Задача A. Декартово дерево

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

Вам даны пары чисел (a_i, b_i). Необходимо построить декартово дерево, такое что i-я вершина имеет ключи (a_i, b_i), вершины с ключом a_i образуют бинарное дерево поиска, а вершины с ключом b_i образуют кучу.

Формат входного файла

В первой строке записано число N — количество пар. Далее следует N ($1 \leqslant N \leqslant 50\,000$) пар (a_i, b_i). Для всех пар $|a_i|, |b_i| \leqslant 30\,000$. $a_i \neq a_j$ и $b_i \neq b_j$ для всех $i \neq j$.

Формат выходного файла

Если декартово дерево с таким набором ключей построить возможно, выведите в первой строке «`YES`», в противном случае выведите «`NO`». В случае ответа «`YES`» выведите N строк, каждая из которых должна описывать вершину. Описание вершины состоит из трёх чисел: номера предка, номера левого сына и номера правого сына. Если у вершины отсутствует предок или какой либо из сыновей, выведите на его месте число 0.

Если подходящих деревьев несколько, выведите любое.

Пример

tree.in	tree.out
7 5 4 2 2 3 9 0 5 1 3 6 6 4 11	YES 2 3 6 0 5 1 1 0 7 5 0 0 2 4 0 1 0 0 3 0 0

Задача B. Следующий

Имя входного файла:	`next.in`
Имя выходного файла:	`next.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Реализуйте структуру данных, которая поддерживает множество S целых чисел, с которым разрешается производить следующие операции:

- $add(i)$ — добавить в множество S число i (если оно там уже есть, то множество не меняется);
- $next(i)$ — вывести минимальный элемент множества, не меньший i; если искомый элемент в структуре отсутствует, необходимо вывести -1.

Формат входного файла

Исходно множество S пусто. Первая строка входного файла содержит целое число n — количество операций ($1 \leqslant n \leqslant 300\,000$). Следующие n строк содержат операции. Каждая операция имеет вид либо «$+\ i$», либо «$?\ i$». Операция «$?\ i$» задаёт запрос $next(i)$.

Если операция «$+\ i$» идёт во входном файле в начале или после другой операции «+», то она задаёт операцию $add(i)$. Если же она идёт после запроса «?» и результат этого запроса был y, то выполняется операция $add((i + y) \bmod 10^9)$.

Во всех запросах и операциях добавления параметры лежат в интервале от 0 до 10^9.

Формат выходного файла

Для каждого запроса выведите одно число — ответ на запрос.

Пример

next.in	next.out
6 + 1 + 3 + 3 ? 2 + 1 ? 4	3 4

Задача C. Range Sum Query

Имя входного файла:	`rsq2.in`
Имя выходного файла:	`rsq2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Дан массив целых чисел, изначально заполненный нулями. С ним необходимо производить два типа операций.

- $!\ i\ x$ — присвоить ячейке i значение x;
- $?\ l\ r$ — узнать сумму значений на отрезке $[l, r]$.

Формат входного файла

Первая строка входного файла содержит два числа n и m - размер массива и количество запросов соответственно ($1 \leqslant n \leqslant (10^9 + 7)$, $1 \leqslant m \leqslant 2 \cdot 10^5$). За ним следуют m запросов по одному на строке.

В процессе работы программы необходимо поддерживать два числа P и Q. Изначально $P = 91$, $Q = 47$.

Запрос вида «$!\ A\ B$» обозначает, что в ячейку $(A + P) \mod n$ необходимо записать число $(B + Q) \mod (10^9 + 7)$.

Запрос вида «$?\ A\ B$» обозначает, что необходимо посчитать сумму между элементами $(A + P) \mod n$, $(B + Q) \mod n$ включительно. Пусть ответ равен Z. Тогда необходимо изменить числа P и Q следующим образом:

$$P = (P \cdot 31 + Z) \bmod (10^9 + 7),$$
$$Q = (Q \cdot 29 + Z) \bmod (10^9 + 7).$$

Нумерация элементов массива начинается с нуля.

Все числа во входном файле не превосходят $10^9 + 7$.

Формат выходного файла

Для каждого запроса на сумму нужно вывести ответ на него на отдельной строке.

Примеры

rsq2.in	rsq2.out
10 8 ! 1 4 ! 2 5 ? 3 6 ! 4 1 ? 5 9 ! 1 4 ? 5 9 ? 5 9	52 1416 42558 103