

## Содержание

<b>Must have</b>	<b>2</b>
Задача 4А. Квадратный корень [0.2 sec, 256 mb]	2
Задача 4В. Поиск [0.2 sec, 256 mb]	3
Задача 4С. Быстрый поиск в массиве [0.2 sec, 256 mb]	4
Задача 4D. Собственно Куча [0.2 sec, 256 mb]	5
<b>Обязательные задачи</b>	<b>6</b>
Задача 4Е. Минимум и максимум [0.2 sec, 256 mb]	6
Задача 4F. Для любителей статистики [0.2 sec, 256 mb]	7
Задача 4G. Вербочки [0.2 sec, 256 mb]	8
Задача 4H. Линейная сумма [2 sec, 256 mb]	9
Задача 4I. Управление Памятью [0.2 sec, 256 mb]	10
<b>Дополнительные задачи</b>	<b>12</b>
Задача 4J. Менеджер памяти [0.3 sec, 256 mb]	12
Задача 4K. Лифт [0.2 sec, 256 mb]	13
Задача 4L. K-Best [0.3 sec, 256 mb]	14

---

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 4А. Квадратный корень [0.2 sec, 256 mb]

Дано целое число  $n$  от 0 до  $2^{64} - 1$ . Ваша задача — найти  $\lfloor \sqrt{n} \rfloor$ .

#### Формат входных данных

Мультитест. На каждой строке по числу  $n$ . Не более 1000 строк.

#### Формат выходных данных

Для каждого  $n$  на отдельной строке ответ на запрос.

#### Примеры

stdin	stdout
0	0
1	1
2	1
3	1
4	2
5	2

#### Замечание

Целые числа такого типа помещаются в `uint64_t` в **C++11**.

Эту задачу обязательно сдать бинпоиском в целых числах.

Можете также поэкспериментировать с вещественными типами `double`, `long double`.

### Задача 4В. Поиск [0.2 сек, 256 mb]

В этой задаче нужно уметь выяснять, содержится ли число в последовательности.

#### Формат входных данных

В первой строке входного файла заданы через пробел два целых числа  $n$  и  $k$  ( $1 \leq n \leq 300\,000$ ,  $1 \leq k \leq 300\,000$ ). Во второй строке задана последовательность из  $n$  отсортированных целых чисел  $a_1, a_2, \dots, a_n$ , записанных через пробел ( $1 \leq a_i \leq 10^9$ ). В третьей строке записаны запросы —  $k$  целых чисел  $b_1, b_2, \dots, b_k$  записанных через пробел, в порядке возрастания ( $1 \leq b_j \leq 10^9$ ).

#### Формат выходных данных

В выходной файл выведите  $k$  строк. В  $j$ -ой строке выведите “YES”, если число  $b_j$  содержится в последовательности  $\{a_i\}$ , и “NO” в противном случае.

#### Примеры

stdin	stdout
3 3	NO
2 3 5	YES
1 2 3	YES
3 4	YES
1 2 2	YES
1 2 4 5	NO
	NO

#### Замечание

В этой задаче можно использовать STL =>

### Задача 4С. Быстрый поиск в массиве [0.2 сек, 256 mb]

Дан массив из  $N$  целых чисел. Все числа от  $-10^9$  до  $10^9$ .

Нужно уметь отвечать на запросы вида “Сколько чисел имеют значения от  $L$  до  $R$ ?”.

#### Формат входных данных

Число  $N$  ( $1 \leq N \leq 10^5$ ). Далее  $N$  целых чисел.

Затем число запросов  $K$  ( $1 \leq K \leq 10^5$ ).

Далее  $K$  пар чисел  $L, R$  ( $-10^9 \leq L \leq R \leq 10^9$ ) — собственно запросы.

#### Формат выходных данных

Выведите  $K$  чисел — ответы на запросы.

#### Пример

stdin	stdout
5	5 2 2 0
10 1 10 3 4	
4	
1 10	
2 9	
3 4	
2 2	

#### Замечание

В этой задаче можно использовать STL => Можно не использовать...

Если не используете, важно написать ровно одну функцию “бинпоиск”.

### Задача 4D. Собственно Куча [0.2 сек, 256 mb]

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  $n$  ( $1 \leq n \leq 200\,000$ ).  
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
+ 1	1
+ 2	2
+ 1	
-	
-	
-	
5	4
+ 4	2
-	
+ 3	
+ 2	
-	

#### Замечание

`priority_queue`, `push_heap` и `set` использовать **запрещено**.  
Нужно написать бинарную кучу ручками и получить удовольствие.

