Задача A. Сумма простая

Имя входного файла:	`sum0.in`
Имя выходного файла:	`sum0.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вам нужно научиться отвечать на запрос «сумма чисел на отрезке».

Массив не меняется. Запросов много. Отвечать на каждый запрос следует за $\mathcal{O}(1)$.

Формат входных данных

Размер массива — n и числа x, y, a_0, порождающие массив a: $a_i = (x \cdot a_{i-1} + y) \bmod 2^{16}$

Далее следует количество запросов m и числа z, t, b_0, порождающие массив b: $b_i = (z \cdot b_{i-1} + t) \bmod 2^{30}$.

Массив c строится следующим образом: $c_i = b_i \bmod n$.

Запросы: i-й из них — найти сумму на отрезке от $\min(c_{2i}, c_{2i+1})$ до $\max(c_{2i}, c_{2i+1})$ в массиве a.

Ограничения: $1 \leqslant n \leqslant 10^7$, $0 \leqslant m \leqslant 10^7$. Все числа целые от 0 до 2^{16}. t может быть равно -1.

Формат выходных данных

Выведите сумму всех сумм.

Пример

sum0.in	sum0.out
3 1 2 3 3 1 -1 4	23

Замечание

$a = \{3, 5, 7\}, b = \{4, 3, 2, 1, 0, 2^{30} - 1\}, c = \{1, 0, 2, 1, 0, 0\}$,
запросы $= \{[0, 1], [1, 2], [0, 0]\}$, суммы $= \{8, 12, 3\}$.

Задача B. RSQ

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

Формат входных данных

В первой строке находится число n — размер массива. ($1 \leqslant n \leqslant 500,000$) Во второй строке находится n чисел a_i — элементы массива. Далее содержится описание операций, их количество не превышает $1,000,000$. В каждой строке находится одна из следующих операций:

- `set` i x — установить $a[i]$ в x.
- `sum` i j — вывести значение суммы элементов в массиве на отрезке с i по j, гарантируется, что $(1 \leqslant i \leqslant j \leqslant n)$.

Все числа во входном файле и результаты выполнения всех операций не превышают по модулю 10^{18}.

Формат выходных данных

Выведите последовательно результат выполнения всех операций `sum`. Следуйте формату выходного файла из примера.

Пример

rsq.in	rsq.out
5 1 2 3 4 5 sum 2 5 sum 1 5 sum 1 4 sum 2 4 set 1 10 set 2 3 set 5 2 sum 2 5 sum 1 5 sum 1 4 sum 2 4	14 15 10 9 12 22 20 10

Задача C. RMQ2

Имя входного файла:	`rmq2.in`
Имя выходного файла:	`rmq2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Формат входных данных

В первой строке находится число n — размер массива. ($1 \leqslant n \leqslant 10^5$) Во второй строке находится n чисел a_i — элементы массива. Далее содержится описание операций, их количество не превышает $2 \cdot 10^5$. В каждой строке находится одна из следующих операций:

- `set` i j x — установить все $a[k]$, $i \leqslant k \leqslant j$ в x.
- `add` i j x — увеличить все $a[k]$, $i \leqslant k \leqslant j$ на x.
- `min` i j — вывести значение минимального элемента в массиве на отрезке с i по j, гарантируется, что ($1 \leqslant i \leqslant j \leqslant n$).

Все числа во входном файле и результаты выполнения всех операций не превышают по модулю 10^{18}.

Формат выходных данных

Выведите последовательно результат выполнения всех операций `min`. Следуйте формату выходного файла из примера.

Пример

rmq2.in	rmq2.out
5 1 2 3 4 5 min 2 5 min 1 5 min 1 4 min 2 4 set 1 3 10 add 2 4 4 min 2 5 min 1 5 min 1 4 min 2 4	2 1 1 2 5 5 8 8

Задача D. Художник

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

Итальянский художник-абстракционист Ф. Мандарино увлекся рисованием одномерных черно-белых картин. Он пытается найти оптимальное местоположение и количество черных участков картины. Для этого он проводит на прямой белые и черные отрезки, и после каждой из таких операций хочет знать количество черных отрезков на получившейся картине и их суммарную длину.

Изначально прямая — белая. Ваша задача — написать программу, которая после каждой из таких операций выводит в выходной файл интересующие художника данные.

