

## 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

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

## Задача В. Двоичное дерево поиска 2

Имя входного файла: `bst2.in`  
Имя выходного файла: `bst2.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. Формат операций смотрите в предыдущей задаче. В каждой строке находится одна из следующих операций:

- `insert  $x$`  — добавить в дерево ключ  $x$ .
- `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.
- `kth  $k$`  — выведите  $k$ -ый по величине элемент (нумерация с единицы). Если такого не существует, то выведите «`none`».

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`, `kth`. Следуйте формату выходного файла из примера.

### Примеры

bst2.in	bst2.out
<code>insert 2</code>	<code>true</code>
<code>insert 5</code>	<code>false</code>
<code>insert 3</code>	<code>5</code>
<code>exists 2</code>	<code>3</code>
<code>exists 4</code>	<code>none</code>
<code>next 4</code>	<code>3</code>
<code>prev 4</code>	<code>2</code>
<code>delete 5</code>	<code>none</code>
<code>next 4</code>	
<code>prev 4</code>	
<code>kth 1</code>	
<code>kth 3</code>	

## Задача С. 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

<code>sum2.in</code>	<code>sum2.out</code>
6	3
+ 1	7
+ 3	
+ 3	
? 2 4	
+ 1	
? 2 4	

## 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	1 4 5 2 3 6
2 4	
3 5	
2 2	

## 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

<code>reverse.in</code>	<code>reverse.out</code>
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 зацикливаются  
в десятки раз эффективнее человека

Рекламный проспект OS Vista-N

Перед возвращением в штаб-квартиру корпорации Аазу и Скиву пришлось заполнить на местной таможене 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

<code>swapper.in</code>	<code>swapper.out</code>
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. Очередная

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

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 Н. Эх, дороги . . .

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

В многострадальном Тридесятом государстве опять готовится дорожная реформа. Впрочем, надо признать, дороги в этом государстве находятся в довольно плачевном состоянии. Так что реформа не повредит. Одна проблема — дорожникам не развернуться, поскольку в стране действует жесткий закон — из каждого города должно вести не более двух дорог. Все дороги в государстве двусторонние, то есть по ним разрешено движение в обоих направлениях (разумеется, разметка отсутствует). В результате реформы некоторые дороги будут строиться, а некоторые другие закрываться на бессрочный ремонт.

Петя работает диспетчером в службе грузоперевозок на дальние расстояния. В связи с предстоящими реформами, ему необходимо оперативно определять оптимальные маршруты между городами в условиях постоянно меняющейся дорожной ситуации. В силу большого количества пробок и сотрудников дорожной полиции в городах, критерием оптимальности маршрута считается количество промежуточных городов, которые необходимо проехать.

Помогите Пете по заданной последовательности сообщений об изменении структуры дорог и запросам об оптимальном способе проезда из одного города в другой, оперативно отвечать на запросы.

### Формат входных данных

В первой строке входного файла заданы числа  $n$  — количество городов,  $m$  — количество дорог в начале реформы и  $q$  — количество сообщений об изменении дорожной структуры и запросов ( $1 \leq n \leq 100\,000$ ,  $0 \leq m \leq 100\,000$ ,  $0 \leq q \leq 200\,000$ ). Следующие  $m$  строк содержат по два целых числа каждая — пары городов, соединенных дорогами перед реформой. Следующие  $q$  строк содержат по три элемента, разделенных пробелами. «+  $i j$ » означает строительство дороги от города  $i$  до города  $j$ , «-  $i j$ » означает закрытие дороги от города  $i$  до города  $j$ , «?  $i j$ » означает запрос об оптимальном пути между городами  $i$  и  $j$ .

### Формат выходных данных

На каждый запрос вида «?  $i j$ » выведите одно число — минимальное количество промежуточных городов на маршруте из города  $i$  в город  $j$ . Если проехать из  $i$  в  $j$  невозможно, выведите  $-1$ .

### Примеры

roads.in	roads.out
5 4 6	0
1 2	-1
2 3	1
1 3	2
4 5	
? 1 2	
? 1 5	
- 2 3	
? 2 3	
+ 2 4	
? 1 5	

## Задача I. Лягушачий хор

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

У лесных зверей не так уж и много развлечений, поэтому каждую среду они ходят на концерт местной группы лягушек. Квакающей компашкой дирижирует цапля с идеальным музыкальным слухом.

Вы попали на репетицию этого необычного ансамбля, где на болоте певицы стоят в ряд, а цапля перед ними. Стоит отметить, что каждая лягушка своего особенно зеленого цвета от 1 до  $n$ .

Как известно, лягушки любят петь высокие ноты, поэтому когда лягушка фальшивит (а это случается раз в минуту) цапля переставляет её в начало ряда, где ей придется петь самые низкие ноты.

Вы тщательно следили за репетицией ровно  $m$  минут и все записывали. Но ваш листок упал и на нем остались только  $m$  строк с информацией о том, что лягушка на месте  $p_i$  цвета  $c_i$  была переставлена в начало ряда в момент времени  $i$ . Эта информация написана ровно в том порядке, в котором цапля и переставляла лягушек.

В конце репетиции цапля поняла, что хоть лягушки и фальшивили, но первоначальное их расположение было лучше всего, поэтому она просит вас помочь ей и по имеющейся информации предоставить хотя бы одно возможное начальное расположение лягушек.

### Формат входных данных

В первой строке записаны целые числа  $n$  и  $m$  ( $1 \leq n, m \leq 3 \cdot 10^5$ ). В каждой из следующих  $m$  строк записана пара целых чисел. В  $i$ -й строке записаны целые числа  $c_i, p_i$  ( $1 \leq p_i, c_i \leq n$ ) — описание  $i$ -го перемещения лягушки. Обратите внимание, что перемещения заданы в том порядке, в котором их выполняла цапля. Вы не потеряли ни одного перемещения.

### Формат выходных данных

Если описанной перестановки не существует (вы ошиблись во время записи, увлекшись замечательной песней лягушечьего ансамбля), выведите  $-1$ . Иначе выведите  $n$  различных целых чисел, каждое от 1 до  $n$ :  $i$ -е число должно обозначать номер цвета лягушки, которая изначально стоит в ряду на позиции  $i$ .

Если существует несколько правильных ответов, нужно вывести лексикографически наименьший.

### Примеры

стандартный ввод	стандартный вывод
2 1 2 1	2 1
3 2 1 2 1 1	2 1 3
3 3 1 3 2 3 1 3	-1