Задача A. Всем чмоки в этом чатике!

Имя входного файла:	`chat.in`
Имя выходного файла:	`chat.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Сегодня Мэри, как программисту социальной сети «Телеграфчик», предстоит реализовать сложную систему управления чатами.

Задача Мэри усложняется тем, что в социальную сеть «Телеграфчик» внедрена продвинутая система шифрования «ZergRus», простая, как всё гениальное. Суть её в том, что в системе хранится одна переменная zerg, которая принимает значения от 0 (включительно) до $p = 10^6 + 3$ (исключая p) и меняется в зависимости от событий в системе.

В социальной сети всего n пользователей ($1 \leqslant n \leqslant 10^5$). В начале дня каждый пользователь оказывается в своём собственном чате, в котором больше никого нет. Переменная zerg в начале дня устанавливается равной 0.

В течение дня происходят события типов:

1. Участник с номером $(i + zerg) \bmod n$ посылает сообщение всем участникам, сидящим с ним в чате (в том числе и себе самому), при этом переменная zerg заменяется на $(30 \cdot zerg + 239) \bmod p$.
2. Происходит слияние чатов, в которых сидят участники с номерами $(i + zerg) \bmod n$ и $(j + zerg) \bmod n$. Если участники и так сидели в одном чате, то ничего не происходит. Если в разных, то чаты объединяются, а переменной zerg присваивается значение $(13 \cdot zerg + 11) \bmod p$.
3. Участник с номером $(i + zerg) \bmod n$ хочет узнать, сколько сообщений он не прочитал, и прочитать их. Если участник прочитал q новых сообщений, то переменной zerg присваивается значение $(100\,500 \cdot zerg + q) \bmod p$.

Вы поможете Мэри реализовать систему, обрабатывающую эти события?

Формат входных данных

В первой строке входных данных записаны натуральные числа n ($1 \leqslant n \leqslant 10^5$) — число пользователей социальной сети. и m ($1 \leqslant m \leqslant 3 \cdot 10^5$) — число событий, произошедших за день. В следующих m строках содержится описание событий. Первое целое число в строке означает тип события t ($1 \leqslant t \leqslant 3$). Если $t = 1$, далее следует число i ($0 \leqslant i < n$), по которому можно вычислить, какой участник послал сообщение. Если $t = 2$, далее следуют числа i и j ($0 \leqslant i, j < n$), по которым можно вычислить номера участников, чаты с которыми должны объединиться. Если $t = 3$, далее следует число i ($0 \leqslant i < n$), по которому можно вычислить номер участника, желающего узнать, сколько у него сообщений, и прочитать их.

Формат выходных данных

Для каждого события типа 3 нужно вывести число непрочитанных сообщений у участника.

Пример

chat.in	chat.out	Пояснение
4 10 1 0 1 2 1 1 1 2 3 1 2 1 2 1 3 3 3 2 3 2 3 2	1 1 2	4 10 1 0 1 1 1 2 1 3 3 0 2 0 1 1 1 3 0 2 2 1 3 1

Замечание

Справа указаны номера участников в запросах после декодирования.

Задача B. Соединение точек

Имя входного файла:	`connect-points.in`
Имя выходного файла:	`connect-points.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Даны N точек на плоскости. Требуется провести отрезки между некоторыми парами точек таким образом, чтобы, во-первых, из любой данной точки в любую можно было пройти по этим отрезкам, а во-вторых, суммарная длина проведённых отрезков была минимальна.

Формат входных данных

В первой строке входных данных задано число N — количество точек ($1 \leqslant N \leqslant 200$). Следующие N строк содержат по два числа X_i Y_i каждая через пробел — координаты i-й точки ($-1000 \leqslant X_i, Y_i \leqslant 1000$). Никакие две

данные точки не совпадают, никакие три не лежат на одной прямой. Все числа во входных данных целые.

Формат выходных данных

В первой строке выведите L — суммарную длину проведённых отрезков с точностью не менее шести десятичных знаков после запятой. Во второй строке выведите K — их количество. В следующих K строках выведите по два числа A_j B_j через пробел в каждой — номера точек, соединённых j-м отрезком ($1 \leqslant A_j, B_j \leqslant N$, $A_j \neq B_j$). Точки нумеруются с единицы в том порядке, в котором они даны во входных данных. Если ответов с минимальным L несколько, разрешается выводить любой из них.

