

## Задача А. Двоичное дерево поиска

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

Реализуйте сбалансированное двоичное дерево поиска.

### Формат входных данных

Входной файл содержит описание операций с деревом, их количество не превышает 100000. В каждой строке находится одна из следующих операций:

- **insert**  $x$  — добавить в дерево ключ  $x$ . Если ключ  $x$  в дереве уже есть, то ничего делать не надо.
- **delete**  $x$  — удалить из дерева ключ  $x$ . Если ключа  $x$  в дереве нет, то ничего делать не надо.
- **exists**  $x$  — если ключ  $x$  есть в дереве, выведите «**true**», иначе «**false**»
- **next**  $x$  — выведите минимальный элемент в дереве, строго больший  $x$ , или «**none**», если такого нет.
- **prev**  $x$  — выведите максимальный элемент в дереве, строго меньший  $x$ , или «**none**», если такого нет.

Все числа во входном файле целые и по модулю не превышают  $10^9$ .

### Формат выходных данных

Выведите последовательно результат выполнения всех операций **exists**, **next**, **prev**. Следуйте формату выходного файла из примера.

### Примеры

bst.in	bst.out
insert 2	true
insert 5	false
insert 3	5
exists 2	3
exists 4	none
next 4	3
prev 4	
delete 5	
next 4	
prev 4	

## Задача В. И снова сумма...

Имя входного файла: `sum2.in`  
Имя выходного файла: `sum2.out`  
Ограничение по времени: 3 секунды  
Ограничение по памяти: 256 мегабайт

Реализуйте структуру данных, которая поддерживает множество  $S$  целых чисел, с которым разрешается производить следующие операции:

- $add(i)$  — добавить в множество  $S$  число  $i$  (если он там уже есть, то множество не меняется);
- $sum(l, r)$  — вывести сумму всех элементов  $x$  из  $S$ , которые удовлетворяют неравенству  $l \leq x \leq r$ .

### Формат входных данных

Исходно множество  $S$  пусто. Первая строка входного файла содержит  $n$  — количество операций ( $1 \leq n \leq 300\,000$ ). Следующие  $n$  строк содержат операции. Каждая операция имеет вид либо «+  $i$ », либо «?  $l$   $r$ ». Операция «?  $l$   $r$ » задает запрос  $sum(l, r)$ .

Если операция «+  $i$ » идет во входном файле в начале или после другой операции «+», то она задает операцию  $add(i)$ . Если же она идет после запроса «?», и результат этого запроса был  $y$ , то выполняется операция  $add((i + y) \bmod 10^9)$ .

Во всех запросах и операциях добавления параметры лежат в интервале от 0 до  $10^9$ .

### Формат выходных данных

Для каждого запроса выведите одно число — ответ на запрос.

### Примеры

sum2.in	sum2.out
6	3
+ 1	7
+ 3	
+ 3	
? 2 4	
+ 1	
? 2 4	

## Задача С. К-ый максимум

Имя входного файла: `kthmax.in`  
Имя выходного файла: `kthmax.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 64 мегабайта

Напишите программу, реализующую структуру данных, позволяющую добавлять и удалять элементы, а также находить  $k$ -й максимум.

### Формат входных данных

Первая строка входного файла содержит натуральное число  $n$  — количество команд ( $n \leq 100\,000$ ). Последующие  $n$  строк содержат по одной команде каждая. Команда записывается в виде двух чисел  $c_i$  и  $k_i$  — тип и аргумент команды соответственно ( $|k_i| \leq 10^9$ ). Поддерживаемые команды:

- $+1$  (или просто  $1$ ): Добавить элемент с ключом  $k_i$ .
- $0$ : Найти и вывести  $k_i$ -й максимум.
- $-1$ : Удалить элемент с ключом  $k_i$ .

Гарантируется, что в процессе работы в структуре не требуется хранить элементы с равными ключами или удалять несуществующие элементы. Также гарантируется, что при запросе  $k_i$ -го максимума, он существует.

### Формат выходных данных

Для каждой команды нулевого типа в выходной файл должна быть выведена строка, содержащая единственное число —  $k_i$ -й максимум.

### Примеры

<code>kthmax.in</code>	<code>kthmax.out</code>
11	7
+1 5	5
+1 3	3
+1 7	10
0 1	7
0 2	3
0 3	
-1 5	
+1 10	
0 1	
0 2	
0 3	

## Задача D. Вперёд!

Имя входного файла: `movetofront.in`  
Имя выходного файла: `movetofront.out`  
Ограничение по времени: 3 секунды  
Ограничение по памяти: 256 мегабайт

Капрал Дукар любит раздавать приказы своей роте. Самый любимый его приказ — «Вперёд!». Капрал строит солдат в ряд и отдаёт некоторое количество приказов, каждый из которых звучит так: «Рядовые с  $l_i$  по  $l_j$  — вперёд!»

