Задача A. Друзья и последовательности

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

Майк и !Майк соперничают еще со школьных лет, они противоположны во всем что делают, кроме программирования. Сегодня у них возникла проблема, которую сами друзья сами решить не могут, но вместе с вами — кто знает?

Каждый из них знает две последовательности n чисел a и b. По запросу в виде пары целых чисел (l, r) Майк может сразу сообщить значение $\max\limits_{i=l}^{r} a_i$, а !Майк — значение $\min\limits_{i=l}^{r} b_i$.

Предположим, что робот задает им каждый из возможных различных запросов в виде пары целых чисел (l, r) $(1 \leqslant l \leqslant r \leqslant n)$ (то есть он сделает ровно $n(n+1)/2$ запросов) и считает, сколько раз их ответы на один и тот же запрос совпадают, то есть для скольких пар выполняется $\max\limits_{i=l}^{r} a_i = \min\limits_{i=l}^{r} b_i$.

Сколько случаев совпадения посчитает робот?

Формат входных данных

В первой строке содержится единственное целое число n $(1 \leqslant n \leqslant 200\,000)$.

Во второй строке содержатся n целых чисел $a_1, a_2, \ldots, a_n$ $(-10^9 \leqslant a_i \leqslant 10^9)$ — элементы последовательности a.

В третьей строке содержатся n целых чисел $b_1, b_2, \ldots, b_n$ $(-10^9 \leqslant b_i \leqslant 10^9)$ — элементы последовательности b.

Формат выходных данных

Выведите одно целое число — количество совпадений ответов, которые посчитает робот, то есть для скольких пар выполняется $\max\limits_{i=l}^{r} a_i = \min\limits_{i=l}^{r} b_i$.

Примеры

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

Задача B. Лунки

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

Маленький Петя очень любит играть. Больше всего он любит играть в игру «Лунки». Это игра для одного игрока со следующими правилами:

Есть N лунок, расположенных в ряд, пронумерованных слева направо числами от 1 до N. У каждой лунки изначально установлена своя сила выброса (у лунки с номером i она равна a_i). Если вбросить шарик в лунку i, то он тут же вылетит из нее и попадет в лунку $i + a_i$, после чего он опять вылелтит и т.д.. Если же лунки с таким номером нету, то он просто вылетит за край ряда. На каждом из M ходов игрок выбирает одно из двух действий:

- Установить силу выброса лунки a равной b.
- Вбросить шарик в лунку a и посчитать количество прыжков шарика, прежде чем он вылетит за край ряда, а так же записать номер лунки, после выпрыгивания из которой шарик вылетел за край.

У Пети есть некоторые проблемы с математикой, поэтому, как Вы уже догадались, именно Вам предстоит произвести все подсчеты.

Формат входных данных

Первая строка содержит два числа N и M ($1 \leqslant N \leqslant 10^5$, $1 \leqslant M \leqslant 10^5$) — количество лунок в ряду и количество ходов. Следующая строка содержит N целых положительных чисел, не превышающих N — изначальные силы выброса лунок. Следующие M строк задают ходы, сделанные Петей. Каждая строка может быть двух типов:

- 0 a b
- 1 a

Тут, первый тип означает что требуется установить силу выброса лунки a равной b, а второй означает что требуется вбросить мячик в лунку с номером a. Числа a и b — целые положительные и не превышают N.

Формат выходных данных

Для каждого хода второго типа (задающего вбрасывание шарика) в порядке следования во входном файле выведите два числа через пробел в отдельной строке — номер последней лунки перед вылетом шарика за край и количество прыжков.

Примеры

holes.in	holes.out
8 5 1 1 1 1 1 2 8 2 1 1 0 1 3 1 1 0 3 4 1 2	8 7 8 5 7 3

Задача D. Фаброзавры-дизайнеры

Имя входного файла:	`fabro.in`
Имя выходного файла:	`fabro.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мегабайта

Фаброзавры известны своим тонким художественным вкусом и увлечением ландшафтным дизайном. Они живут около очень живописной реки и то и дело перестраивают тропинку, идущую вдоль реки: либо насыпают дополнительной земли, либо срывают то, что есть. Для того, чтобы упростить эти работы, они поделили всю тропинку на горизонтальные участки, пронумерованные от 1 до N, и их переделки устроены всегда одинаково: они выбирают часть дороги от L-го до R-го участка (включительно) и изменяют (увеличивают или уменьшают) высоту на всех этих участках на одну и ту же величину (если до начала переделки высоты были разными, то и после переделки они останутся разными).

Поскольку, как уже говорилось, у фаброзавров тонкий художественный вкус, каждый из них считает, что их река лучше всего выглядит с определенной высоты. Поэтому им хочется знать, есть ли поблизости от их дома место на тропинке, где высота на их взгляд оптимальна. Помогите им в этом разобраться.

Формат входных данных

Первая строка входного файла содержит два числа N и M — длину дороги и количество запросов соответственно ($1 \leqslant N, M \leqslant 10^5$). На второй строке содержатся N чисел, разделенных пробелами — начальные высоты соответствующих частей дороги; высоты не превосходят 10^4 по модулю. В следующих M строках содержатся запросы по одному на строке.