Примеры

connect-points.in	connect-points.out
4 0 0 0 1 1 0 1 1	3 3 1 2 2 4 4 3
5 0 0 0 2 1 1 3 0 3 2	7.064495 4 3 1 3 2 3 4 4 5

Задача C. Разрезание графа

Имя входного файла:	cutting.in
Имя выходного файла:	cutting.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

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 входных данных

Первая строка ввода содержит три целых числа, разделённых 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 выходных данных

Для каждой операции `ask` во вводе 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 вводе.

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 D. Система непересекающихся множеств

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

Система непересекающихся множеств — структура данных, хранящая для некоторого множества его разбиение на подмножества в каждый момент времени. Она поддерживает две операции: слияние двух подмножеств и проверку принадлежности двух элементов одному подмножеству.

В этой задаче структура используется так. Дан граф из n вершин, изначально не имеющий рёбер. В граф последовательно предлагаются рёбра для добавления. Каждый раз, когда предлагается ребро между какими-то вершинами u и v веса c, следует проверить, лежат ли на этот момент u и v в одной компоненте связности. Если да, ребро игнорируется. Если нет, ребро добавляется в граф.

Найдите суммарный вес ребёр, добавленных в граф после всех указанных операций.

Формат входных данных

В первой строке заданы через пробел целые числа n и k — количество вершин в графе и количество строк, описывающих рёбра ($1 \leqslant n \leqslant 10^7$, $0 \leqslant k \leqslant 10^5$).

В следующих k строках заданы рёбра. Каждая из них имеет вид $u\ v\ c\ \Delta u\ \Delta v\ \Delta c\ m$ ($0 \leqslant u, v < n$, $0 \leqslant c < 10^9$, $|\Delta u|, |\Delta v|, |\Delta c| < 10^9$, $1 \leqslant m \leqslant 10^7$, все числа в строке целые). Такая строка означает, что последовательно поступают предложения о добавлении в граф m рёбер. Первое из них — ребро между вершинами u и v, имеющее вес c. Второе — ребро между $(u+\Delta u) \bmod n$ и $(v+\Delta v) \bmod n$, имеющее вес $(c+\Delta c) \bmod 10^9$, и так далее. Последнее из этих m рёбер соединяет вершины $(u + (m-1) \cdot \Delta u) \bmod n$ и $(v + (m-1) \cdot \Delta v) \bmod n$, а его вес равен $(c + (m-1) \cdot \Delta c) \bmod 10^9$. Напомним, что $x \bmod y$ — это наименьшее неотрицательное число z такое, что величина $z - x$ делится нацело на y.

Общее количество предлагаемых рёбер, равное сумме чисел m во всех строках, не превосходит 10^7. Среди предлагаемых рёбер могут быть кратные рёбра и петли. Обратите внимание на то, что вершины в графе нумеруются с нуля.

Формат выходных данных

В первой строке выведите одно число — суммарный вес рёбер, добавленных в граф после всех указанных операций.

Примеры

dsu.in	dsu.out
8 3 0 1 1 2 2 0 4 0 1 2 1 1 0 5 0 3 6 9 12 15 2	29
4 1 1 2 1 1 1 1 2	3

Задача E. ЛАЖУ

Имя входного файла:	`exam.in`
Имя выходного файла:	`exam.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

На мат-мехе, в аудитории 00, стоит бесконечное количество парт, расставленных в ряд и пронумерованных целыми положительными числами. Сегодня в этой аудитории проходит экзамен по Линейному Анализу Жутких Уравнений. Время от времени приходят студенты. Каждый студент знает, за какую парту он хочет сесть, но на экзамене двум студентам нельзя сидеть вместе, поэтому, если парта занята, он занимает первую свободную парту с бóльшим номером (подальше от преподавателя).

Кроме того, некоторые студенты не сдают экзамен и уходят. После этого за парту, которую занимал ушедший студент, может сесть вновь прибывший.

Преподаватель знает, когда студенты будут приходить и когда они будут уходить учить заново. Вам требуется по этой информации узнать, за какой партой будет сидеть каждый из студентов (чтобы заранее положить туда конспект).

Формат входных данных

Первая строка входных данных содержит натуральное число n — количество событий, происходящих в течение экзамена ($n \leqslant 100\,000$).