Перед тем, как Дукар отдал первый приказ, солдаты были пронумерованы от 1 до  $n$  слева направо. Услышав приказ «Рядовые с  $l_i$  по  $l_j$  — вперёд!», солдаты, стоящие на местах с  $l_i$  по  $l_j$  включительно, продвигаются в начало ряда в том же порядке, в котором были.

Например, если в какой-то момент солдаты стоят в порядке 2, 3, 6, 1, 5, 4, то после приказа «Рядовые с 2 по 4 — вперёд!», порядок будет таким: 3, 6, 1, 2, 5, 4. А если потом Капрал вышлет вперёд солдат с 3 по 4, то порядок будет уже таким: 1, 2, 3, 6, 5, 4.

Вам дана последовательность приказов Капрала. Найдите порядок, в котором будут стоять солдаты после исполнения всех приказов.

### Формат входных данных

В первой строке входного файла указаны числа  $n$  и  $m$  ( $2 \leq n \leq 100\,000$ ,  $1 \leq m \leq 100\,000$ ) — число солдат и число приказов. Следующие  $m$  строк содержат приказы в виде двух целых чисел:  $l_i$  и  $r_i$  ( $1 \leq l_i \leq r_i \leq n$ ).

### Формат выходных данных

Выведите в выходной файл  $n$  целых чисел — порядок, в котором будут стоять солдаты после исполнения всех приказов.

### Примеры

<code>movetofront.in</code>	<code>movetofront.out</code>
6 3 2 4 3 5 2 2	1 4 5 2 3 6

## Задача Е. Переворот

Имя входного файла: `reverse.in`  
Имя выходного файла: `reverse.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 64 мегабайта

Дан массив. Надо научиться обрабатывать два типа запросов.

- 1 L R - перевернуть отрезок [L, R]
- 2 L R - найти минимум на отрезке [L, R]

### Формат входных данных

Первая строка файла содержит два числа  $n, m$ . ( $1 \leq n, m \leq 10^5$ ) Во второй строке находится  $n$  чисел  $a_i$  ( $1 \leq a_i \leq 10^9$ )- исходный массив. Остальные  $m$  строк содержат запросы, в формате описанном в условии. Для чисел L,R выполняется ограничение ( $1 \leq L \leq R \leq n$ ).

### Формат выходных данных

На каждый запрос типа 2, во входной файл выведите ответ на него, в отдельной строке.

### Примеры

reverse.in	reverse.out
10 7	3
5 3 2 3 12 6 7 5 10 12	2
2 4 9	2
1 4 6	2
2 1 8	
1 1 8	
1 8 9	
2 1 7	
2 3 6	

## Задача F. Своппер

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

Современные компьютеры закидываются  
в десятки раз эффективнее человека

---

Рекламный проспект OS Vista-N

Перед возвращением в штаб-квартиру корпорации Аазу и Скиву пришлось заполнить на местной таможне декларацию о доходах за время визита. Получилась довольно внушительная последовательность чисел. Обработка этой последовательности заняла весьма долгое время.

— Своппер кривой, — со знанием дела сказал таможенник.

— А что такое своппер? — спросил любопытный Скив.

Ааз объяснил, что своппер — это структура данных, которая умеет делать следующее.

- Взять отрезок чётной длины от  $x$  до  $y$  и поменять местами число  $x$  с  $x + 1$ ,  $x + 2$  с  $x + 3$ , и т.д.
- Посчитать сумму чисел на произвольном отрезке от  $a$  до  $b$ .

Учитывая, что обсчёт может затянуться надолго, корпорация «МИФ» попросила Вас решить проблему со своппером и промоделировать ЭТО эффективно.

### Формат входных данных

Во входном файле заданы один или несколько тестов. В первой строке каждого теста записаны число  $N$  — длина последовательности и число  $M$  — число операций ( $1 \leq N, M \leq 100\,000$ ). Во второй строке теста содержится  $N$  целых чисел, не превосходящих  $10^6$  по модулю — сама последовательность. Далее следуют  $M$  строк — запросы в формате 1  $x_i$   $y_i$  — запрос первого типа, и 2  $a_i$   $b_i$  — запрос второго типа. Сумма всех  $N$  и  $M$  по всему файлу не превосходит 200 000. Файл завершается строкой из двух нулей. Гарантируется, что  $x_i < y_i$ , а  $a_i \leq b_i$ .