## Обязательные задачи

### Задача 4Е. Минимум и максимум [0.2 sec, 256 mb]

Пусть есть мультимножество целых чисел (множество с возможными повторениями). Необходимо реализовать структуру данных для их хранения, поддерживающую следующие операции: `GetMin` — извлечение минимума, `GetMax` — извлечение максимума, `Insert(N)` — добавление числа в множество.

#### Формат входных данных

В первой строке входного файла записано одно целое число  $N$  ( $1 \leq N \leq 100\,000$ ) — число запросов к структуре. Затем в  $N$  строках следуют запросы по одному в строке: `GetMin`, `GetMax`, `Insert(A)` — извлечение минимума, максимума и добавление числа  $A$  ( $1 \leq A \leq 2^{31} - 1$ ). Запросы корректны, то есть нет операций извлечения для пустого множества.

#### Формат выходных данных

Для каждого запроса `GetMin` или `GetMax` выведите то число, которое было извлечено.

#### Примеры

stdin	stdout
10	1
Insert(100)	100
Insert(99)	1
Insert(1)	2
Insert(2)	99
GetMin	
GetMax	
Insert(1)	
GetMin	
GetMin	
GetMax	

#### Замечание

В этой задаче **запрещается** использовать STL-контейнеры сложнее `vector`-а. Где-то тут вам могут помочь обратные ссылки. Ещё может помочь трюк с «удалением произвольного элемента».

#### Задача 4F. Для любителей статистики [0.2 сек, 256 mb]

Вы никогда не задумывались над тем, сколько человек за год перевозят трамваи города с десятиллионным населением, в котором каждый третий житель пользуется трамваем по два раза в день?

Предположим, что на планете Земля  $n$  городов, в которых есть трамваи. Любители статистики подсчитали для каждого из этих городов, сколько человек перевезено трамваями этого города за последний год. Из этих данных была составлена таблица, в которой города были отсортированы по алфавиту. Позже выяснилось, что для статистики названия городов несущественны, и тогда их просто заменили числами от 1 до  $n$ . Поисковая система, работающая с этими данными, должна уметь быстро отвечать на вопрос, есть ли среди городов с номерами от  $l$  до  $r$  такой, что за год трамваи этого города перевезли ровно  $x$  человек. Вам предстоит реализовать этот модуль системы.

#### Формат входных данных

В первой строке дано целое число  $n$ ,  $0 < n < 70\,000$ . В следующей строке приведены статистические данные в виде списка целых чисел через пробел,  $i$ -е число в этом списке — количество человек, перевезенных за год трамваями  $i$ -го города. Все числа в списке положительны и не превосходят  $10^9 - 1$ . В третьей строке дано количество запросов  $q$ ,  $0 < q < 70\,000$ . В следующих  $q$  строках перечислены запросы. Каждый запрос — это тройка целых чисел  $l$ ,  $r$  и  $x$ , записанных через пробел ( $1 \leq l \leq r \leq n$ ,  $0 < x < 10^9$ ).

#### Формат выходных данных

Выведите строку длины  $q$ , в которой  $i$ -й символ равен 1, если ответ на  $i$ -й запрос утвердителен, и 0 в противном случае.

#### Пример

stdin	stdout
5	10101
123 666 314 666 434	
5	
1 5 314	
1 5 578	
2 4 666	
4 4 713	
1 1 123	

#### Замечание

Разобрано на практике.

### Задача 4G. Вербочки [0.2 sec, 256 mb]

С утра шел дождь, и ничего не предвещало беды. Но к обеду выглянуло солнце, и в лагерь заглянула СЭС. Пройдя по всем домикам и корпусам, СЭС вынесла следующий вердикт: бельевые веревки в жилых домиках не удовлетворяют нормам СЭС. Как выяснилось, в каждом домике должно быть ровно по одной бельевой веревке, и все веревки должны иметь одинаковую длину. В лагере имеется  $N$  бельевых веревок и  $K$  домиков. Чтобы лагерь не закрыли, требуется так нарезать данные веревки, чтобы среди получившихся вервочек было  $K$  одинаковой длины. Размер штрафа обратно пропорционален длине бельевых веревок, которые будут развешены в домиках. Поэтому начальство лагеря стремится максимизировать длину этих вервочек.

#### Формат входных данных

В первой строке заданы два числа —  $N$  ( $1 \leq N \leq 10\,001$ ) и  $K$  ( $1 \leq K \leq 10\,001$ ). Далее в каждой из последующих  $N$  строк записано по одному числу — длине очередной бельевой веревки. Длина веревки задана в сантиметрах. Все длины лежат в интервале от 1 сантиметра до 100 километров включительно.

#### Формат выходных данных

В выходной файл следует вывести одно целое число — максимальную длину вервочек, удовлетворяющую условию, в сантиметрах. В случае, если лагерь закроют, выведите 0.