Следующие n строк содержат информацию об этих событиях. Число $a > 0$ обозначает, что пришёл студент, желающий занять парту номер a ($a \leqslant 100\,000$). Число $a < 0$ обозначает, что студент освободил парту с номером $-a$. (Гарантируется, что он за ней действительно сидел.)

Формат выходных данных

Для каждого студента выведите одно целое положительное число — номер парты, которую он займёт.

Пример

exam.in	exam.out
6 5 5 5 -6 5 5	5 6 7 6 8

Задача F. Ёлки

Имя входного файла:	firs.in
Имя выходного файла:	firs.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Граница между царствами Пети и Васи имеет вид отрезка прямой линии длиной $N - 1$ метр. Царь Вася для соблюдения секретности распорядился вдоль границы высадить N пушистых ёлок (на расстоянии одного метра друг от друга). Он думает, что благодаря этому агенты царя Пети не смогут наблюдать за его страной. Для ухода за ёлками он нанял садовника, который каждое утро проходил вдоль всех ёлок, выбирал наименее пушистую (то есть с наименьшим количеством иголок) и опрыскивал её специальным средством. (Если таких ёлок было несколько, то он выбирал первую). От этого средства количество иголок на всех ёлках в радиусе одного метра (то есть от одной до трёх ёлок) удваивалось. Однако царь Петя решил противодействовать ему и нанял другого садовника. Он каждый вечер действовал по той же схеме, что и садовник Васи (в целях конспирации, конечно), однако средство у него было другое. От этого средства все ёлки в радиусе одного метра погибали.

Вас нанял министр финансов царя Пети, чтобы узнать, через сколько дней умрут все ёлки. Напишите программу, которая скажет ему это.

Формат входных данных

В первой строке задано число деревьев N ($1 \leqslant N \leqslant 100\,000$). Во второй строке задано N целых чисел a_i ($1 \leqslant a_i \leqslant 100\,000$) — количество иголок на ёлках (в том порядке, в котором они растут). Все данные приведены по состоянию на первое утро, до прохода Васиного садовника.

Формат выходных данных

Выведите количество дней, в течение которых на границе остаётся хотя бы одна ёлка.

Примеры

firs.in	firs.out
3 3 2 2	1
3 2 2 3	2

Задача G. Гирлянда

Имя входного файла:	garland.in
Имя выходного файла:	garland.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Новогодняя гирлянда, установленная за окном Васиного дома, представляет из себя n лампочек, расположенных на одной прямой и пронумерованных подряд числами от 0 до $(n - 1)$. Вася, глядя в окно, заинтересовался порядком включения лампочек. После внимательных наблюдений он установил следующее.

Изначально все лампочки выключены. Далее каждую секунду включается одна лампочка. В первую секунду включается лампочка с номером a, во вторую — с номером $a^2 \bmod n$, в третью — с номером $a^3 \bmod n$, ..., в k-ю — лампочка с номером $a^k \bmod n$.

Вася выяснил значения чисел a и n, и теперь его заинтересовал следующий вопрос. Рассмотрим лампочку, включённую в r-ю секунду (она имеет номер $a^r \bmod n$). Сколько лампочек слева от неё, то есть с меньшими номерами, уже включены? Вася считает, что именно свойства этих чисел для разных значений r и обеспечивают гирлянде особую красоту. Однако сам он будет слишком долго вручную их вычислять. Помогите Васе — в ответ на каждый заданный им вопрос о числе r выведите количество лампочек, зажжённых до r-й зажжённой лампочки и находящихся слева от неё.

Формат входных данных

В первой строке входных данных записаны целые числа a, n и q через

пробел ($1 \leqslant a < n \leqslant 1\,000\,000$, $1 \leqslant q \leqslant 10\,000$). Следующие q строк содержат по одному целому числу r_i каждая ($1 \leqslant r_i \leqslant 100\,000$) и описывают Васины вопросы. Вася не знает, что будет, когда по этому алгоритму надо будет зажечь уже зажжённую лампочку, поэтому его вопросы таковы, что для любого i за первые r_i секунд никакая лампочка не должна будет загореться дважды.

Формат выходных данных

Выведите q строк, по одному числу в каждой строке. В i-й строке выведите количество лампочек, зажжённых раньше r_i-й зажжённой лампочки и имеющих меньшие номера.