### Формат выходных данных

Для каждого теста выведите ответы на запросы второго типа, как показано в примере. Разделяйте ответы на тесты пустой строкой.

### Примеры

swapper.in	swapper.out
5 5	Swapper 1:
1 2 3 4 5	10
1 2 5	9
2 2 4	2
1 1 4	
2 1 3	
2 4 4	
0 0	

## Задача G. Река

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

Во Флатландии протекает богатая рыбой река Большой Флат. Много лет назад река была поделена между  $n$  рыболовными предприятиями, каждое из которых получило непрерывный отрезок реки. При этом  $i$ -е предприятие, если рассматривать их по порядку, начиная от истока, изначально получило отрезок реки длиной  $a_i$ .

С тех пор с рыболовными предприятиями во Флатландии  $k$  раз происходили различные события. Каждое из событий было одного из двух типов: банкротство некоторого предприятия или разделение некоторого предприятия на два. При некоторых событиях отрезок реки, принадлежащий предприятию, с которым это событие происходит, делится на две части. Каждый такой отрезок имеет длину большую или равную 2. Деление происходит по следующему правилу. Если отрезок имеет четную длину, то он делится на две равные части. Иначе он делится на две части, длины которых различаются ровно на единицу, при этом часть, которая ближе к истоку реки, имеет меньшую длину.

При банкротстве предприятия происходит следующее. Отрезок реки, принадлежавший обанкротившемуся предприятию, переходит к его соседям. Если у обанкротившегося предприятия один сосед, то этому соседу целиком передается отрезок реки обанкротившегося предприятия. Если же соседей двое, то отрезок реки делится на две части описанным выше способом, после чего каждый из соседей присоединяет к своему отрезку ближайшую к нему часть. При разделении предприятия отрезок реки, принадлежавший разделяемому предприятию, всегда делится на две части описанным выше способом. Разделившееся предприятие ликвидируется, и образуются два новых предприятия. Таким образом, после каждого события каждое предприятие владеет некоторым отрезком реки.

Министерство финансов Флатландии предлагает ввести налог на рыболовные предприятия, пропорциональный квадрату длины отрезка реки, принадлежащего соответствующему предприятию. Чтобы проанализировать, как будет работать этот налог, министр хочет по имеющимся данным узнать, как изменялась величина, равная сумме квадратов длин отрезков реки, принадлежащих предприятиям, после каждого произошедшего события.

Требуется написать программу, которая по заданному начальному разделению реки между предприятиями и списку событий, происходивших с предприятиями, определит, чему равна сумма квадратов длин отрезков реки, принадлежащих предприятиям, в начальный момент времени и после каждого события.

### Формат входных данных

Первая строка входного файла содержит два целых числа:  $n$  и  $p$  — исходное количество предприятий ( $2 \leq n \leq 100000$ ) и номер подзадачи ( $0 \leq p \leq 4$ ) (считайте его просто так).

Вторая строка входного файла содержит  $n$  целых чисел  $a_1, a_2, \dots, a_n$  — длины исходных отрезков реки.

Третья строка входного файла содержит целое число  $k$  — количество событий, происходивших с предприятиями ( $1 \leq k \leq 100000$ ).

Последующие  $k$  строк содержат описания событий,  $i$ -я строка содержит два целых числа:  $e_i$  и  $v_i$  — тип события и номер предприятия, с которым оно произошло. Значение  $e_i = 1$  означает, что предприятие, которое после всех предыдущих событий является  $v_i$ -м по порядку, если считать с единицы от истока реки, обанкротилось, а значение  $e_i = 2$  означает, что это предприятие разделилось на два.

Гарантируется, что значение  $v_i$  не превышает текущее количество предприятий. Гарантируется, что если отрезок предприятия при банкротстве или разделении требуется поделить на две части, то он имеет длину большую или равную 2. Гарантируется, что если на реке осталось единственное предприятие, оно не банкротится.

## Формат выходных данных

Выходной файл должен содержать  $(k + 1)$  целых чисел, по одному в строке. Первая строка должна содержать исходную сумму квадратов длин отрезков реки, а каждая из последующих  $k$  строк — сумму квадратов длин отрезков реки после очередного события.

