Задача A. Persistent Array

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

Дан массив (вернее, первая, начальная его версия).

Нужно уметь отвечать на два запроса:

∘ $a_i[j] = x$ — создать из i-й версии новую, в которой j-й элемент равен x, а остальные элементы такие же, как в i-й версии.

∘ `get` $a_i[j]$ — сказать, чему равен j-й элемент в i-й версии.

Формат входных данных

Количество чисел в массиве N ($1 \leqslant N \leqslant 10^5$) и N элементов массива. Далее количество запросов M ($1 \leqslant M \leqslant 10^5$) и M запросов. Формат описания запросов можно посмотреть в примере. Если уже существует K версий, новая версия получает номер $K + 1$. И исходные, и новые элементы массива — целые числа от 0 до 10^9. Элементы в массиве нумеруются числами от 1 до N.

Формат выходных данных

На каждый запрос типа `get` вывести соответствующий элемент нужного массива.

Примеры

parray.in	parray.out
6 1 2 3 4 5 6 11 create 1 6 10 create 2 5 8 create 1 5 30 get 1 6 get 1 5 get 2 6 get 2 5 get 3 6 get 3 5 get 4 6 get 4 5	6 5 10 5 10 8 6 30

Задача B. Откат

Имя входного файла:	`rollback.in`
Имя выходного файла:	`rollback.out`
Ограничение по времени:	1.5 секунды
Ограничение по памяти:	256 мегабайта

Сергей работает системным администратором в очень крупной компании. Естественно, в круг его обязанностей входит резервное копирование информации, хранящейся на различных серверах и «откат» к предыдущей версии в случае возникновения проблем.

В данный момент Сергей борется с проблемой недостатка места для хранения информации для восстановления. Он решил перенести часть информации на новые сервера. К сожалению, если что-то случится во время переноса, он не сможет произвести откат, поэтому процедура переноса должна быть тщательно спланирована.

На данный момент у Сергея хранятся n точек восстановления различных серверов, пронумерованных от 1 до n. Точка восстановления с номером i позволяет произвести откат для сервера a_i. Сергей решил разбить перенос на этапы, при этом на каждом этапе в случае возникновения проблем будут доступны точки восстановления с номерами $l, l+1, \ldots, r$ для некоторых l и r.

Для того, чтобы спланировать перенос данных оптимальным образом, Сергею необходимо научиться отвечать на запросы: для заданного l, при каком минимальном r в процессе переноса будут доступны точки восстановления не менее чем k различных серверов.

Помогите Сергею.

Формат входных данных

Первая строка входного файла содержит два целых числа n и m, разделенные пробелами — количество точек восстановления и количество серверов ($1 \leqslant n, m \leqslant 100\,000$). Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ — номера серверов, которым соответствуют точки восстановления ($1 \leqslant a_i \leqslant m$).

Третья строка входного файла содержит q — количество запросов, которые необходимо обработать ($1 \leqslant q \leqslant 100\,000$). В процессе обработки запросов необходимо поддерживать число p, исходно оно равно 0. Каждый запрос задается парой чисел x_i и y_i, используйте их для получения данных запроса следующим образом: $l_i = ((x_i + p) \bmod n) + 1$, $k_i = ((y_i + p) \bmod m) + 1$ ($1 \leqslant l_i, x_i \leqslant n$, $1 \leqslant k_i, y_i \leqslant m$). Пусть ответ на i-й запрос равен r. После выполнения этого запроса, следует присвоить p значение r.

Формат выходных данных

На каждый запрос выведите одно число — искомое минимальное r, либо 0, если такого r не существует.

Примеры

rollback.in	rollback.out
7 3 1 2 1 3 1 2 1 4 7 3 7 1 7 1 2 2	1 4 0 6

Задача C. Сумма различных

Имя входного файла:	`distinct.in`
Имя выходного файла:	`distinct.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мегабайт

Есть набор натуральных чисел — a_1, a_2, ..., a_n. Необходимо делать две операции — обновлять какое-то число и запрашивать сумму различных чисел на отрезке $a_{i..j}$. К примеру, если $a_4 = 5$, $a_5 = 13$, $a_6 = 5$, то результат запроса на отрезке $a_{4...6}$ — 18, потому что множество чисел, которое встречается на данном отрезке — $\{5, 13\}$.

Формат входных данных

В первой строке дано натуральное n ($1 \leqslant n \leqslant 50\,000$). Во второй строке даны n натуральных чисел — исходные значения a_i ($1 \leqslant a_i \leqslant 10^9$). В третьей строке дано натуральное m — количество операций, которые необходимо выполнить ($1 \leqslant m \leqslant 10^5$). Далее в m строках заданы описания операций. Если очередная операция — обновление, то соответствующее описание имеет вид "`U` u v" ($1 \leqslant u \leqslant n$, $1 \leqslant v \leqslant 10^9$), означающее, что после выполнения операции a_u должно равняться v. Если очередная операция — запрос, то соответствующее описание имеет вид "`Q` l r" ($1 \leqslant l \leqslant r \leqslant n$). В качестве ответа на запрос необходимо вывести сумму различных чисел на отрезке $a_{l..r}$.

Формат выходных данных

Для каждого запроса выведите ответ на него на отдельной строке.