#### Пример

stdin	stdout
4 11 802 743 457 539	200



### Задача 4Н. Линейная сумма [2 сек, 256 mb]

Есть  $n$  случайных точек на прямой с координатами от 0 до  $2^{32} - 1$ . У каждой точки есть значение от 0 до  $2^{32} - 1$ . Вам нужно обработать  $q$  случайных запросов вида “сумма значений точек, с координатами от  $l$  до  $r$  включительно”.

#### Формат входных данных

На первой строке числа  $n, q$ . ( $1 \leq n \leq 2^{20}, 1 \leq q \leq 2^{23}$ ). На второй строке пара целых чисел  $a, b$  от 1 до  $10^9$ , используемая в генераторе случайных чисел.

```
1. uint32_t cur = 0; // беззнаковое 32-битное число
2. uint32_t nextRand24() {
3.     cur = cur * a + b; // вычисляется с переполнениями
4.     return cur >> 8; // число от 0 до  $2^{24} - 1$ .
5. }
6. uint32_t nextRand32() {
7.     uint32_t a = nextRand24(), b = nextRand24();
8.     return (a << 8) ^ b; // число от 0 до  $2^{32} - 1$ .
9. }
```

Каждая точка генерируется следующим образом:

```
1. value = nextRand32(); // значение точки
2. x = nextRand32(); // координата точки
```

Каждый запрос генерируется следующим образом:

```
1. l = nextRand32();
2. r = nextRand32();
3. if (l > r) swap(l, r); // получили отрезок [l..r]
   Сперва генерируются точки, затем запросы.
```

#### Формат выходных данных

Выведите сумму ответов на все запросы по модулю  $2^{32}$ .

#### Примеры

stdin	stdout
5 5	3950632748
13 239	

#### Замечание

Заметьте, координаты точек могут повторяться.

Если есть несколько точек с равным  $x$ , не выбрасывайте их, сохраните их все ;)

$p = \{value, x\}$

$p[0] = \{13, 41645\}$

$p[1] = \{7695587, 1253435649\}$

$p[2] = \{749170640, 2683600557\}$

$p[3] = \{2444595881, 1270561959\}$

$p[4] = \{3436107648, 486388002\}$

Напомним, сложение по модулю  $2^{32}$  – вычисление в типе `unsigned int (uint32_t)`.

#### Задача 4I. Управление Памятью [0.2 sec, 256 mb]

Недавно школьники изобрели новый язык программированию. Язык называется D++. На самом деле не важно, слышали ли вы уже про этот язык, или нет. Важно, что, чтобы запускать программы на языке D++, нужна новая операционная система. Новая ОС должна быть достаточно мощной, должна работать быстро и иметь кучу возможностей. Но это всё в будущем. А сейчас нужно... Нет. Не нужно придумывать название для новой ОС. Вам предстоит реализовать первый модуль в новой ОС. Конечно, этот модуль – модуль управления памятью. Давайте обсудим, как он должен работать.

Наша ОС будет выделять память кусками, назовём их “блоки”. Блоки нумеруются целыми числами от 1 до  $N$ . Когда ОС требуется больше памяти, она обращается к модулю управления памятью. Чтобы обработать такой запрос, модуль должен найти свободный блок памяти с минимальным номером. Можете смело предположить, что памяти достаточно, чтобы обработать все запросы. Определим понятие “свободный блок”. В момент первого запроса к памяти все блоки свободны. Занятый блок становится свободным, если к нему нет запросов доступа в течении  $T$  минут. Вас может удивить запись “запрос доступа к занятому блоку”. Что же это значит? Ответ прост: в любой момент времени модуль управления памятью может получить запрос доступа к занятому блоку памяти. Обработка такого запроса происходит следующим образом: модуль проверяет, правда ли запрашиваемый блок занят. Если да, запрос считается успешным и блок остаётся занятым ещё на  $T$  минут. Иначе запрос считается ошибочным. Это всё, что нужно знать. Ваша задача – реализовать алгоритм для  $N = 30\,000$  и  $T = 10$  минутам.

#### Формат входных данных

Каждая строка ввода содержит запрос доступа к блоку памяти или запрос на выделение памяти. Запрос выделения памяти имеет форму “<Time> +”, где <Time> – неотрицательное целое число не более 65 000. Время даётся в секундах. Запрос доступа к памяти имеет форму “<Time> . <BlockNo>”, где <Time> значит то же, что и выше, а <BlockNo> – целое число от 1 до  $N$ . Всего будет не более 80 000 запросов.