## Примеры

river.in	river.out
4 0	75
3 5 5 4	105
5	73
1 1	101
2 1	83
1 3	113
2 2	
1 3	

## Задача Н. Очередная

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
Ограничение по времени: 3 секунды  
Ограничение по памяти: 256 мегабайт

Изначально вам дана перестановка чисел от 1 до  $N$ . Вам поступают запросы двух видов:

- 1  $l_1$   $r_1$   $l_2$   $r_2$  для выполнения требуется взять два подмассива нашей перестановки с границами  $[l_1, r_1]$  и  $[l_2, r_2]$  и поменять местами содержимое подмассивов друг с другом.
- 2  $x$  найти место в перестановке, где находится число  $x$  и вывести 3 следующих за ним числа

### Формат входных данных

В первой строке находится два числа  $N$  и  $Q$  — размер перестановки и общее количество запросов ( $2 \leq N \leq 10000$ ,  $1 \leq Q \leq 200000$ ). Во второй строке — перестановка чисел от одного до  $N$ . В следующих  $Q$  строках описаны запросы в виде либо 1  $l_1$   $r_1$   $l_2$   $r_2$  ( $1 \leq l_1 \leq r_1 < l_2 \leq r_2 \leq N$ ,  $r_1 - l_1 = r_2 - l_2$ ) либо 2  $x$  ( $1 \leq x \leq N$ ).

### Формат выходных данных

Для каждого запроса второго типа выведите три числа — следующие числа за заданным, либо -1, если какого-то числа нет.

### Примеры

стандартный ввод	стандартный вывод
6 6	5 6 -1
1 2 3 4 5 6	5 3 1
2 4	2 6 -1
1 1 2 4 5	2 6 4
2 4	
2 1	
1 1 3 4 6	
2 1	

## Задача I. Декартово дерево

Имя входного файла: `tree.in`  
Имя выходного файла: `tree.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 256 мегабайт

Вам даны пары чисел  $(a_i, b_i)$ . Необходимо построить декартово дерево, такое что  $i$ -я вершина имеет ключи  $(a_i, b_i)$ , вершины с ключом  $a_i$  образуют бинарное дерево поиска, а вершины с ключом  $b_i$  образуют кучу.

### Формат входных данных

В первой строке записано число  $N$  — количество пар. Далее следует  $N$  ( $1 \leq N \leq 50\,000$ ) пар  $(a_i, b_i)$ . Для всех пар  $|a_i|, |b_i| \leq 30\,000$ .  $a_i \neq a_j$  и  $b_i \neq b_j$  для всех  $i \neq j$ .

### Формат выходных данных

Если декартово дерево с таким набором ключей построить возможно, выведите в первой строке «YES», в противном случае выведите «NO». В случае ответа «YES» выведите  $N$  строк, каждая из которых должна описывать вершину. Описание вершины состоит из трёх чисел: номера предка, номера левого сына и номера правого сына. Если у вершины отсутствует предок или какой либо из сыновей, выведите на его месте число 0.

Если подходящих деревьев несколько, выведите любое.

### Пример

tree.in	tree.out
7	YES
5 4	2 3 6
2 2	0 5 1
3 9	1 0 7
0 5	5 0 0
1 3	2 4 0
6 6	1 0 0
4 11	3 0 0

## Задача J. Легчайшая

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

Имеется  $N$  наборов чисел, которые изначально известны.

Поступают  $Q$  запросов вида  $x \ y \ k$ , для выполнения запроса нужно взять все числа из набора под номером  $x$ , которые не меньше числа  $k$ , и переместить их в набор номер  $y$ . После выполнения всех запросов необходимо вывести конечные состояния наборов.

### Формат входных данных

В первой строке дано два числа —  $N$  и  $Q$  ( $2 \leq N \leq 10^5$ ,  $1 \leq Q \leq 10^5$ ).

В последующих  $N$  строках заданы наборы. Каждый набор задается строкой вида: число  $k$ , за ним  $k$  чисел в неубывающем порядке. Суммарный размер наборов не превышает  $10^5$ . Все числа в наборах от 1 до  $10^6$ .

Далее в  $Q$  строках заданы запросы — по три числа  $x, y$  и  $k$  ( $1 \leq x, y \leq N$ ,  $x \neq y$ ,  $1 \leq k \leq 10^6$ ).

### Формат выходных данных

Выведите  $N$  строк — конечные состояния наборов, выводить числа набора следует в неубывающем порядке.

### Примеры

theeasiest.in	theeasiest.out
3 2	1 1
3 1 2 3	3 1 2 2
3 1 2 4	2 3 4
0	
1 2 2	
2 3 3	