Задача A. 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 входного файла

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 выходного файла

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 B. Цветные волшебники

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

Сказочная страна представляет собой множество городов, соединенных дорогами с двухсторонним движением. Причем из любого города страны можно добраться в любой другой город либо непосредственно, либо через другие города. Известно, что в сказочной стране не существует дорог, соединяющих город сам с собой и между любыми двумя разными городами, существует не более одной дороги.

В сказочной стране живут желтый и синий волшебники. Желтый волшебник, пройдя по дороге, перекрашивает ее в желтый цвет, синий — в синий. Как известно, при наложении желтой краски на синюю, либо синей краски на желтую, краски смешиваются и превращаются в краску зеленого цвета, который является самым нелюбимым цветом обоих волшебников.

В этом году в столице страны (городе f) проводится конференция волшебников. Поэтому желтый и синий волшебники хотят узнать, какое минимальное количество дорог им придется перекрасить в зеленый цвет, чтобы добраться в столицу. Изначально все дороги не покрашены.

Начальное положение желтого и синего волшебников заранее не известно. Поэтому необходимо решить данную задачу для k возможных случаев их начальных расположений.

Формат входного файла

Первая строка входного файла содержит целые числа: n ($1 \le n \le 100\,000$) и m ($1 \le m \le 500\,000$) — количество городов и дорог в волшебной стране соответственно. Третья строка содержит одно целое число f ($1 \le f \le n$) — номер города, являющегося столицей сказочной страны. В следующих m строках, находится описание дорог страны. В этих m строк записано по два целых числа a_i и b_i, означающих, что существует дорога, соединяющая города a_i и b_i. Следующая строка содержит целое число k ($1 \le k \le 100\,000$) — количество возможных начальных расположений волшебников. Далее следуют k строк, каждая из которых содержит два целых числа — номера городов, в которых изначально находится желтый и синий волшебники соответственно.

Формат выходного файла

Для каждого из k случаев, ваша программа должна вывести в выходной минимальное количество дорог, которое придется покрасить в зеленый цвет волшебникам для того, чтобы добраться в столицу.

Примеры

magic.in	magic.out
6 6 1 1 2 2 3 3 4 4 2 4 5 3 6 2 5 6 6 6	1 2

Задача C. 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 входного файла

В первой строке находятся два целых числа N и K — количество вершин и количество запросов, сответственно ($1 \le N \le 300\,000$, $0 \le K \le 300\,000$). 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 выходного файла

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

Задача D. Перемножение матриц

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

Дана бинарная матрица. Вам необходимо возвести ее в квадрат по модулю 2.

Пусть имеется две матрицы, A и B. Тогда их произведение $C = AB$ определяется следующим образом: каждый элемент вычисляется по формуле

$$c_{ij} = \bigoplus_{k=1}^{n} a_{ik} \cdot b_{kj}$$

Таким образом, c_{ij} — четность количества индексов k таких, что $a_{ik} = b_{kj} = 1$.

Формат входного файла

В первой строке входных данных содержится число n — размер матрицы ($1 \le n \le 4\,000$). Далее следуют n строк, каждая из которых состоит из n символов 0 или 1, описывающих соответствующие элементы матрицы.

Формат выходного файла

Выведите единственное число — количество единиц в результирующей матрице.

Примеры

matrixmul.in	matrixmul.out
3 011 100 101	5

Задача E. Легенды и мифы Южного Бутово

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

В 3141 году Бутово превратилось в очень опасный район, полный убийц и прочих жуликов. Настолько опасный, что там стало страшно перемещаться даже на танке!

Напомним, что Бутово состоит из N проспектов, идущих по направлению с севера на юг, и M улиц, идущих с востока на запад. Каждый проспект пересекается с каждой улицей ровно на одном перекрестке. Перемещение вдоль проспекта или улице на высокой скорости безопасно, а поворот где-либо, наоборот, очень опасен, так как для этого приходится снизить скорость, и в тот же миг вас начинают атаковать группировки местных жителей.

Управление полицией Бутово решило немного исправить ситуацию. Руководители управления решили, что можно поставить несколько постов на некоторых перекрестках, чтобы на них жители могли спокойно поворачивать, не опасаясь быть атакованными, так как полиция уже взяла на вооружение новейшие винтовки «Мушкет-1812» и, в случае чего, сможет защитить честь и достоинство каждого законопослушного гражданина.

К сожалению, именно в этом году начальник управления решил построить себе новый элитный особняк, поэтому управление решило потратить как можно меньше денег на строительство новых постов. Однако, чтобы все выглядело чистенько, необходимо построить посты так, чтобы можно было добраться от любого перекрестка до любого другого, поворачивая лишь на специально оборудованных постах. Иначе нагрянет проверка и всех уволят.

На этом проблемы Бутово не закончились. Некоторые районы Бутово являются более опасными, и посты, которые будут построены в более опасных районах, стоят дороже. У каждого района есть центр, который находится возле некоторого перекрестка. Любой другой перекресток принадлежит району, центр которого является ближайшим к перекрестку. Расстояние между перекрестками измеряется с помощью так называемого Бутовского расстояния: расстояние между двумя перекрестками — это минимальное количество промежуточных перекрестков, которое нужно преодолеть, чтобы попасть из одного в другой, плюс один. Если же существует несколько районных центров, находящихся на минимальном расстоянии от некоторого перекрестка, то на этом перекрестке постоянно происходят стычки двух авторитетов, поэтому на этом перекрестке построить пост невозможно.

Вам требуется, зная расположение районов Бутово, определить минимальную стоимость постройки постов так, чтобы можно было безопасно добраться от любого перекрестка до любого другого.

Формат входного файла

Первая строка входного файла содержит три числа N, M, R — количество проспектов, количество улиц и количество районов в Бутово, соответственно ($2 \le N, M \le 500$, $1 \le R \le 1000$). Следующие R строк содержат по три целых числа — номер проспекта и номер улицы, на которых расположен центр соответствующего района, и стоимость постройки поста в этом районе. Проспекты и улицы нумеруются с единицы, стоимость постройки всюду положительна и не превосходит одной тысячи. Никакие два района не имеют совпадающих центров.

Формат выходного файла

На первой строке выходного файла выведите два числа: количество перекрестков K, на которых стоит разместить посты, и найденную минимальную суммарную стоимость постройки. На следующих K строках должны содержаться описания найденных перекрестков: номер проспекта и номер улицы, на пересечении которых расположен соответствующий перекресток. Если посты нужным образом разместить невозможно, выведите в выходной файл единственное число -1

Примеры

police.in	police.out
4 3 2 3 1 200 1 3 150	6 1050 2 3 1 3 1 2 4 1 4 2 3 2
3 3 2 1 2 200 3 2 150	-1