Примеры

garland.in	garland.out
2 3 2 1 2	0 0
5 16 4 1 3 4 2	0 2 0 1
2 8 2 2 2	1 1

Пояснения к примерам

В первом примере порядок зажигания лампочек — 2, 1.

Во втором примере — 5, 9, 13, 1.

В третьем примере — 2, 4, 0.

Задача H. Гитара

Имя входного файла:	guitar.in
Имя выходного файла:	guitar.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Спать днём нельзя!

А. С. Лопатин

Девочка Соня всегда берёт с собой в поход гитару. Однажды вечером Соня и её друзья собрались вокруг костра и стали петь песни. Все очень утомились, поэтому решили, что каждый сыграет не больше одной песни за этот вечер — это поможет раньше уйти спать.

После того, как очередной человек исполнил песню, он передаёт гитару человеку, которого заранее выбрал, а сам уходит спать. Если же оказывается, что тот выбранный человек уже не сидит у костра, спать уходит вся компания.

Кроме того, как только у человека сыграют оба его соседа, ему становится скучно и он тоже уходит спать.

Определите, в каком порядке люди будут играть на гитаре.

Формат входных данных

В первой строке входных данных записано целое число n — количество ребят у костра ($3 \leqslant n \leqslant 100\,000$). Во второй строке записаны n чисел, отражающих предпочтения: i-е число — это номер человека, которому i-й человек хотел бы передать гитару. Люди перечислены в порядке обхода против часовой стрелки.

Изначально гитара у Сони, которая имеет номер 1. Гарантируется, что никто не планирует передавать гитару сам себе.

Формат выходных данных

В первой строке выведите целое число — количество песен, которые будут исполнены этим вечером. В следующей строке выведите номера людей, которые будут играть на гитаре, в том порядке, в котором они будут это делать.

Пример

guitar.in	guitar.out
5 3 3 5 1 2	3 1 3 5

Задача I. Лего

Имя входного файла:	`lego.in`
Имя выходного файла:	`lego.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Рассмотрим трёхмерную конструкцию из блоков, аналогичных блокам известного конструктора «Лего». Конструкция состоит из нескольких прямоугольных слоёв одинакового размера, расположенных один над другим. Каждой слой состоит из блоков. Каждый блок — это плоская фигура из кубиков, связная по стороне. Соседние слои скреплены между собой таким образом, что нижний кубик в каждом столбике скреплен с верхним. Различные же блоки в одном слое никак не скреплены между собой.

Два блока A и B считаются связанными, если существует такая цепочка блоков $A = C_1, C_2, C_3, \ldots, C_{n-1}, C_n = B$, что каждые два соседних блока C_i и C_{i+1} в ней скреплены. Очевидно, всю конструкцию можно поделить на компоненты связности. Найдите их количество.

Формат входных данных

В первой строке ввода записаны через пробел три числа x, y и z — длина, ширина и высота конструкции, соответственно ($1 \leqslant x, y, z \leqslant 100$). Далее размещены z блоков по y строк, описывающие конструкцию. В каждой из этих строк ровно x символов. Перед каждым блоком расположена дополнительно одна пустая строка. Каждый кубик описывается одной заглавной буквой английского алфавита. Два кубика в одном слое принадлежат одному блоку, если они соседние по стороне и обозначены одной и той же буквой.

Формат выходных данных

Выведите одно число — количество компонент связности в заданной конструкции.

Примеры

lego.in	lego.out
2 2 2 AA AA AB CD	1

lego.in	lego.out
3 3 2 AAB BCB BAA CAA CEB DDB	2

lego.in	lego.out
4 3 1 AABB ACCB AABB	3

Пояснения к примерам

В первом примере четыре отдельных кубика второго слоя крепятся к одному блоку, занимающему весь первый слой. В конструкции одна компонента связности.

Во втором примере блоки из двух кубиков образуют связный «колодец». Два блока в центре связаны друг с другом, но не связаны с «колодцем». Количество компонент связности в этой конструкции равно двум.

В третьем примере в конструкции всего один слой, состоящий из трёх блоков. Отметим, что блоки не обязательно имеют прямоугольную форму. В этой конструкции три компоненты связности.