Формат входных данных

В первой строке входного файла содержится общее количество нарисованных отрезков ($1 \leqslant n \leqslant 100\,000$). В последующих n строках содержится описание операций. Каждая операция описывается строкой вида $c\ x\ l$, где c — цвет отрезка (`W` для белых отрезков, `B` для черных), а сам отрезок имеет вид $[x; x + l]$, причем координаты обоих концов — целые числа, не превосходящие по модулю 500 000. Длина задается положительным целым числом.

Формат выходных данных

После выполнения каждой из операций необходимо вывести в выходной файл на отдельной строке количество черных отрезков на картине и их суммарную длину, разделенные одним пробелом.

Пример

painter.in	painter.out
7 W 2 3 B 2 2 B 4 2 B 3 2 B 7 2 W 3 1 W 0 10	0 0 1 2 1 4 1 4 2 6 3 5 0 0

Задача E. Криптография

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

Задано n матриц $A_1, A_2, \ldots, A_n$ размера 2×2. Необходимо для нескольких запросов вычислить произведение матриц $A_i, A_{i+1}, \ldots, A_j$. Все вычисления производятся по модулю r.

Формат входных данных

Первая строка входного файла содержит числа r ($1 \leqslant r \leqslant 10\,000$), n ($1 \leqslant n \leqslant 200\,000$) и m ($1 \leqslant m \leqslant 200\,000$). Следующие n блоков по две строки содержащие по два числа в строке — описания матриц. Затем следуют m пар целых чисел от 1 до n, запросы на произведение на отрезке.

Формат выходных данных

Выведите m блоков по две строки,по два числа в каждой — произведения на отрезках. Разделяйте блоки пустой строкой. Все вычисления производятся по модулю r

Пример

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

Задача F. Разреженные таблицы

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

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

Задача G. Окна

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

На экране расположены прямоугольные окна, каким-то образом перекрывающиеся (со сторонами, параллельными осям координат). Вам необходимо найти точку, которая покрыта наибольшим числом из них.

Формат входных данных

В первой строке входного файла записано число окон n ($1 \leqslant n \leqslant 50000$). Следующие n строк содержат координаты окон $x_{(1,i)}$ $y_{(1,i)}$ $x_{(2,i)}$ $y_{(2,i)}$, где $(x_{(1,i)}, y_{(1,i)})$ — координаты левого верхнего угла i-го окна, а $(x_{(2,i)}, y_{(2,i)})$ — правого нижнего (на экране компьютера y растет сверху вниз, а x — слева направо). Все координаты — целые числа, по модулю не превосходящие $2 \cdot 10^5$.

Формат выходных данных

В первой строке выходного файла выведите максимальное число окон, покрывающих какую-либо из точек в данной конфигурации. Во второй строке выведите два целых числа, разделенные пробелом — координаты точки, покрытой максимальным числом окон. Окна считаются замкнутыми, т.е. покрывающими свои граничные точки.

Примеры

стандартный ввод	стандартный вывод
2 0 0 3 3 1 1 4 4	2 1 3
1 0 0 1 1	1 0 1

Задача H. RMQ наоборот

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

Рассмотрим массив $a[1..n]$. Пусть $Q(i, j)$ — ответ на запрос о нахождении минимума среди чисел $a[i], \ldots, a[j]$. Вам даны несколько запросов и ответы на них. Восстановите исходный массив.

Формат входных данных

Первая строка входного файла содержит число n — размер массива, и m — число запросов ($1 \leqslant n, m \leqslant 100\,000$). Следующие m строк содержат по три целых числа i, j и q, означающих, что $Q(i, j) = q$ ($1 \leqslant i \leqslant j \leqslant n$, $-2^{31} \leqslant q \leqslant 2^{31} - 1$).

Формат выходных данных

Если искомого массива не существует, выведите строку «`inconsistent`».

В противном случае в первую строку выходного файла выведите «`consistent`». Во вторую строку выходного файла выведите элементы массива. Элементами массива должны быть целые числа в интервале от -2^{31} до $2^{31} - 1$ включительно. Если решений несколько, выведите любое.

Примеры

rmq.in	rmq.out
3 2 1 2 1 2 3 2	consistent 1 2 2
3 3 1 2 1 1 1 2 2 3 2	inconsistent

Задача I. Горы

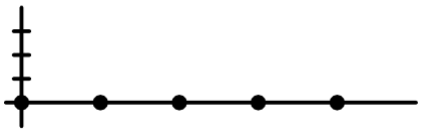
Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В парке развлечений «Ай-ой-ай» открылся новейший аттракцион: польские горки. Трек состоит из n рельс, присоединенных одна к концу другой. Начало первой рельсы находится на высоте 0. Оператор Петя может конфигурировать аттракцион, изменяя по своему желанию подъём нескольких последовательных рельс. При этом подъём всех остальных рельс не изменяется. При каждом изменении конфигурации рельс положение следующих за изменяемыми подбирается таким образом, чтобы весь трек оставался связным.

