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 входных данных

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 выходных данных

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. Очередь

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

В зимний холодный вечер наш герой Вася стоял в очереди на вокзале за билетом на финал чемпионата Codeforces. Как обычно это бывает, кассир, сказав, что он отлучится на 5 минут, ушел на целый час... Тогда Вася, чтобы не скучать в это время, стал изучать такой механизм, как очередь. Наблюдения потрясли Васю.

Каждый человек характеризуется числами a_i — важностью его дел (чем это число больше, тем важнее его дело), и числом c_i — характеристика его совести. Числа a_i образуют перестановку чисел от 1 до n.

Пусть на данный момент очередь состоит из $n-1$ людей. Рассмотрим, как ведет себя пришедший номер n. Первым делом он встает в конец текущей очереди, а дальше поступает следующим образом: если у человека, который стоит перед ним важность дел a_i меньше чем a_n, то они меняются местами (выглядит это так: человек номер n спрашивает у предыдущего: «Эээ... простите, пожалуйста, у меня очень важное дело... можете меня пропустить вперед?»), затем он опять спрашивает у впереди стоящего человека, и так далее. Если же a_i больше чем a_n, то продвижение вперед заканчивается. Но такую операцию обмена человек номер n может совершить не более, чем c_n раз.

В нашей задаче будем считать, что к моменту, когда человек номер n придет в очередь, процесс обменов в очереди из $n-1$ людей уже закончится. Если обмен возможен, то он обязательно происходит.

Ваша задача — помочь Васе промоделировать описанный процесс и найти, в каком порядке пришедшие люди будут располагаться в очереди после окончания всех обменов.

Формат входных данных

В первой строке входных данных находится целое число n — количество людей, пришедших в данную очередь ($1 \leqslant n \leqslant 10^5$). Далее в n строках заданы описания людей в порядке их прихода — целые числа a_i и c_i ($1 \leqslant a_i \leqslant n$, $0 \leqslant c_i \leqslant n$), записанные через пробел. Каждое описание находится на отдельной строке. Все a_i различны.

Формат выходных данных

Выведите перестановку чисел от 1 до n, означающую образованную по описанным правилам очередь в порядке от начала к концу. В этой последовательности i-ое число означает номер человека, который будет стоять в очереди на месте номер i после завершения процесса обменов. Люди нумеруются с 1 в порядке, в котором они заданы во входных данных. Числа разделяйте пробелом.

Примеры

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

Задача C. 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 входных данных

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 выходных данных

Для каждого запроса выведите одно число — ответ на запрос.

Пример

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

Задача D. Своппер

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

Современные компьютеры зацикливаются
в десятки раз эффективнее человека

Рекламный проспект OS Vista-N

Перед возвращением в штаб-квартиру корпорации Аазу и Скиву пришлось заполнить на местной таможне декларацию о доходах за время визита. Получилась довольно внушительная последовательность чисел. Обработка этой последовательности заняла весьма долгое время.

— Своппер кривой, — со знанием дела сказал таможенник.

— А что такое своппер? — спросил любопытный Скив.

Ааз объяснил, что своппер — это структура данных, которая умеет делать следующее.

- Взять отрезок чётной длины от x до y и поменять местами число x с $x+1$, $x+2$ с $x+3$, и т.д.
- Посчитать сумму чисел на произвольном отрезке от a до b.

Учитывая, что обсчёт может затянуться надолго, корпорация «МИФ» попросила Вас решить проблему со своппером и промоделировать ЭТО эффективно.

Формат входных данных

Во входном файле заданы один или несколько тестов. В первой строке каждого теста записаны число N — длина последовательности и число M — число операций ($1 \leqslant N, M \leqslant 100\,000$). Во второй строке теста содержится N целых чисел, не превосходящих 10^6 по модулю — сама последовательность. Далее следуют M строк — запросы в формате `1` x_i y_i — запрос первого типа, и `2` a_i b_i — запрос второго типа. Сумма всех N и M по всему файлу не превосходит 200 000. Файл завершается строкой из двух нулей. Гарантируется, что $x_i < y_i$, а $a_i \leqslant b_i$.