Запрос `+` L R X означает, что высоту частей дороги от L-й до R-й (включительно) нужно изменить на X. При этом $1 \leqslant L \leqslant R \leqslant N$, а $|X| \leqslant 10^4$.

Запрос `?` L R X означает, что нужно проверить, есть ли между L-м и R-м участками (включая эти участки) участок, где дорога проходит точно на высоте X. Гарантируется, что $1 \leqslant L \leqslant R \leqslant N$, а $|X| \leqslant 10^9$.

Формат выходных данных

На каждый запрос второго типа нужно вывести в выходной файл на отдельной строке одно слово «`YES`» (без кавычек), если нужный участок существует, и «`NO`» в противном случае.

Примеры

fabro.in	fabro.out
10 5 0 1 1 3 3 3 2 0 0 1 ? 3 5 2 + 1 4 1 ? 3 5 2 + 7 10 2 ? 9 10 3	NO YES YES

Задача E. Озера и лягушки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В Берендеевском лесу живет много лягушек. Кроме того, этот лес славится тем, что в нем находится n небольших озер, в котором и живут квакушки. Эти озера соединены между собой m ручейками. Чюваки, живущие неподалеку, уже обошли их все и точно знают, что ручеек не может течь из одного озера в него же и что между любыми двумя озерами течет не более одного ручейка.

В одну прекрасную среду вы пришли к этим озерам и заметили, что в каждом озере лягушки имеют собственный цвет. Понаблюдав еще немного, вы узнали, что иногда лягушки в каком-то из озер меняют цвет.

Вы решили посидеть еще немного с лягушками и поиграть в игру. А именно каждую минуту вы либо видите, что в озере с номером v лягушки поменяли свой цвет на цвет c, либо вы хотите посчитать для озера с номером v количество различных цветов лягушек в соседних озера (соседними озерами вы называете те озера, которые соединены ручейком).

В конце игры у вас остался листочек с изначальным цветом лягушек в n озерах, а также q действий которые происходили в тот день.

Вам хочется еще поиграть в эту увлекательную игру, поэтому для простоты вы хотите написать программу, которая будет считать количество различных цветов лягушек в соседних озерах за вас.

Формат входных данных

В первой строке даны три целых числа n, m и q $\left(1 \leqslant n \leqslant 10^5, 1 \leqslant m, q \leqslant 2 \cdot 10^5\right)$.

Во второй строке через пробел даны n целых чисел $1 \leqslant c_1, \ldots, c_n \leqslant n$ – изначальные цвета лягушек в озерах.

В следующих m строках заданы по два целых числа u и v – номера озер, которые соединены очередным ручейком $(1 \leqslant u, v \leqslant n, u \neq v)$.

В следующих q строках заданы запросы. Каждый запрос имеет один из двух типов:

- $1\ v\ c$ — поменять цвет лягушек в озере с номером v на цвет c $(1 \leqslant v, c \leqslant n)$;
- $2\ v$ — вычислить количество различных цветов лягушек среди соседей озера под номером v $(1 \leqslant v \leqslant n)$.

Формат выходных данных

Для каждого запроса второго типа выведите в отдельной строке ответ на него.

Пример

стандартный ввод	стандартный вывод
4 3 3 1 2 3 4 1 2 1 3 1 4 2 1 1 4 3 2 1	3 2

Задача F. Жесть

Имя входного файла:	`sqrtrev.in`
Имя выходного файла:	`sqrtrev.out`
Ограничение по времени:	10 секунд
Ограничение по памяти:	256 мегабайт

Дам массив из N чисел. Нужно уметь обрабатывать 3 типа запросов:

- `get(L, R, x)` — сказать, сколько элементов отрезка массива $[L..R]$ не меньше x.
- `set(L, R, x)` — присвоить всем элементам массива на отрезке $[L..R]$ значение x.
- `reverse(L, R)` — перевернуть отрезок массива $[L..R]$.

Формат входных данных

Число N ($1 \leqslant N \leqslant 10^5$) и массив из N чисел. Далее число запросов M ($1 \leqslant M \leqslant 10^5$) и M запросов. Формат описания запросов предлагается понять из примера. Для всех отрезков верно $1 \leqslant L \leqslant R \leqslant N$. Исходные числа в массиве и числа x в запросах — целые от 0 до 10^9.

Формат выходных данных

Для каждого запроса типа `get` нужно вывести ответ.

Примеры

sqrtrev.in	sqrtrev.out
5 1 2 3 4 5 6 get 1 5 3 set 2 4 2 get 1 5 3 reverse 1 2 get 2 5 2 get 1 1 2	3 1 3 1

Задача G. Варенье

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

Малыш и Карлсон решили пойти на прогулку. Они знают, что прогулка будет совсем скучной, если перед ней не опустошить несколько банок варенья.

Малыш достал из кладовки N банок варенья и выставил их в ряд. В банке номер i содержится ровно a_i грамм варенья. Карлсон немного подумал и решил, что в некоторых банках недостаточно варенья, и что в банке номер i должно быть хотя бы b_i грамм варенья.

