Задача A. Масло

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

Между пунктами с номерами $1, 2, \ldots, N$ ($N \leqslant 1500$) проложено несколько дорог. Длина каждой дороги известна. По этой системе дорог можно добраться из любого упомянутого пункта в любой другой. Автозаправки расположены только в пунктах. Требуется определить, какое максимальное расстояние без заправки должен быть в состоянии проезжать автомобиль, чтобы без проблем передвигаться между пунктами.

Формат входных данных

В первой строке входного файла находятся числа N и K (количество дорог), $1 \leqslant N \leqslant 1500$, $1 \leqslant K \leqslant 400\,000$. В следующих K строках указаны пары пунктов, связанных дорогами, и расстояние между ними — целое число километров, не превышающее $10\,000$.

Формат выходных данных

В выходном файле должно оказаться одно число — длина максимального пробега без дозаправки.

Примеры

oil.in	oil.out
3 2 1 2 5 1 3 10	10

Задача B. АлгоЛэнд

Имя входного файла:	`algoland.in`
Имя выходного файла:	`algoland.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	32 мегабайта

Недавно королева страны AlgoLand придумала новый способ отмывания денег для своего королевского двора. Она решила, что всякий житель, желающий совершить путешествие из одного города страны в другой, должен расплатиться за это желание своими деньгами.

В стране AlgoLand есть N городов, пронумерованных от 1 до N. Некоторые города соединены дорогами, движение по которым разрешено в двух направлениях. Начиная движение по какой-нибудь дороге, путешественник обязательно должен доехать до ее конца.

Предположим теперь, что житель страны хочет совершить путешествие из города A в город B. Новый указ королевы гласит, что при проезде по любой дороге страны во время этого путешествия, полицейские могут взять с этого жителя таможенную пошлину в пользу королевского двора (а могут и не взять). Если при этом у жителя недостаточно денег для уплаты пошлины, то он автоматически попадает в тюрьму. Указ также устанавливает величину пошлины для каждой дороги страны. Так как королева заботится о жителях своей страны, то она запретила полицейским брать с жителя пошлину более чем два раза во время одного путешествия.

Отметим, что если существует несколько способов попасть из города A в город B, то житель может выбрать для путешествия любой из них по собственному желанию.

Напишите программу, которая определяет, какую минимальную сумму денег должен взять с собой житель, чтобы гарантированно не попасть в тюрьму во время путешествия.

Формат входных данных

Первая строка входного файла содержит числа N и M ($2 \leqslant N \leqslant 10^4, 1 \leqslant M \leqslant 10^5$), разделенные пробелом — количества городов и дорог. Следующие M строк описывают дороги. Каждая из этих строк описывает одну дорогу и содержит три числа X, Y, Z ($1 \leqslant X, Y \leqslant N$, $X \neq Y$, $1 \leqslant Z \leqslant 10^9$) разделенных пробелами, означающие, что дорога соединяет города X и Y и пошлина за ее проезд равна Z денежных единиц. Последняя строка содержит числа A и B ($1 \leqslant A, B \leqslant N, A \neq B$) - номера начального и конечного городов путешествия. Гарантируется, что существует хотя бы один способ проезда из A в B.

Формат выходных данных

Единственная строка выходного файла должна содержать одно число, равное минимальной сумме денег, которую должен взять с собой житель, чтобы иметь возможность совершить путешествие из города A в город B и при этом гарантированно не попасть в тюрьму независимо от действий полицейских.

Примеры

algoland.in	algoland.out
5 6 1 5 1 5 4 1 5 2 2 4 2 1 3 2 1000000000 3 1 1000000000 1 3	1000000000

Задача C. Электричество в каждый дом!