Примеры

distinct.in	distinct.out
5 1 2 4 2 3 3 Q 2 4 U 4 7 Q 2 4	6 13

Задача D. Прямоугольники-2

Имя входного файла:	`rect2.in`
Имя выходного файла:	`rect2.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

В связи с большим количеством покупок дачных участков, два больших, но от этого не менее гордых государства (назовем их условно «первое» и «второе»), установили ряд соглашений, касающихся участков земли около их границы.

Чтобы лучше понять нововведения, рассмотрим границу между этими государствами на карте, которая висит на стене так, что север находится вверху. Введём ортонормированную систему координат, в которой ось OX направлена с запада на восток, а OY — с юга на север. Рассмотрим n равных по величине отрезков на оси OX, i-ый из этих отрезков имеет координаты $[i-1, i]$. Каждому из них сопоставим вертикальную полосу, образованную всеми возможными прямыми, параллельными OY и проходящими через сам отрезок. Теперь, чтобы разделить государства, рассмотрим придуманную систему уровней, основанную на введённых вертикальных полосах. Для каждой полосы определим её уровень, который задается некоторым числом z_i. Точки, принадлежащие вертикальной полосе соответствующего отрезка, лежащие выше уровня, принадлежат первому государству, а ниже — второму.

Когда коренной житель одного из государств хочет купить прямоугольный участок земли со сторонами, параллельными осям координат (участки другого вида никого не интересуют), он может это сделать, если его родное государство доминирует на выбранном участке. Это происходит, если государство доминирует на большей, чем другое государство, части вертикальных полос, образованных отрезками на оси OX. Для вертикальных полос свойство преобладания определяется следующим образом: если площадь участка на этой полосе, принадлежащего одному из государств, строго больше площади, принадлежащей другому, то первое из них доминирует на этой полосе.

Вас просят написать программу, которая могла бы определять государство, доминирующее на участке, а также изменять границу между государствами.

Формат входных данных

В первой строке входного файла записано n — количество отрезков, на которые разделена ось OX ($1 \leqslant n \leqslant 50\,000$). Во второй строке — n чисел z_i, определяющих границу между государствами ($0 \leqslant z_i \leqslant 10^9$). В третьей строке задано m — число запросов к Вашей программе ($1 \leqslant m \leqslant 100\,000$). Далее следует m строк с запросами. Каждый запрос имеет вид «`C` x z» или «`Q` x_1 y_1 x_2 y_2». Запрос вида «`C` x z» означает, что уровень вертикальной полосы номер x стал равным z ($1 \leqslant x \leqslant n, 1 \leqslant z \leqslant 10^9$). Запрос вида «`Q` x_1 y_1 x_2 y_2» ($1 \leqslant x_1 \leqslant x_2 \leqslant n$, $0 \leqslant y_1 < y_2 \leqslant 10^9$) означает, что требуется вывести государство, доминирующее на участке, левой границей которого является вертикальная полоса номер x_1 (включительно), правой границей — вертикальная полоса номер x_2 (включительно), а с юга и с севера участок ограничен координатами y_1 и y_2 соответственно. Все числа во входном файле целые.

Формат выходных данных

Для каждого запроса вида «`Q` x_1 y_1 x_2 y_2» выведите 1, если на этом участке доминирует первое государство, 2, если второе, и 0, если ни у одного из государств преимущества нет.

Примеры

rect2.in	rect2.out
2 0 0 5 Q 1 0 2 2 C 1 1 Q 1 0 2 2 C 2 1 Q 1 0 2 2	1 1 0

Задача E. Не курить!

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

Вася — хороший парень. Но у него есть плохая привычка — он курит. Все то время, сколько Петя дружит с Васей, он пытается отучить его от этого. Но ему это так и не удалось, потому что Вася не хочет бросать курить.

Недавно Петя придумал способ, как отучить своего друга от курения. Вася — неряха, поэтому его сигареты не лежат в пачке, а разбросаны по огромному столу. Петя хочет брать несколько сигарет в день незаметно для Васи. Вася не заметит пропажи сигарет, если в день будет пропадать не более одной сигареты. Кроме того, Петя должен брать только ту сигарету, которая пересекается с какой-нибудь другой сигаретой на столе. Помогите Пете узнать, сможет ли он начать реализацию своего плана.

Формат входных данных

Сигарета представляется как отрезок прямой. В первой строке входного файла записано число N ($1 \leqslant N \leqslant 125\,000$) — количество сигарет на Васином столе. Следующие N строк содержат описания сигарет: $(i+1)$-я строка содержит координаты концов i-й сигареты — целые числа x_1, y_1, x_2, y_2 ($-10\,000 \leqslant x_1, y_1, x_2, y_2 \leqslant 10\,000$).

Формат выходных данных

В первой строке выходного файла выведите слово "`YES`", если Пете удастся начать реализацию своего плана. Вторая строка должна содержать числа i и j: i — номер сигареты, которую должен взять Петя, j — номер сигареты, с которой она пересекается.

Если Петя не сможет взять ни одной сигареты, выведите в единственной строке выходного файла "`NO`".

Примеры

smoking.in	smoking.out
2 0 0 2 2 0 2 2 0	YES 2 1