Выходить из этой ситуации Карлсон хочет в M этапов. На каждом этапе он выбирает числа l, r, x и y, а затем выполняет следующие операции: в банку номер l он добавляет x грамм варенья, в банку номер $l+1$ — $x+y$ грамм варенья, в банку номер $l+2$ — $x+2\cdot y$, и так далее. В банку номер r наш герой добавит $x+y\cdot(r-l)$ грамм варенья.

Малышу хочется определить для каждой банки i наименьший номер операции, после которой в ней станет хотя бы b_i грамм варенья. Помогите Малышу: найдите соответствующее число для каждой банки.

Формат входных данных

В первой строке входного файла задано одно число N ($1 \leqslant n \leqslant 10^5$) — количество банок. Во второй строке заданы N чисел a_i ($0 \leqslant a_i \leqslant 2\cdot 10^9$) — изначальное количество варенья в банке номер i. В третьей строке заданы N чисел b_i ($0 \leqslant b_i \leqslant 2\cdot 10^9$) — минимальное количество варенья, которое должно быть в банке номер i.

В четвертой строке задано M ($0 \leqslant M \leqslant 10^5$) — число этапов добавления варенья в банки, которые выполнит Карлсон. В следующих M строках описаны сами этапы в хронологическом порядке. Каждый этап задан четырьмя числами l, r, x и y ($1 \leqslant l \leqslant r \leqslant N$, $0 \leqslant x, y \leqslant 3\cdot 10^5$).

Формат выходных данных

Выведите N чисел в одной строке, разделенные пробелом. Число номер i должно быть равно нулю, если в банке номер i изначально было достаточно варенья, номеру этапа, после которого в ней станет хотя бы b_i варенья, или `-1`, если даже после выполнения всех этапов, в этой банке будет недостаточно варенья. Этапы нумеруются с единицы.

Примеры

jam.in	jam.out
5 5 4 4 2 1 7 7 4 7 7 3 1 2 2 0 2 5 1 1 3 4 2 2	1 2 0 3 -1

Задача I. Разреженные таблицы

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

Дан массив из n чисел. Требуется написать программу, которая будет отвечать на запросы следующего вида: найти минимум на отрезке между u и v включительно.

Формат входных данных

В первой строке входного файла даны три натуральных числа n, m ($1 \leqslant n \leqslant 10^5$, $1 \leqslant m \leqslant 10^7$) и a_1 ($0 \leqslant a_1 < 16\,714\,589$) — количество элементов в массиве, количество запросов и первый элемент массива соответственно. Вторая строка содержит два натуральных числа u_1 и v_1 ($1 \leqslant u_1, v_1 \leqslant n$) — первый запрос.

Элементы $a_2, a_3, \ldots, a_n$ задаются следующей формулой:

$$a_{i+1} = (23 \cdot a_i + 21563) \bmod 16714589.$$

Например, при $n = 10$, $a_1 = 12345$ получается следующий массив: a = (12345, 305498, 7048017, 11694653, 1565158, 2591019, 9471233, 570265, 13137658, 1325095).

Запросы генерируются следующим образом:

$$u_{i+1} = \big((17 \cdot u_i + 751 + ans_i + 2i) \bmod n\big) + 1,$$
$$v_{i+1} = \big((13 \cdot v_i + 593 + ans_i + 5i) \bmod n\big) + 1,$$

где ans_i — ответ на запрос номер i.

Обратите внимание, что u_i может быть больше, чем v_i.

Формат выходных данных

В выходной файл выведите u_m, v_m и ans_m (последний запрос и ответ на него).

Примеры

sparse.in	sparse.out
10 8 12345 3 9	5 3 1565158

Замечание

Пояснение к тесту из примера: запросы и результаты.

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
12345	305498	7048017	11694653	1565158	2591019	9471233	570265	13137658	1325095

#	u	v	ans
1	3	9	570265
2	10	1	12345
3	1	2	12345
4	10	10	1325095
5	5	9	570265
6	2	1	12345
7	3	2	305498
8	5	3	1565158

Задача J. Сумма на отрезке

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

Дан массив из N элементов, нужно научиться находить сумму чисел на отрезке.

Формат входных данных

Первая строка входного файла содержит два целых числа N и K — количество чисел в массиве и количество запросов ($1 \leqslant N \leqslant 100\,000$, $0 \leqslant K \leqslant 100\,000$). Следующие K строк содержат следующие запросы:

1. `A i x` — присвоить i-му элементу массива значение x ($1 \leqslant i \leqslant n$, $0 \leqslant x \leqslant 10^9$);
2. `Q l r` — найти сумму чисел в массиве на позициях от l до r ($1 \leqslant l \leqslant r \leqslant n$).

Изначально в массиве живут нули.

Формат выходных данных

На каждый запрос вида `Q l r` нужно вывести единственное число — сумму на отрезке.

Примеры

sum.in	sum.out
5 9 A 2 2 A 3 1 A 4 2 Q 1 1 Q 2 2 Q 3 3 Q 4 4 Q 5 5 Q 1 5	0 2 1 2 0 5

Замечание

TL для Python 4 секунды