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

Известный чешский математик Отакар Борувка крайне увлечен проектированием электросети Моравии. В Моравии N городов, и различные строительные фирмы уже предложили Отакару M проектов построения ЛЭП между какими-то двумя городами. Известно, что в погоне за индивидуальностью и неповторимым строительным почерком, каждая фирма предлагает все свои проекты по одинаковой стоимости, отличной от стоимостей проектов других фирм. Также известно, что каждая фирма предлагает не более *трех* проектов. Можете считать, что фирмы достаточно сообразительны, чтобы не предлагать проектов соединения какого то города с самим собой, но вполне может возникнуть ситуация, что одна или несколько фирм предлагают больше одного проекта соединения одной и той же пары городов.

Борувка собрал всех своих друзей и поручил им задачу спроектировать электросеть минимальной стоимости. Как вы, наверное, уже догадались, электрическая сеть является остовным деревом, а Борувку интересуют только сети, стоимость постройки которых минимальна.

Борувка всегда был уверен, что минимальное остовное дерево у графа одно, и представьте себе его удивление, когда каждый из друзей принес ему свой проект, утверждая что его-то дерево и есть минимальное. Подозревая неладное, он думает, что причиной разных ответов стали ребра одинакового веса. Помогите ему — посчитайте количество возможных электросетей минимальной стоимости, состоящих из ЛЭП, предложенных Борувке.

Формат входных данных

В первой строке входного файла заданы два целых положительных числа N и M, не превосходящие $100\,000$.

Следующие M строк содержат по 3 целых числа каждая: $1 \leqslant a_i, b_i \leqslant N$ и $1 \leqslant c_i \leqslant 10^9$ — города, соединенные соответствующей ЛЭП, и ее стоимость. Гарантируется, что для любого числа c_i найдется не более трех ЛЭП, имеющих такую стоимость.

Формат выходных данных

Выведите одно число — ответ на задачу. Так как это число может оказаться довольно большим, выведите остаток от деления на $10^9 + 7$.

Примеры

countspans.in	countspans.out
2 2 1 2 1 2 1 1	2
3 5 1 2 2 2 3 3 3 1 3 3 1 1 2 3 1	1
5 10 2 5 8 4 5 8 2 4 3 4 2 3 4 2 3 5 4 6 5 1 6 1 3 5 3 1 5 4 5 7	6

Задача D. Разрезание графа

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

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 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-я 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 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 E. Соединение и разъединение

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

Вы когда-нибудь слышали про обход в глубину? Например, используя этот алгоритм, вы можете проверить является ли граф связным за время $O(E)$. Вы можете даже посчитать количество компонент связности за то же время.

А вы когда нибудь слышали про систему непересекающихся множеств? Используя эту структуру, вы можете быстро обрабатывать запросы "Добавить ребро в граф" и "Посчитать количество компонент связности в графе".

А вы когда-нибудь слышали о *динамической* задаче связности? В этой задаче вам необходимо обрабатывать три типа запросов:

1. Добавить ребро в граф.
2. Удалить ребро из графа.
3. Посчитать количество компонент связности в графе.

Можно считать, что граф является неориентированным. Изначально граф является пустым.

Формат входных данных

В первой строке находятся два целых числа N и K — количество вершин и количество запросов, сответствено ($1 \leqslant N \leqslant 300\,000$, $0 \leqslant K \leqslant 300\,000$). Следующие K строк содержат запросы, по одному в строке. Каждый запрос имеет один из трех типов:

1. `+` u v: Добавить ребро между вершинами u и v. Гарантируется, что такого ребра нет.
2. `-` u v: Удалить ребро между u и v. Гарантируется, что такое ребро есть.
3. `?`: Посчитать количество компонент связности в графе.

Вершины пронумерованы целыми числами от 1 до N. Во всех запросах $u \neq v$.

Формат выходных данных

Для каждого запроса типа '`?`', Выведите количество компонент связности в момент запроса.

Примеры

connect.in	connect.out
5 11 ? + 1 2 + 2 3 + 3 4 + 4 5 + 5 1 ? - 2 3 ? - 4 5 ?	5 1 1 2