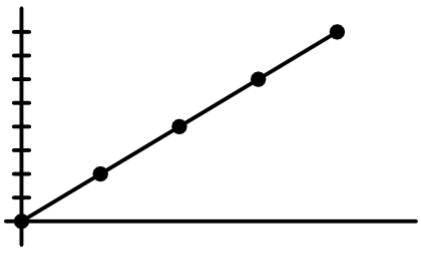
Каждый запуск вагонетки осуществляется с энергией, достаточной для достижения высоты h. Это значит, что вагонетка будет двигаться до тех пор, пока высота не превысит h, либо пока не закончится трек.

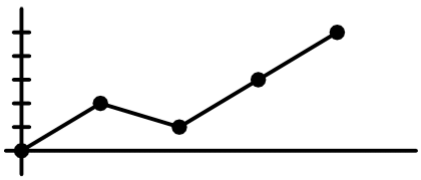
По записям о всех изменениях конфигурации рельс и временах запусков вагонетки для каждого запуска определите, сколько рельс вагонетка проедет до остановки.

Трек можно представить как последовательность n подъемов d_i, по одному на рельс. Изначально рельсы горизонтальны, то есть $d_i = 0$ для всех i.



Каждое изменение конфигурации определяется числами a, b и D: все рельсы с a-й по b-ю включительно после этого действия имеют подъем, равный D.





Каждый запуск вагонетки определяется единственным целым числом h — максимальной высотой, на которую способна подняться вагонетка.

Формат входных данных

В первой строке записано целое число n ($1 \leqslant n \leqslant 10^9$) — число рельс. Следующие строки содержат запросы трех видов:

- I a b D — изменение конфигурации. Рельсы с a-й по b-ю включительно после выполнения запроса имеют подъем, равный D.
- Q h — запуск вагонетки. Требуется найти число рельс, которое проедет вагонетка, которая способна подняться на высоту h.

- E — конец ввода. Этот запрос встретится ровно один раз в конце файла.

В любой момент времени высота любой точки трека лежит в промежутке от 0 до 10^9. Во вводе не более 100 000 строк.

Формат выходных данных

Для каждого запроса Q выведите единственное целое число — количество рельс, которое проедет вагонетка.

Пример

стандартный ввод	стандартный вывод
4 Q 1 I 1 4 2 Q 3 Q 1 I 2 2 -1 Q 3 E	4 1 0 3

Задача J. Великая Китайская Стена

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

В этой задаче мы проследим альтернативную историю Великой Китайской Стены.

Великая Китайская Стена состоит из n метровых участков, пронумерованных по порядку целыми числами от 1 до n. Каждый участок характеризуется своей высотой в метрах — целым неотрицательным числом. До начала нашей истории Стена ещё не построена, поэтому высота каждого участка равна нулю.

Происходят события двух видов.

1. *Укрепление Стены* (запись: «`defend` a b c»). Император вызывает к себе вассалов из приграничных провинций и велит им сделать так, чтобы промежуток Стены, охватывающий участки от a до b включительно, имел высоту не менее c метров. Это значит, что все участки меньшей высоты на этом промежутке нужно достроить до высоты c, а остальные оставить нетронутыми. Приказ императора выполняется немедленно, то есть до наступления следующего события.

2. *Нападение варваров* (запись: «`attack` d e»). Варвары подходят к Стене снаружи и занимают позиции напротив промежутка Стены, охватывающего участки от d до e включительно. После этого они находят такой участок на этом промежутке, у которого высота как можно меньше, и пытаются через него проникнуть на территорию Китая. Нападение также происходит немедленно, до наступления следующего события.

Для восстановления достоверной альтернативно-исторической картины не хватает одного: для каждого нападения варваров указать минимальную высоту Стены на соответствующем промежутке, а также какой-нибудь участок из этого промежутка с такой высотой. По заданной последовательности событий найдите эти числа.

