | heap.out |
|---------|----------|
| 7       | 100      |
| 0 100   | 50       |
| 0 10    |          |
| 1       |          |
| 0 5     |          |
| 0 30    |          |
| 0 50    |          |
| 1       |          |

## Задача В. Хип ли?

Имя входного файла: `stdin`  
Имя выходного файла: `stdout`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 64 мегабайта

Структуру данных Неар можно реализовать на основе массива.

Для этого должно выполняться *основное свойство Неар'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 Неар'ом.

### Формат входных данных

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 выходных данных

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

### Примеры

| stdin          | stdout |
|----------------|--------|
| 5<br>1 0 1 2 0 | NO     |
| 5<br>1 3 2 5 4 | YES    |

## Задача С. Сортировка кучей

Имя входного файла: `qsort.in`  
Имя выходного файла: `qsort.out`  
Ограничение по времени: 4 секунды  
Ограничение по памяти: 256 мегабайт

Отсортируйте список с помощью `heapsort`.

### Формат входных данных

В единственной строке входного файла содержится последовательность, содержащая не более чем 100000 целых чисел.

### Формат выходных данных

В единственной строке выходного файла выведите последовательность в неубывающем порядке.

### Примеры

| <code>qsort.in</code> | <code>qsort.out</code> |
|-----------------------|------------------------|
| 4 1 4 8 6 6 5         | 1 4 4 5 6 6 8          |

## Задача Е. Монополия

Имя входного файла: `monopoly.in`  
Имя выходного файла: `monopoly.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

В новом варианте игры “Монополия” появилась возможность объединять несколько предприятий в одно для увеличения приносимого ими дохода. При это в игре действуют следующие правила:

1. За один ход можно объединить ровно два предприятия в одно. При этом стоимость нового предприятия равна сумме стоимостей двух предприятий до объединения.
2. За совершение операции по объединению предприятий необходимо заплатить налог в размере 5% от стоимости объединяемых предприятий.

Коля уже заработал в игре много денег и теперь хочет объединить все свои предприятия в одно. Он заметил, что общая сумма уплаченного налога зависит от того, в каком порядке будут объединяться предприятия. Например, пусть у Коли есть четыре предприятия стоимостью 10, 11, 12 и 13. Если Коля сначала объединит предприятия 10 и 11 (это обойдется ему в \$1.05), потом результат — с 12 (\$1.65), и затем — с 13 (\$2.3), то всего он заплатит \$5. Если же сначала отдельно объединить 10 и 11 (\$1.05), потом — 12 и 13 (\$1.25) и, наконец, объединить два полученных предприятия (\$2.3), то в итоге он заплатит лишь \$4.6.

Помогите Коле определить минимальную сумму денег, необходимую для объединения всех его предприятий в одно.

### Формат входных данных

В единственной строке входного файла записано  $N$  натуральных чисел ( $2 \leq N \leq 200\,000$ ), каждое из которых не превосходит 10000 — стоимости Колиных предприятий.

### Формат выходных данных

В выходной файл выведите минимальную сумму денег необходимую для объединения всех Колиных предприятий в одно

### Примеры

| <code>monopoly.in</code> | <code>monopoly.out</code> |
|--------------------------|---------------------------|
| 10 11 12 13              | 4.6000000000              |
| 1 1                      | 0.1000000000              |

## Задача F. Фибоначчиева куча

|                         |                   |
|-------------------------|-------------------|
| Имя входного файла:     | стандартный ввод  |
| Имя выходного файла:    | стандартный вывод |
| Ограничение по времени: | 1 секунда         |
| Ограничение по памяти:  | 1024 мегабайта    |

Эта задача для тех, кто завалил регион (или его дисквалифицировали), отчаялся в олимпиадной проге и решил полностью посвятить себя теоретической информатике и продвинутым алгоритмам. Вам предстоит написать самую настоящую Фибоначчиеву кучу. Все запросы будут задаваться массивом  $a_i$ , где  $a_1$  вам дано изначально, а для  $i > 1$   $a_i = (a_{i-1} * b + c) \bmod 2^{32}$ .

К вам будут поступать запросы двух типов:

1. В случае, если  $a_i$  чётно, то в  $i$ -м запросе от вас будет требоваться добавить число  $a_i$  в кучу.
2. В случае, если  $a_i$  нечётно от вас будет требоваться узнать значение верхнего (максимального) элемента в куче.

### Формат входных данных

В первой строке вам дано единственное число  $n$  ( $1 \leq n \leq 10^8$ ). В следующей строке вам дано 3 числа  $a_1$   $b$  и  $c$  ( $0 \leq a_1, b, c \leq 2^{32}$ ). Гарантируется, что  $a_1$  чётно.

### Формат выходных данных

Для всех запросов второго типа выведите сумму ответов на них.

### Пример

| стандартный ввод | стандартный вывод |
|------------------|-------------------|
| 10<br>2 2 1      | 18                |

### Замечание

Так как Фибоначчиева куча сложная и её код трудно читать, в этой задаче не будет ревью кода.

## Задача G. И там клад зарыт!

Имя входного файла: `priority-queue.in`  
Имя выходного файла: `priority-queue.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Реализуйте структуру данных “очередь с приоритетами”, поддерживающую следующие операции:

1. Добавление элемента в очередь.
2. Удаление из очереди элемента с наибольшим приоритетом.
3. Изменение приоритета для произвольного элемента, находящегося в очереди.

### Формат входных данных

Программа получает на вход последовательность команд, по одной команде в каждой строке. Общее число команд не превосходит 30 000. Команда может иметь один из следующих форматов:

**ADD id priority** — добавить в очередь новый элемент с идентификатором `id` и приоритетом `priority`. Гарантируется, что в очереди нет элемента с таким идентификатором.

**POP** — удалить из очереди элемент с наибольшим значением приоритета. Если таких элементов несколько, то удаляется один (любой) из них. Гарантируется, что очередь не пуста.

**CHANGE id new\_priority** — изменить значение приоритета элемента с идентификатором `id` на значение `new_priority`. Гарантируется, что в очереди есть элемент с таким идентификатором.

Идентификаторы элементов — строки, состоящие из строчных латинских букв длиной не более 10 символов. Идентификаторы — произвольные целые числа.

В самом начале очередь пуста.

### Формат выходных данных

Для каждой команды типа **POP** выведите идентификатор удаленного элемента и, через пробел, значение его приоритета.

### Примеры

| priority-queue.in | priority-queue.out |
|-------------------|--------------------|
| ADD one 1         | three 3            |
| ADD two 2         | one 5              |
| ADD three 3       | two 2              |
| POP               |                    |
| CHANGE one 5      |                    |
| POP               |                    |
| POP               |                    |