Задача A. Skip List

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

Реализуйте Skip List.

Внимание! Решать задачу с использованием чего-то другого (особенно std::set) запрещено, однако рекомендуется стрессить ваше решение с чем-то стандартным для поиска багов.

Формат входных данных

Входной файл содержит описание операций со списком, их количество не превышает 100000. В каждой строке находится одна из следующих операций:

- `insert` x — добавить в список ключ x. Если ключ x уже в списке, то ничего делать не надо.
- `delete` x — удалить из списка ключ x. Если ключа x в списке нет, то ничего делать не надо.
- `exists` x — если ключ x есть в списке, выведите «`YES`», иначе «`NO`»

Все числа во входном файле целые и по модулю не превышают 10^9.

Формат выходных данных

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

Примеры

skiplist.in	skiplist.out
insert 2 insert 5 insert 3 exists 2 exists 4 delete 5	YES NO

Задача B. Дерево интервалов

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

Вам дано n отрезков на прямой и q точек. Нужно для каждой точки найти все отрезки, в которых она содержится. Если точка совпадает с каким-либо концом отрезка, считается, что она лежит на этом отрезке.

Несмотря на то, что все входные данные в этой задаче даются сразу, нужно решать её online, то есть, отвечать на запрос сразу же, как он поступил.

Формат входных данных

В первой строке задано целое n ($1 \leqslant n \leqslant 200\,000$) — число интервалов. В следующих n строках заданы по два целых числа l_i и r_i ($-10^9 \leqslant l_i \leqslant r_i \leqslant 10^9$).

В следующей строке записано число запросов q ($1 \leqslant q \leqslant 200\,000$). Далее, в q строках заданы сами запросы x_i ($-10^9 \leqslant x_i \leqslant 10^9$).

Формат выходных данных

Для каждого запроса в своей строке выведите c_i — число отрезков, содержащих точку x_i, а затем c_i чисел $a_{i1} < a_{i2} < \ldots < a_{ic_i}$ — номера этих отрезков. Гарантируется, что $\sum_{i=1}^{q} c_i \leqslant 500\,000$.

Примеры

interval.in	interval.out
4 10 20 30 50 40 60 70 80 2 45 15	2 2 3 1 1

Задача C. Отрезки, пересекающие прямоугольник

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

Будем считать, что отрезок и прямоугольник пересекаются, если они имеют хотя бы одну общую точку.

Вам дан набор из n непересекающихся отрезков на плоскости. Выполните q запросов. Каждый запрос представляет из себя прямоугольник, со сторонами параллельными осям координат, а ответом на запрос является множество отрезков, пересекающих этот прямоугольник.

Формат входных данных

В первой строке дано целое число n ($1 \leqslant n \leqslant 50000$) — количество отрезков.

В следующих n строках дано по четыре целых числа x_1, y_1, x_2, y_2 — координаты двух концов отрезка. Все координаты по модулю не превосходят 10^6. Гарантируется, что все отрезки не параллельны осям координат.

В следующей строке дано целое число q $1 \leqslant q \leqslant 50000$) — количество запросов.

В следующих q строках даны по четыре целых числа x_1, y_1, x_2, y_2 ($x_1 \leqslant x_2, y_1 \leqslant y_2$) — координаты левого нижнего и правого верхнего угла прямоугольника из запроса.

Формат выходных данных

Для каждого запроса выведите число k — количество отрезков, пересекающих прямоугольник из запроса. Затем выведите номера этих отрезков без повторений в любом порядке. Отрезки нумеруются в порядке, в котором они идут во входных данных. Гарантируется, что суммарное число отрезков в ответе не превышает 10^6.

Примеры

src.in	src.out
2 0 0 6 -1 2 2 4 5 3 1 -1 3 2 3 2 5 4 1 1 5 6	2 1 2 1 2 1 2

Задача D. Локализация в триангуляции

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

Вам дана триангуляция множества точек, имеющих выпуклой оболочкой треугольник и множество точек-запросов. Нужно для каждой точки запроса выяснить, в каком треугольнике триангуляции она лежит. Если точка лежит на границе нескольких треугольников, можно вывести любой из них.

Нам лень было портить формат входного файла, так что надо просто поверить, что задачу нужно решать online (отвечать на запрос сразу как он приходит).

Формат входных данных

В первой строке заданы числа n и m — число вершин и граней триангуляции ($3 \leqslant n \leqslant 100\,000$, $m = 2n - 5$. В следующих n строках заданы по паре целых чисел x_i, y_i — координаты вершин ($-10^9 \leqslant x_i, y_i \leqslant 10^9$). В следующих m строках задано по три числа a_i, b_i, c_i ($1 \leqslant a_i, b_i, c_i \leqslant n$) — номера вершин, составляющих i-й треугольник. Все треугольники заданы в порядке обхода против часовой стрелки. Все треугольники не вырожденные. Охватывающий треугольник (внешняя грань) состоит из вершин 1, 2 и 3 (против часовой стрелки) и не задан во входном файле.

В следующией строке задано число запросов q ($1 \leqslant q \leqslant 100\,000$). В следующих q строках заданы запросы x_{qi} y_{qi} ($-10^9 \leqslant x_{qi}, y_{qi} \leqslant 10^9$). Гарантируется, что все запросы лежат строго внутри охватывающего треугольника.

Формат выходных данных

Для каждой точки-запроса выведите номер треугольника, в котором она лежит. Треугольники нумеруются в порядке следования во входном файле начиная с 1.

Примеры

kirkpatrick.in	kirkpatrick.out
5 5 -2 -3 3 0 -2 3 0 -1 0 1 1 5 3 1 4 5 4 2 5 5 2 3 1 2 4 4 1 0 -2 0 0 1 -1 1	3 1 2 1