Формат входных данных

В первой строке заданы через пробел два целых числа n и m — длина Стены в метрах и количество событий соответственно ($1 \leqslant n \leqslant 10^6$, $0 \leqslant m \leqslant 10^5$). В следующих m строках описаны события в порядке их следования. Если событие описывает укрепление Стены, оно задано в форме «`defend` a b c» ($1 \leqslant a \leqslant b \leqslant n$, $1 \leqslant c \leqslant 10^7$). Если же событие описывает нападение варваров, оно задано в форме «`attack` d e» ($1 \leqslant d \leqslant e \leqslant n$).

Формат выходных данных

В ответ на каждое нападение варваров выведите строку, содержащую два числа, разделённые пробелом. Первое из этих чисел — минимальная высота Стены на соответствующем промежутке. Второе — номер любого метрового участка Стены на этом промежутке, имеющего такую высоту.

Пример

стандартный ввод	стандартный вывод
5 4 defend 1 3 10 attack 1 4 attack 2 3 attack 1 2	0 4 10 2 10 1

Задача K. Перестановка

Имя входного файла:	`permutation.in`
Имя выходного файла:	`permutation.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт
Отображение результатов:	только полное решение подзадачи будет засчитано

На новый год Дед Мороз подарил НурлашКО большой массив целых чисел. Узнав это, его учитель математики решил проверить, как хорошо он освоил одну из последних тем — перестановки.

Чтобы проверить это он спрашивает: «Образуют ли перестановку элементы массива с индексами от L до R включительно?» Также иногда он может изменять некоторые числа. Напомним, что перестановка из n элементов — это упорядоченный набор, состоящий из чисел 1, 2, ..., n. В нашем случае $n = R - L + 1$.

После новогодних контестов НурлашКО еще не пришел в себя. Поэтому он попросил вас о помощи, чтобы не упасть в глазах своего учителя.

Формат входных данных

Первая строка входного файла содержит число N ($1 \leqslant N \leqslant 100\,000$). Во второй строке содержатся N целых чисел $a_1, a_2, \ldots, a_N$ ($1 \leqslant a_i \leqslant N$). Третья строка входного файла содержит число M ($1 \leqslant M \leqslant 100\,000$), количество запросов учителя.

В каждой из следующих M строк записано по три целых числа — t, X, Y ($1 \leqslant t \leqslant 2$, $1 \leqslant X, Y \leqslant N$). Если t равно 1, то это запрос изменения элемента, в этом случае следует выполнить присвоение `a[X] = Y`. Если t равно 2, то следует проверить, является ли подотрезок с индексами от X до Y перестановкой, гарантируется что $X \leqslant Y$.

Формат выходных данных

Для каждого запроса второго типа в отдельной строке выведите `YES` если данный под отрезок является перестановкой, иначе `NO`.

Примеры

permutation.in	permutation.out
5 1 5 3 4 1 5 2 1 4 1 2 2 2 2 5 1 5 5 2 1 5	NO YES YES

Система оценки

Данная задача содержит шесть подзадач:

1. $1 \leqslant N, M \leqslant 1000$. Оценивается в 21 балл.
2. $1 \leqslant N, M \leqslant 50\,000$. Оценивается в 28 баллов.
3. $1 \leqslant N, M \leqslant 100\,000$, при этом во всех запросах $t = 2$. Оценивается в 22 балла.
4. $1 \leqslant N, M \leqslant 100\,000$. Оценивается в 29 баллов.

Подзадача 2 оценивается только в случае прохождения всех тестов подзадачи 1. Подзадача 4 оценивается только в случае прохождения всех тестов подзадач 1 и 2. Подзадача 3 оценивается независимо.

Задача L. Парковка

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

На кольцевой парковке есть n мест пронумерованых от 1 до n. Есть два вида событий прибытие машину на парковку и отъезд машины с парковки. Если машина приезжает на парковку, а её место занято, то она едет далее по кругу и встаёт на первое свободное место.

Формат входных данных

В первой строке входного файла находится два числа n и m — размер парковки и количество запросов($1 \leqslant n, m \leqslant 100000$). В следующих m строках находятся события. Каждая из этих строк имеет следующий вид:

- enter x — приехала машина, которая хочет встать на место x. Для каждой такой команды выведите какое место займёт эта машина.
- exit x — уехала машина занимавшая место x. Гарантируется, что на этом месте была машина.

Формат выходных данных

Выведите последовательно результаты выполнения всех операций `enter`.

Пример

parking.in	parking.out
3 5 enter 1 enter 1 exit 1 enter 2 enter 2	1 2 3 1