Задача J. Числа на отрезке

Имя входного файла:	`numbers.in`
Имя выходного файла:	`numbers.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вова нарисовал на доске горизонтальную прямую, отметил на ней N точек и пронумеровал их слева направо натуральными числами от 1 до N. После этого он стал обводить некоторые точки кружочками. Время от времени Паша, чтобы оторвать его от этого занятия, спрашивает его, сколько точек на отрезке от A до B, включительно, Вова уже обвёл кружочками. Ответьте Паше на все его вопросы, чтобы не отвлекать Вову.

Формат входных данных

В первой строке входных данных записаны два целых числа N и K — количество точек на отрезке и количество событий, соответственно ($1 \leqslant N \leqslant 1\,000\,000$, $1 \leqslant K \leqslant 100\,000$).

В следующих K строках заданы события в порядке, в котором они случались. Каждая из этих строк либо содержит целое число C от 1 до N, включительно, которое означает, что Вова обвёл кружочком точку с номером C, либо имеет вид 0 A B, где $1 \leqslant A \leqslant B \leqslant N$, что означает, что Паша спросил, сколько точек на отрезке от A до B, включительно, уже обведено кружочками.

Вова обводит каждую точку не более одного раза.

Формат выходных данных

Выведите ответ на каждый вопрос Паши на отдельной строке в том порядке, в котором эти вопросы даны во входных данных.

Примеры

numbers.in	numbers.out
3 4 1 0 1 1 2 0 1 3	1 2
10 6 0 1 10 6 1 4 0 2 9 8	0 2

Задача K. Поиск минимума на отрезке

Имя входного файла:	rmq.in
Имя выходного файла:	rmq.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Задана последовательность целых чисел A длины N. Для каждой заданной пары чисел (k, l) требуется найти значение индекса минимального элемента в последовательности A, начиная с индекса k и заканчивая индексом l.

Формат входных данных

В первой строке входных данных содержатся два числа N и M — количество элементов последовательности $1 \leqslant N \leqslant 100\,000$ и количество запросов $1 \leqslant M \leqslant 100\,000$. В следующей строке через пробел перечислены элементы последовательности A. Все числа не превышают границ 32-битного числа со знаком. Последующие M строк содержат по два числа (k, l) — начало и конец отрезков, на которых требуется найти индекс минимального элемента.

Формат выходных данных

Выведите индексы минимальных элементов последовательности A на заданных отрезках для всех запросов.

Пример

rmq.in	rmq.out
6 3 1 8 4 5 3 7 1 6 2 2 3 5	1 2 5

Задача L. Поиск суммы на отрезке

Имя входного файла:	rsq.in
Имя выходного файла:	rsq.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Задана последовательность целых чисел A длины N. Для каждой заданной пары чисел (k, l) требуется найти значение суммы элементов в последовательности A, начиная с индекса k и заканчивая индексом l.

Формат входных данных

В первой строке входных данных содержатся два числа N и M — количество элементов последовательности $1 \leqslant N \leqslant 100\,000$ и количество запросов $1 \leqslant M \leqslant 100\,000$. В следующей строке через пробел перечислены элементы последовательности A. Все числа не превышают границ 32-битного числа со знаком. Последующие M строк содержат по два числа (k, l) — начало и конец отрезков, на которых требуется найти сумму элементов.

Формат выходных данных

Выведите суммы элементов последовательности A на заданных отрезках для всех запросов.

Пример

rsq.in	rsq.out
6 3 1 8 4 5 3 7 1 6 2 2 3 5	28 8 12

Задача M. Точки и отрезки

Имя входного файла:	`segments.in`
Имя выходного файла:	`segments.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Дано n отрезков на числовой прямой и m точек на этой же прямой. Для каждой из данных точек определите, скольким отрезкам она принадлежит. Точка x считается принадлежащей отрезку с концами a и b, если выполняется двойное неравенство $\min(a, b) \leqslant x \leqslant \max(a, b)$.

Формат входных данных

Первая строка содержит два целых числа n ($1 \leqslant n \leqslant 10^5$) — число отрезков и m ($1 \leqslant m \leqslant 10^5$) — число точек. В следующих n строках записаны по два целых числа a_i и b_i — координаты концов соответствующего отрезка. В последней строке записаны m целых чисел — координаты точек. Все числа во входных данных не превосходят по модулю 10^9.

Формат выходных данных

Выведите m чисел — для каждой точки выведите количество отрезков, в которых она содержится.