Формат выходных данных

Для каждого теста выведите ответы на запросы второго типа, как показано в примере. Разделяйте ответы на тесты пустой строкой.

Примеры

swapper.in	swapper.out
5 5 1 2 3 4 5 1 2 5 2 2 4 1 1 4 2 1 3 2 4 4 0 0	Swapper 1: 10 9 2

Задача E. Нервных просим условие не читать

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

> От такого нахальства Кормен бы в гробу перевернулся!
>
> Д. Э. Кнут

На одной из станций «Веревочного курса» участникам предлагается следующее испытание: они должны встать в одну линию на узкую скамейку и, не слезая с нее (то есть не касаясь земли), развернуться на ней в обратном порядке.

Несмотря на то, что это очень веселый и интересный конкурс, некоторые люди, считающие его отвратительным, утверждают, что на самом деле его идея не нова и берет корни из обрядов древних программистов. Древние программисты уверовали, что для закрепления знаний алгоритмов на строках необходимо собираться ночью и танцевать на гробе, в котором лежит мумия Дейкстры.

Древние программисты верили, что для достижения нужного эффекта совершенно необходимо было надеть сюртуки своих предков. Так как тогда жило 27-е поколение программистов, у всех них была возможность надеть один из 26 сюртуков, по одному на каждое поколение предков. Забавно, но за 26 поколений программисты ни разу не повторились с цветами сюртуков. Поэтому к моменту начала ритуала все программистыбыли одеты в сюртуки одного из 26 цветов.

Ритуальный танец выглядел следующим образом. Все участники вставали на гроб в один ряд. На земле оставался только ведущий, руководивший танцем. Время от времени он называл два числа l и r — порядковые номера каких-то танцующих. После этого все участники ритуала с порядковыми номерами с l-й по r-й включительно должны были встать в обратном порядке, то есть на позиции l после этого должен был оказаться тот, кто был на r-й позиции, на $(l+1)$-й позиции — участник с $(r-1)$-й позиции и так далее.

Легенда гласит, что за ритуалом наблюдали духи Кнута и Кормена. Время от времени забавы ради они выбирали позиции l и r. Затем они находили самое большое k, такое что для всех i от 0 до $k-1$ включительно цвета сюртуков лршат на позициях $l+i$ и $r+i$ совпадали. Потом они подлетали к гробу и шептали полученное k на ухо мумии. Узнав о таком нахальстве, мумия Дейкстры в ярости переворачивалась в гробу ровно k раз.

Сейчас на дворе 2017-й год, и противники конкурса на скамейке хотят привести неопровержимые доказательства того, что этот конкурс отвратителен, и Дейкстре бы не понравился. Но для того, чтобы это доказать, им надо сперва проанализировать его переворачивания во времена древних программистов.

Помогите им и скажите, сколько раз переворачивалась мумия Дейкстры во время ритуала.

Формат входных данных

В первой строке входного файла содержится информация о начальном расположении участников ритуала на гробе — строка длины n ($1 \leqslant n \leqslant 1000\,000$), состоящая из строчных латинских букв: i-й символ описывает цвет сюртука участника на позиции i.

Во второй строке записано единственное число m — количество событий во время ритуала ($0 \leqslant m \leqslant 10\,000$).

В следующих m строках содержится информация о произошедших событиях. В каждой из m строк содержится три числа t, l и r, описывающих событие ($t \in \{1, 2\}$, $1 \leqslant l \leqslant r \leqslant n$). Первое число t описывает тип произошедшего события. Если $t = 1$, то в этот момент ведущий называет числа l и r и лршата на позициях с l-й по r-ю встают в обратном порядке. Если $t = 2$, духи Кнута и Кормена выбирают числа l и r и нашептывают мумии соответствующее число k.

Формат выходных данных

Для каждого события типа $t = 2$ выведите на отдельной строке, сколько раз мумия Дейкстры перевернется в результате невинной забавы духов Кнута и Кормена.

Примеры

grob.in	grob.out
abacaba 4 1 3 6 1 2 7 2 1 2 2 2 5	2 1