Задача A. RMQ наоборот

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

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 B. MP3-плеер

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

Во времена столь давние, что некоторые из вас ходили под стол пешком, а кто-то так и вообще только находился в проекте, люди покупали портативные MP3-плееры, и слушали с них музыку.

Гоги купил себе MP3-плеер. Его плеер обладает полезной функцией блокировки — после t секунд неактивности все кнопки плеера перестают выполнять свои функции, однако, будучи после этого нажатыми, они снимают с плеера блокировку и возвращают его в обычное состояние.

К примеру, пусть $t = 5$ и плеер сейчас заблокирован. Гоги нажимает на кнопку A, ждёт 3 секунды, нажимает на кнопку B, ждёт 5 секунд, нажимает на C, ждёт 6 секунд и нажимает на D. В результате такой последовательности действий только кнопки B и C выполнят свою функцию. Между нажатиями на кнопки C и D плеер становится заблокированным.

Уровень громкости MP3-плеера контролируется нажатиями на кнопки `'+'` и `'-'`, первая кнопка увеличивает уровень громкости на 1, вторая уменьшает на 1. Уровень громкости в любой момент времени — целое число между 0 и V_{max}. Нажатие на кнопку `'+'` в момент, когда громкость равна V_{max}, либо на кнопку `'-'`, когда громкость равна 0, не меняет уровень громкости.

Гоги не знает исходное значение t. Он хочет определить его экспериментально. Он производит последовательность из n нажатий на кнопки `'+'` и `'-'`. После конца эксперимента Гоги смотрит на дисплей плеера и определяет окончательный уровень громкости. К сожалению, он забыл записать начальный уровень громкости (до нажатия на какую-нибудь кнопку). Обозначим исходный уровень громкости за V_1 и известный финальный уровень громкости за V_2. Вам дано значение V_2 и список нажатых клавиш и времена их нажатия по порядку. Ваша задача — определить наибольшее возможное целочисленное значение t, удовлетворяющее описанной ситуации.

Формат входных данных

Первая строка ввода содержит три числа n, V_{max} и V_2 ($0 \leqslant n \leqslant 100\,000$, $0 \leqslant V_2 \leqslant V_{max} \leqslant 100\,000$)

Каждая из n последующих строк содержит описание одного нажатия: символ '+' или '-' и целое число c_i ($0 \leqslant c_i \leqslant 2 \cdot 10^9$), обозначающее количество секунд, прошедшее с момента начала эксперимента до произведения этого нажатия ($c_i < c_{i+1}$ для $1 \leqslant i \leqslant n-1$).

Формат выходных данных

Если t может быть произвольно большим, то выведите единственное слово "`infinity`"

В противном случае выведите единственную строку, содержащую два целых числа t и V_1. Число t должно быть максимальным возможным, а если возможных значений V_1 несколько, то число V_1 также должно быть максимальным.

Примеры

стандартный ввод	стандартный вывод
6 4 3 - 0 + 8 + 9 + 13 - 19 - 24	5 4
3 10 10 + 1 + 2 + 47	infinity

Задача C. Перестановки

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

Вася выписал на доске в каком-то порядке все числа от 1 по N, каждое число ровно по одному разу. Количество чисел оказалось довольно большим, поэтому Вася не может окинуть взглядом все числа. Однако ему надо всё-таки представлять эту последовательность, поэтому он написал программу, которая отвечает на вопрос — сколько среди чисел, стоящих на позициях с x по y, по величине лежат в интервале от k до l. Сделайте то же самое.

Формат входных данных

В первой строке лежит два натуральных числа — $1 \leqslant N \leqslant 100\,000$ — количество чисел, которые выписал Вася и $1 \leqslant M \leqslant 100\,000$ — количество вопросов, которые Вася хочет задать программе. Во второй строке дано N чисел — последовательность чисел, выписанных Васей. Далее в M строках находятся описания вопросов. Каждая строка содержит четыре целых числа $1 \leqslant x \leqslant y \leqslant N$ и $1 \leqslant k \leqslant l \leqslant N$.

Формат выходных данных

Выведите M строк, каждая должна содержать единственное число — ответ на Васин вопрос.

Примеры

permutation.in	permutation.out
4 2 1 2 3 4 1 2 2 3 1 3 1 3	1 3

Замечание

Данную задачу обязательно решать с использованием двумерного дерева отрезков.

Задача D. Окна

Имя входного файла:	`windows.in`
Имя выходного файла:	`windows.out`
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 ($1 \leqslant n \leqslant 50\,000$). 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 10^6.

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, разделенных пробелом — координаты точки, покрытой максимальным числом окон. Окна считаются замкнутыми, т. е. покрывающими свои граничные точки.

Примеры

windows.in	windows.out
2 0 0 3 3 1 1 4 4	2 1 3
1 0 0 1 1	1 0 1
4 0 0 1 1 0 1 1 2 1 0 2 1 1 1 2 2	4 1 1
5 0 0 1 1 0 1 1 2 0 0 2 2 1 0 2 1 1 1 2 2	5 1 1