Примеры

segments.in	segments.out
2 2 0 5 7 10 1 6	1 0
1 3 -10 10 -100 100 0	0 0 1

Задача N. СНМ

Имя входного файла:	`snm.in`
Имя выходного файла:	`snm.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Ваша задача — реализовать **Persistent Disjoint Set Union** (персистентную систему непересекающихся множеств). Что это значит?

Про **Disjoint Set Union**:

Изначально у вас есть n элементов, пронумерованных целыми числами от 1 до n и лежащих каждый в своём множестве. Нужно научиться отвечать на два типа запросов.

- `+ a b` — объединить множества, в которых лежат элементы a и b
- `? a b` — сказать, лежат ли элементы a и b сейчас в одном множестве

Про **Persistent**:

Теперь у нас будет несколько копий (версий) структуры данных **Disjoint Set Union**.

Запросы будут выглядеть так:

- `+ i a b` — запрос к i-й структуре: объединить множества, в которых лежат элементы a и b. При этом i-я структура остаётся неизменной, создаётся новая версия, ей присваивается новый номер (какой? читайте дальше)
- `? i a b` — запрос к i-й структуре: сказать, лежат ли элементы a и b сейчас в одном множестве

Формат входных данных

На первой строке два числа N ($1 \leqslant N \leqslant 10^5$) и K ($0 \leqslant K \leqslant 10^5$) — число элементов и число запросов. Эта изначальная копия (версия) структуры имеет номер 0.

Далее следуют K строк, на каждой описание очередного запроса. Формат запросов описан выше. Запросы нумеруются целыми числами от 1 до K.

При обработки j-го запроса, если он имеет вид «`+ i a b`», новая версия получит номер j.

Формат выходных данных

Для каждого запроса вида «`? i a b`» на отдельной строке нужно вывести «`YES`» или «`NO`».

Пример

snm.in	snm.out
4 7 + 0 1 2 ? 0 1 2 ? 1 1 2 + 1 2 3 ? 4 3 1 ? 0 4 4 ? 4 1 4	NO YES YES NO

Задача O. Лексикографически минимальная подсеть

Имя входного файла:	tree.in
Имя выходного файла:	tree.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Ты думаешь по-японски! Если взялся думать — думай по-немецки!

Neon Genesis Evangelion

Рей Аянами изучает сети из генетически модифицированных деревьев. Сетью называется набор деревьев, соединённых двусторонними биокордами. Биокорды пронумерованы последовательными целыми числами, начиная с единицы. Каждый биокорд соединяет ровно два различных дерева.

Подсеть — это подмножество биокордов сети. Подсеть S устойчива, если:

- между любыми двумя различными деревьями есть путь из биокордов S,
- никакое собственное подмножество S не обладает этим свойством.

Подсети можно сравнивать лексикографически. Чтобы это сделать, следует записать номера биокордов подсети в возрастающем порядке, после чего сравнить лексикографически полученные последовательности чисел. Напомним, что последовательность a_1, a_2, …, a_p *лексикографически меньше* последовательности b_1, b_2, …, b_p, если для минимального i, для которого $a_i \neq b_i$, верно $a_i < b_i$.

Рей нужно найти одну конкретную подсеть заданной сети. Во-первых, эта подсеть должна быть устойчивой. Во-вторых, если устойчивых подсетей несколько, нужна та из них, у которой суммарная длина биокордов минимальна. В-третьих, если и таких подсетей несколько, из них нужна лексикографически минимальная.

Рей считает, что справится с этой задачей за несколько минут. А сможете ли вы это сделать?

Формат входных данных

Первая строка входных данных содержит два целых числа n и m — количество генетически модифицированных деревьев и количество двусторонних биокордов между ними ($2 \leqslant n \leqslant 100\,000$, $m \leqslant 100\,000$). Каждая из следующих m строк содержит три целых числа: номера деревьев, соединённых очередным биокордом, и длину этого биокорда. Длины биокордов неотрицательны и не превосходят 100 000. Гарантируется, что входные данные таковы, что ответ существует.

Формат выходных данных

Выведите $n - 1$ число: номера биокордов в искомой подсети. Биокорды нумеруются целыми числами от 1 до m в том порядке, в котором они заданы во входных данных.

Пример

tree.in	tree.out
4 4 1 2 1 2 3 1 3 4 1 4 1 1	1 2 3