Содержание

Обратите внимание, входные данные лежат в **стандартном потоке ввода** (он же stdin), вывести ответ нужно в **стандартный поток вывода** (он же stdout).

Обратите внимание на `GNU C++` компиляторы с суффиксом `inc`.
Подни можно пользоваться дополнительной библиотекой (`optimization.h`).
То есть, использовать быстрый ввод-вывод: **пример про числа и строки**.
И быструю аллокацию памяти (ускоряет vector-set-map-весь-STL): пример.

Для тех, кто хочет разобраться, как всё это работает.
Короткая версия быстрого ввода-вывода (тык) и короткая версия аллокатора (тык).

Must have

Задача 3A. Сортировка пар [0.5, 256]

Дан массив пар целых чисел a. Отсортируйте элементы массива по первому числу в паре. При равенстве первого числа, сортируйте по второму числу. При равенстве обоих элементов в паре, сортируйте по номеру пары во входных данных. В качества ответа выведите номера пар в порядке после сортировки массива (пары нумеруются от 1 до n).

Формат входных данных

На первой строке длина массива n ($1 \leqslant n \leqslant 10^5$).
На следующих n строках элементы массива a — пары целых чисел от 1 до 10^9.

Формат выходных данных

В одной строке выведите номера элементов массива a после его сортировки.

Примеры

stdin	stdout
9 3 2 1 1 2 1 2 2 3 2 1 2 1 2 3 2 2 3	2 6 7 3 4 9 1 5 8

Замечание

Пояснение к первому примеру:
Давайте допишем к каждой паре её номер, чтобы уметь отличать одинаковые пары.
Сортировка массива по правилам из условия будет выглядеть так, как показано ниже.
Оригинальный массив: [(3 2 1) (1 1 2) (2 1 3) (2 2 4) (3 2 5) (1 2 6) (1 2 7) (3 2 8) (2 3 9)]
Отсортированный массив: [(1 1 2) (1 2 6) (1 2 7) (2 1 3) (2 2 4) (2 3 9) (3 2 1) (3 2 5) (3 2 8)]

Подсказка по решению

Задача о том, чтобы научиться пользоваться стандартной сортировкой.
C++: `std::sort`

Задача 3B. Собственно Куча [0.2, 256]

Вам даны n запросов вида
«добавить целое число от 1 до 10^9 в множество» и
«извлечь из множества минимальное по значению число».

Формат входных данных

На первой строке n ($1 \leqslant n \leqslant 200\,000$).
Далее n запросов по одному строке, подробности в примерах.
Гарантируется, что не будет извлечений из пустой кучи.

Формат выходных данных

При каждом извлечении вывести извлекаемый элемент.

Пример

stdin	stdout
6 + 1 + 2 + 1 - - -	1 1 2
5 + 4 - + 3 + 2 -	4 2

Подсказка по решению

Задача о том, чтобы научиться пользоваться стандартной библиотекой.
C++: `set`, `priority_queue`

Задачи здорового человека

Задача 3C. Глубина быстрой сортировки [0.2, 256]

По сути вам нужно написать свой quick-sort.

Дан массив целых чисел, вам предлагается написать для него быструю сортировку, которая, на каждом уровне рекурсии делает разделение по элементу, который лежит по середине массива (если длина массива чётная, нужно взять левый из двух элементов).

При разделении массив делится на три части A — элементы меньшие x, B — элементы равные x, C — элементы большие x. Важно, чтобы элементы в A и C шли в том же порядке, в котором они шли в p.

Далее происходят рекурсивные вызовы от A и C. Рекурсия останавливается, если ей передали массив длины не более 1. Ваша задача — реализовать ровно описанное и запомнить максимальную глубину, на которую спускалась рекурсия.

Формат входных данных

В первой строке целое число n ($1 \leqslant n \leqslant 100$) — длина массива.

Формат выходных данных

Выведите одно число — глубину рекурсии варации быстрой сортировки, описанной в условии.

Примеры

stdin	stdout
6 1 3 2 4 5 6	3
6 1 1 2 1 1 1	2

Замечание

Пояснение к примерам:

Глубина 0. $[1, 3, 2, 4, 5, 6]$: делимся по 2, вызываемся от $[1]$, $[3, 4, 5, 6]$.
Глубина 1. $[3, 4, 5, 6]$: делимся по 4, вызываемся от $[3]$, $[5, 6]$.
Глубина 2. $[5, 6]$ делимся по 5, вызываемся от $[]$, $[6]$.
Глубина 3. $[6]$. Выходим из рекурсии.
Итоговая максимальная глубина 3.

Глубина 0. $[1, 1, 2, 1, 1, 1]$: делимся по 2, вызываемся от $[1, 1, 1, 1, 1]$ и $[]$.
Глубина 1. $[1, 1, 1, 1, 1]$: делимся по 1, вызываемся от $[]$ и $[]$.
Глубина 2. $[]$. Выходим из рекурсии.
Итоговая максимальная глубина 2.

Задача 3D. Куча исправлений [0.2, 256]

Жил был массив $p = [1, 2, \ldots, n]$, где $n \leqslant 1\,000$. Массив испортили в $k \leqslant 10$ местах, в этих местах исходные числа заменили на числа от 0 до $n+1$.

