Задача A. Мосты

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

Дан неориентированный граф. Требуется найти все мосты в нём.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количества вершин и рёбер графа соответственно ($1 \leqslant n \leqslant 20\,000$, $1 \leqslant m \leqslant 200\,000$).

Следующие m строк содержат описание рёбер по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — номерами концов ребра ($1 \leqslant b_i, e_i \leqslant n$).

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число b — количество мостов в заданном графе. На следующей строке выведите b целых чисел — номера рёбер, которые являются мостами, в возрастающем порядке. Рёбра нумеруются с единицы в том порядке, в котором они заданы во входном файле.

Примеры

bridges.in	bridges.out
6 7 1 2 2 3 3 4 1 3 4 5 4 6 5 6	1 3

Задача B. Точки сочленения

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

Дан неориентированный граф. Требуется найти все точки сочленения в нём.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количества вершин и рёбер графа соответственно ($1 \leqslant n \leqslant 20\,000$, $1 \leqslant m \leqslant 200\,000$).

Следующие m строк содержат описание рёбер по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — номерами концов ребра ($1 \leqslant b_i, e_i \leqslant n$).

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число b — количество точек сочленения в заданном графе. На следующей строке выведите b целых чисел — номера вершин, которые являются точками сочленения, в возрастающем порядке.

Примеры

points.in	points.out
6 7 1 2 2 3 2 4 2 5 4 5 1 3 3 6	2 2 3

Задача C. 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 D. 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 любой точки любой улицы до любой точки любой другой, 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 различными 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 любой точки любой улицы до любой точки любой другой.

Формат входных данных

Первая строка входного файла содержит три числа N, M, R — количество проспектов, количество улиц и количество районов в Бутово, соответственно ($2 \leqslant N, M \leqslant 500$, $1 \leqslant R \leqslant 1000$). 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 выходных данных

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

Задача E. Магнитные подушки

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

Город будущего застроен небоскребами, для передвижения между которыми и парковки транспорта многие тройки небоскребов соединены треугольной подушкой из однополярных магнитов. Каждая подушка соединяет ровно 3 небоскреба и вид сверху на нее представляет собой треугольник, с вершинами в небоскребах. Это позволяет беспрепятственно передвигаться между соответствующими небоскребами. Подушки можно делать на разных уровнях, поэтому один небоскреб может быть соединен различными подушками с парами других, причем два небоскреба могут соединять несколько подушек (как с разными третьими небоскребами, так и с одинаковым). Например, возможны две подушки на разных уровнях между небоскребами 1, 2 и 3, и, кроме того, магнитная подушка между 1, 2, 5.

Система магнитных подушек организована так, что с их помощью можно добираться от одного небоскреба, до любого другого в этом городе (с одной подушки на другую можно перемещаться внутри небоскреба), но поддержание каждой из них требует больших затрат энергии.

Требуется написать программу, которая определит, какие из магнитных подушек нельзя удалять из подушечной системы города, так как удаление даже только этой подушки может привести к тому, что найдутся небоскребы из которых теперь нельзя добраться до некоторых других небоскребов, и жителям станет очень грустно.

Формат входных данных

В первой строке входного файла находятся числа N и M — количество небоскребов в городе и количество работающих магнитных подушек соответственно ($3 \leqslant N \leqslant 100000$, $1 \leqslant M \leqslant 100000$). В каждой из следующих M строк через пробел записаны три числа — номера небоскребов, соединенных подушкой. Небоскребы пронумерованы от 1 до N. Гарантируется, что имеющиеся воздушные подушки позволяют перемещаться от одного небоскреба до любого другого.

Формат выходных данных

Выведите в выходной файл сначала количество тех магнитных подушек, отключение которых невозможно без нарушения сообщения в городе, а потом их номера. Нумерация должна соответствовать тому порядку, в котором подушки перечислены во входном файле. Нумерация начинается с единицы.

Примеры

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