Задача A Сумма на отрезке

Имя входного файла:	a.in
Имя выходного файла:	a.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 MB

Условие можно прочесть здесь: http://acm.math.spbu.ru/trains/WS2010_Day2_Junior.pdf
Читать нужно задачу A.

Задача B Сумма на матрице

Имя входного файла:	c.in
Имя выходного файла:	c.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 MB

Условие можно прочесть здесь: http://acm.math.spbu.ru/trains/WS2010_Day2_Junior.pdf
Читать нужно задачу C.

Задача C Минимум на стэке

Имя входного файла:	l.in
Имя выходного файла:	l.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 MB

Условие можно прочесть здесь: http://acm.math.spbu.ru/trains/WS2010_Day2_Junior.pdf
Читать нужно задачу L.

Задача D Минимум на очереди

Имя входного файла:	m.in
Имя выходного файла:	m.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 MB

Условие можно прочесть здесь: http://acm.math.spbu.ru/trains/WS2010_Day2_Junior.pdf
Читать нужно задачу M.

Задача E Сумма

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

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

Формат входного файла

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

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

Изначально в массиве живут нули.

Формат выходного файла

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

Задача F Среднее на отрезках

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

Дана последовательность из N чисел и M запросов. Каждый запрос представляет собой пару чисел i, j. После каждого запроса необходимо заменить все элементы последовательности от i до j на их среднее арифметическое. Если сумма элементов текущей последовательности меньше либо равна сумме элементов первоначальной последовательности, среднее округляется вверх, иначе — вниз. Ваша задача — найти последовательность после проведенных преобразований.

Формат входного файла

Первая строка входных данных содержит два натуральных числа N и M $(1 \leqslant M, N \leqslant 30\,000)$. Во второй строке записаны N чисел, каждое из которых не превосходит 10^9 по модулю — элементы последовательности. Затем идут M строчек по два числа в каждой — запросы, которые необходимо выполнить.

Формат выходного файла

Выведите N чисел — элементы результирующей последовательности.

Примеры

middle.in	middle.out
6 4 1 2 3 4 5 6 1 2 2 5 5 6 4 6	2 3 3 5 5 5

Задача G Мега-инверсии

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

Инверсией в перестановке $p_1, p_2, ..., p_N$ называется пара (i, j) такая, что $i < j$ и $p_i > p_j$. Назовем мега-инверсией в перестановке $p_1, p_2, ..., p_N$ тройку (i, j, k) такую, что $i < j < k$ и $p_i > p_j > p_k$. Придумайте алгоритм для быстрого подсчета количества мега-инверсий в перестановке.

Формат входного файла

Первая строка входного файла содержит целое число N ($1 \leqslant N \leqslant 100\,000$). Следующие N чисел описывают перестановку: $p_1, p_2, ..., p_N$ ($1 \leqslant p_i \leqslant N$), все p_i попарно различны. Числа разделяются пробелами и/или переводами строк.

Формат выходного файла

Единственная строка выходного файла должна содержать одно число, равное количеству мега-инверсий в перестановке $p_1, p_2, ..., p_N$.

Примеры

mega.in	mega.out
4 4 3 2 1	4

Задача H Обмен

Имя входного файла:	swap.in
Имя выходного файла:	swap.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 MB

Пусть все натуральные числа исходно организованы в список в естественном порядке. Разрешается выполнить следующую операцию: $swap(a, b)$. Эта операция возвращает в качестве результата расстояние в текущем списке между числами a и b и меняет их местами.

Задана последовательность операций *swap*. Требуется вывести в выходной файл результат всех этих операций.

Формат входного файла

Первая строка входного файла содержит число n ($1 \leqslant n \leqslant 200\,000$) — количество

операций. Каждая из следующих n строк содержит по два числа в диапазоне от 1 до 10^9 — аргументы операций *swap*.

Формат выходного файла

Для каждой операции во входном файле выведите ее результат.

Пример

swap.in	swap.out
4 1 4 1 3 4 5 1 4	3 1 4 2

Задача I Разрезание графа

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

Дан неориентированный граф. Над ним в заданном порядке производят операции следующих двух типов:

- `cut` — разрезать граф, то есть удалить из него ребро;
- `ask` — проверить, лежат ли две вершины графа в одной компоненте связности.

Известно, что после выполнения всех операций типа `cut` рёбер в графе не осталось. Найдите результат выполнения каждой из операций типа `ask`.

Формат входного файла

Первая строка входного файла содержит три целых числа, разделённые пробелами — количество вершин графа n, количество рёбер m и количество операций k ($1 \leqslant n \leqslant 50\,000$, $0 \leqslant m \leqslant 100\,000$, $m \leqslant k \leqslant 150\,000$).

Следующие m строк задают рёбра графа; i-ая из этих строк содержит два числа u_i и v_i ($1 \leqslant u_i, v_i \leqslant n$), разделённые пробелами — номера концов i-го ребра. Вершины нумеруются с единицы; граф не содержит петель и кратных рёбер.

Далее следуют k строк, описывающих операции. Операция типа `cut` задаётся строкой "`cut` u v" ($1 \leqslant u, v \leqslant n$), которая означает, что из графа удаляют ребро между вершинами u и v. Операция типа `ask` задаётся строкой "`ask` u v" ($1 \leqslant u, v \leqslant n$), которая означает, что необходимо узнать, лежат ли в данный момент вершины u и v в одной компоненте связности. Гарантируется, что каждое ребро графа встретится в операциях типа `cut` ровно один раз.

Формат выходного файла

Для каждой операции `ask` во входном файле выведите на отдельной строке слово "`YES`", если две указанные вершины лежат в одной компоненте связности, и "`NO`" в противном случае. Порядок ответов должен соответствовать порядку операций `ask` во входном файле.

Пример

cutting.in	cutting.out
3 3 7 1 2 2 3 3 1 ask 3 3 cut 1 2 ask 1 2 cut 1 3 ask 2 1 cut 2 3 ask 3 1	YES YES NO NO

Задача J Points on the plane

Имя входного файла:	fenwick.in
Имя выходного файла:	fenwick.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 MB

Есть квадратная клетчатая плоскость состоящая из $n \times n$ клеток ($1 \leqslant n \leqslant 1000$). Изначально в каждой клетке записано значение ноль. Ваша задача — написать программу, умеющую отвечать на следующие запросы:

- ADD x y — увеличить значение в ячейке x, y на 1.
- GET x_1 y_1 x_2 y_2 — вернуть сумму значений в прямоугольнике с углами в x_1, y_1 и x_2, y_2 соответственно.

Формат входного файла

В первой строке входного файла содержится два числа — n и k — размер доски и число запросов соответственно. Следующие k строк содержат сами запросы. Гарантируется, что общее число запросов не превосходит 300 000.

Формат выходного файла

Для каждого запроса типа GET выведите в отдельную строку одно целое число — ответ на соответствующий запрос.

Примеры

fenwick.in	fenwick.out
5 15 ADD 1 1 ADD 2 2 ADD 3 3 ADD 4 4 ADD 5 5 ADD 1 5 ADD 2 4 ADD 3 3 ADD 4 2 ADD 5 1 GET 1 1 5 5 GET 2 1 5 5 GET 1 2 5 5 GET 2 2 4 4 GET 3 3 3 3	10 8 8 6 2

Задача K Окна

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

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 входного файла

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 выходного файла

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.

Пример

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

Задача L RMQ. RMQ Наоборот (*)

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

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 входного файла

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 выходного файла

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 3
3 3 1 2 1 1 1 2 2 3 2	inconsistent