Массив a называется кучей, если в a_1 хранится минимум, и $\forall i$ $a_i \leqslant a_{2i}$ и $a_i \leqslant a_{2i+1}$.

Вам нужно за не более чем 90 swap-ов элементов сделать из массива корректную кучу.

Формат входных данных

На первой строке числа n $(1 \leqslant n \leqslant 1\,000)$.

На следующей строке испорченная версия массива p.

Формат выходных данных

На первой строке выведите число m swap-ов, которые вы хотите совершить.

Следующие m строк должны содержать пары i j — номера элементов, которые вы хотите поменять местами.

Примеры

stdin	stdout
6 1 2 3 0 5 6	2 4 2 2 1
8 1 9 3 4 5 6 7 8	2 4 2 8 4
6 5 2 3 4 5 0	4 2 1 4 2 6 3 3 1

Замечание

Пояснение к первому примеру:
Проталкиваем 0 вверх: swap(4,2), swap(2,1).

Пояснение ко второму примеру:
Проталкиваем 9 вниз: swap(2,4), swap(4,8).

Пояснение к третьему примеру:
Нужно 5 из корня протолкнуть вниз: swap(1,2), swap(2,4)
и 0 из конца массива протолкнуть вверх: swap(6,3), swap(3,1).

Подсказка по решению

Напишите свою кучу.

Для искателей острых ощущений

Задача 3E. Q-я порядковая статистика [0.7, 256]

Вам дан массив из n случайных целых чисел. Ваша задача — отсортировать массив и вывести q-е число в получившемся порядке.

Формат входных данных

На первой строке числа n, q. ($1 \leqslant q \leqslant n \leqslant 10^7$). На второй строке пара целых чисел a, b от 1 до 10^9, используемая в генераторе случайных чисел.

```
1. unsigned int cur = 0; // беззнаковое 32-битное число
2. unsigned int nextRand24() {
3.     cur = cur * a + b; // вычисляется с переполнениями
4.     return cur >> 8; // число от 0 до 2^24 - 1.
5. }
6. unsigned int nextRand32() {
7.     unsigned int a = nextRand24(), b = nextRand24();
8.     return (a << 8) ^ b; // число от 0 до 2^32 - 1.
9. }
```

Элементы массива генерируются последовательно. x_i = nextRand32();

Формат выходных данных

Выведите ответ на запрос.

Примеры

stdin	stdout
6 3 239 13	197852696

Замечание

Сгенерированный массив: $12, 130926, 3941054950, 2013898548, 197852696, 2753287507$.

Внимание: нельзя пользоваться стандартными функциями, например, nth_element.

Задача 3F. Менеджер памяти [0.5, 256]

Пете поручили написать менеджер памяти для новой стандартной библиотеки языка H++. В распоряжении у менеджера находится массив из N последовательных ячеек памяти, пронумерованных от 1 до N. Задача менеджера – обрабатывать запросы приложений на выделение и освобождение памяти. Запрос на выделение памяти имеет один параметр K. Такой запрос означает, что приложение просит выделить ему K последовательных ячеек памяти. Если в распоряжении менеджера есть хотя бы один свободный блок из K последовательных ячеек, то он обязан в ответ на запрос выделить такой блок. При этом непосредственно перед самой первой ячейкой памяти выделяемого блока не должно располагаться свободной ячейки памяти. После этого выделенные ячейки становятся занятыми и не могут быть использованы для выделения памяти, пока не будут освобождены. Если блока из K последовательных свободных ячеек нет, то запрос отклоняется. Запрос на освобождение памяти имеет один параметр T. Такой запрос означает, что менеджер должен освободить память, выделенную ранее при обработке запроса с порядковым номером T. Запросы нумеруются, начиная с единицы. Гарантируется, что запрос с номером T – запрос на выделение, причем к нему еще не применялось освобождение памяти. Освобожденные ячейки могут снова быть использованы для выделения памяти. Если запрос с номером T был отклонен, то текущий запрос на освобождение памяти игнорируется. Требуется написать менеджер памяти, удовлетворяющий приведенным критериям.

Формат входных данных

Первая строка входного файла содержит числа N и M – количество ячеек памяти и количество запросов соответственно ($1 \leqslant N \leqslant 2^{31} - 1$; $1 \leqslant M \leqslant 10^5$). Каждая из следующих M строк содержит по одному числу: $(i+1)$-я строка входного файла ($1 \leqslant i \leqslant M$) содержит либо положительное число K, если i-й запрос – запрос на выделение с параметром K ($1 \leqslant K \leqslant N$), либо отрицательное число $-T$, если i-й запрос – запрос на освобождение с параметром T ($1 \leqslant T < i$).

Формат выходных данных

Для каждого запроса на выделение памяти выведите в выходной файл результат обработки этого запроса: для успешных запросов выведите номер первой ячейки памяти в выделенном блоке, для отклоненных запросов выведите число -1. Результаты нужно выводить в порядке следования запросов во входном файле.

Примеры

stdin	stdout
6 8 2 3 -1 3 3 -5 2 2	1 3 -1 -1 1 -1