#### Формат выходных данных

Для каждой строки ввода нужно вывести ровно одну строку с ответом на запрос. На запрос выделения памяти выведите целое число – номер выделенного блока. Как было уже замечено выше, вы можете предположить, что одновременно будет нужно не более  $N$  блоков. Для запроса доступа к блоку памяти нужно вывести один символ:

- “+” если запрос успешен (блок занят)
- “-” если запрос ошибочен (блок свободен)

Запросы даны в порядке возрастания времени. Запросы с одинаковым временем должны обрабатываться в том порядке, в котором даны.

**Примеры**

stdin	stdout
1 +	1
1 +	2
1 +	3
2 . 2	+
2 . 3	+
3 . 30000	-
601 . 1	-
601 . 2	+
602 . 3	-
602 +	1
602 +	3
1202 . 2	-

## Дополнительные задачи

### Задача 4J. Менеджер памяти [0.3 sec, 256 mb]

Пете поручили написать менеджер памяти для новой стандартной библиотеки языка C++. 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 ( $1 \leq N \leq 2^{31} - 1$ ;  $1 \leq M \leq 10^5$ ). Каждая из следующих  $M$  строк содержит по одному числу:  $(i+1)$ -я строка входного файла ( $1 \leq i \leq M$ ) содержит либо положительное число  $K$ , если  $i$ -й запрос – запрос на выделение с параметром  $K$  ( $1 \leq K \leq N$ ), либо отрицательное число  $-T$ , если  $i$ -й запрос – запрос на освобождение с параметром  $T$  ( $1 \leq T < i$ ).

#### 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	1
2	3
3	-1
-1	-1
3	1
3	-1
-5	
2	
2	

### Задача 4К. Лифт [0.2 sec, 256 mb]

Высокое здание, состоящее из  $N$  этажей, оснащено только одним лифтом. Парковка находится ниже фундамента здания, что соответствует одному этажу ниже первого. Этажи пронумерованы от 1 до  $N$  снизу вверх. Про каждый этаж известно количество человек, желающих спуститься на лифте на парковку. Пусть для  $i$ -го этажа эта величина равна  $A_i$ . Известно, что лифт не может перевозить более  $C$  человек одновременно, а также то, что на преодоление расстояния в один этаж (не важно вверх или вниз) ему требуется  $P$  секунд. Какое наибольшее количество человек лифт может перевезти на парковку за  $T$  секунд, если изначально он находится на уровне парковки?

#### Формат входных данных

В первой строке входного файла содержатся целые числа  $N, C, P, T$  ( $1 \leq N \leq 100$ ,  $1 \leq C \leq 10^9$ ,  $1 \leq P \leq 10^9$ ,  $1 \leq T \leq 10^9$ ). Вторая строка содержит последовательность  $N$  целых чисел  $A_1, A_2, \dots, A_N$  ( $0 \leq A_i \leq 10^9$ ). Сумма всех значений последовательности не превосходит  $10^9$ .

#### Формат выходных данных

Выведите наибольшее количество человек, которое лифт успеет перевезти на парковку.

#### Пример

stdin	stdout
4 5 2 15 0 1 2 3	3
4 5 2 18 0 1 2 3	5
3 2 1 9 1 1 1	3

#### Задача 4L. K-Best [0.3 sec, 256 mb]

У Демьяны есть  $n$  драгоценностей. Каждая из драгоценностей имеет ценность  $v_i$  и вес  $w_i$ . С тех пор, как её мужа Джонни уволили в связи с последним финансовым кризисом, Демьяна решила продать несколько драгоценностей. Для себя она решила оставить лишь  $k$  лучших. Лучших в смысле максимизации достаточно специфического выражения: пусть она оставила для себя драгоценности номер  $i_1, i_2, \dots, i_k$ , тогда максимальной должна быть величина

$$\frac{\sum_{j=1}^k v_{i_j}}{\sum_{j=1}^k w_{i_j}}$$

Помогите Демьяне выбрать  $k$  драгоценностей требуемым образом.

#### Формат входных данных

На первой строке  $n$  и  $k$  ( $1 \leq k \leq n \leq 100\,000$ ).

Следующие  $n$  строк содержат пары целых чисел  $v_i, w_i$  ( $0 \leq v_i \leq 10^6, 1 \leq w_i \leq 10^6$ , сумма всех  $v_i$  не превосходит  $10^7$ , сумма всех  $w_i$  также не превосходит  $10^7$ ).

#### Формат выходных данных

Выведите  $k$  различных чисел от 1 до  $n$  — номера драгоценностей. Драгоценности нумеруются в том порядке, в котором перечислены во входных данных. Если есть несколько оптимальных ответов, выведите любой.

#### Пример

stdin	stdout
3 2 1 1 1 2 1 3	1 2