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, не обязательно связный, но не содержащий петель и кратных рёбер. 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. Конденсация графа

Имя входного файла:	`condense2.in`
Имя выходного файла:	`condense2.out`
Ограничение по времени:	0.65 секунда
Ограничение по памяти:	256 мегабайт

Требуется найти количество рёбер в конденсации ориентированного графа. Примечание: конденсация графа не содержит кратных рёбер и петель.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и рёбер графа соответственно ($n \leqslant 10\,000, m \leqslant 100\,000$). Следующие m строк содержат описание рёбер, по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — началом и концом ребра соответственно ($1 \leqslant b_i, e_i \leqslant n$). В графе могут присутствовать кратные рёбра и петли.

Формат выходных данных

Первая строка выходного файла должна содержать одно число — количество рёбер в конденсации графа.

Примеры

condense2.in	condense2.out
4 4 2 1 3 2 2 3 4 3	2

Задача D. Компьютерная сеть

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт

Компьютерная сеть «Plunder & Flee Inc.» состоит из n серверов и m двусторонних каналов связи. Два сервера могут общаться либо по прямому каналу, либо по цепочке каналов, передавая информацию от сервера к серверу. Текущая настройка сети обеспечивает связь для любой пары серверов.

Сетевой администратор стремится максимально повысить надежность сети. Некоторые каналы связи в сети были определены как критические. Сбой на любом критическом канале разделит сеть на несвязанные сегменты. Руководство компании отреагировало на опасения администратора и согласилось профинансировать ещё один канал связи, при условии, что когда новый канал появится в сети, количество критических каналов будет сведено к минимуму.

Напишите программу, которая по заданной конфигурации сети будет выбирать пару серверов подключиться по новой линии связи. Если несколько таких пар позволяют минимизировать количество критических ссылок, то любая из них будет считаться правильным ответом.

Формат входных данных

В первой строке содержится два натуральных числа n и m ($1 \leqslant n \leqslant 10\,000$, $1 \leqslant m \leqslant 100\,000$). В следующих m строках через пробел заданы по два натуральных числа x_i и y_i, которые описывают номера серверов соединённых каналом ($1 \leqslant x_i, y_i \leqslant n,\ x_i \neq y_i$).

Формат выходных данных

Выведите пару чисел через пробел, номера серверов, которые необходимо соединить каналом, чтобы минимизировать количество критических каналов.

Примеры

стандартный ввод	стандартный вывод
7 7 1 2 2 3 2 4 2 6 3 4 4 5 6 7	7 1
5 6 1 2 2 3 3 1 4 3 4 5 5 4	4 1

Задача E. Размещение данных

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

Телекоммуникационная сеть крупной IT-компании содержит n серверов, пронумерованных от 1 до n. Некоторые пары серверов соединены двусторонними каналами связи, всего в сети m каналов. Гарантируется, что сеть серверов устроена таким образом, что по каналам связи можно передавать данные с любого сервера на любой другой сервер, возможно с использованием одного или нескольких промежуточных серверов. Множество серверов A называется отказоустойчивым, если при недоступности любого канала связи выполнено следующее условие. Для любого не входящего в это множество сервера X существует способ передать данные по остальным каналам на сервер X хотя бы от одного сервера из множества A. В условиях показан пример сети и отказоустойчивого множества из серверов с номерами 1 и 4. Данные на сервер 2 можно передать следующим образом. При недоступности канала между серверами 1 и 2 — с сервера 4, при недоступности канала между серверами 2 и 3 — с сервера 1. На серверы 3 и 5 при недоступности любого канала связи можно по другим каналам передать данные с сервера 4.

В рамках проекта группе разработчиков компании необходимо разместить свои данные в сети. Для повышения доступности данных и устойчивости к авариям разработчики хотят продублировать свои данные, разместив их одновременно на нескольких серверах, образующих отказоустойчивое множество. Чтобы минимизировать издержки, необходимо выбрать минимальное по количеству серверов отказоустойчивое множество. Кроме того, чтобы узнать, насколько гибко устроена сеть, необходимо подсчитать количество способов выбора такого множества, и поскольку это количество способов может быть большим, необходимо найти остаток от деления этого количества способов на число $10^9 + 7$. Требуется написать программу, которая по заданному описанию сети определяет следующие числа: k — минимальное количество серверов в отказоустойчивом множестве серверов, c — остаток от деления количества способов выбора отказоустойчивого множества из k серверов на число $10^9 + 7$

Формат входных данных

Первая строка входного файла содержит целые числа n и m — количество серверов и количество каналов связи соответственно ($2 \leqslant n \leqslant 200000, 1 \leqslant m \leqslant 200000$). Следующие m строк содержат по два целых числа и описывают каналы связи между серверами. Каждый канал связи задается двумя целыми числами: номерами серверов, которые он соединяет. Гарантируется, что любые два сервера соединены напрямую не более чем одним каналом связи, никакой канал не соединяет сервер сам с собой, и для любой пары серверов существует способ передачи данных с одного из них на другой, возможно с использованием одного или нескольких промежуточных серверов.

Формат выходных данных

Выведите два целых числа, разделенных пробелом: k — минимальное число серверов в отказоустойчивом множестве серверов, c — количество способов выбора отказоустойчивого множества из k серверов, взятое по модулю $10^9 + 7$

Примеры

data.in	data.out
5 5 1 2 2 3 3 4 3 5 4 5	2 3

Замечание

В приведенном примере отказоустойчивыми являются следующие множества $\{1, 3\}, \{1, 4\}, \{1, 5\}$.

Задача F. 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 магнитные 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

Задача G. Пожарные депо

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

В городе есть n перекрестков, некоторые из которых соединены односторонними дорогами. Всего в городе m дорог.

Мэр решил построить несколько пожарных депо. Для этого ему надо выбрать, на каких перекрестках их построить. При этом когда пожарная машина едет на пожар, она может ехать как по направлению движения по дороге, так и против этого направления. А вот при возвращении в депо срочности нет и ехать «по встречке» уже нельзя.

Помогите мэру построить минимальное число депо, чтобы при пожаре дома на любом перекрестке пожарная машина из одного из депо могла доехать до этого перекрестка (возможно против движения), а затем могла вернуться в свое депо (уже двигаясь только по направлению дорог).

Формат входных данных

В первой строке входного файла находятся два натуральных числа n и m ($1 \leqslant N \leqslant 20\,000, 1 \leqslant M \leqslant 200\,000$) — количество вершин и улиц. Далее в m строках перечислены улицы, каждая улица задаётся парой чисел — номерами начального и конечного перекрестка.

Формат выходных данных

Выведите одно число — минимальное число депо, которые необходимо построить.

Примеры

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

Замечание

В приведенном примере можно построить депо, например, на перекрестках 4 и 5.

Задача H. Сигнализация

Имя входного файла:	`alarm.in`
Имя выходного файла:	`alarm.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 мегабайт

Подземный бункер состоит из n комнат, соединённых $n-1$ коридорами. Каждый коридор соединяет две различные комнаты и имеет определённую длину. Бункер устроен таким образом, что из любой комнаты i можно дойти в любую другую комнату j. Заметим, что существует единственный такой путь, не проходящий по одному и тому же коридору дважды. Сумма длин коридоров, составляющих этот путь, называется расстоянием между комнатами i и j и обозначается $\rho(i, j)$.

Каждая комната бункера оборудована звуковой сигнализацией, состоящей из сирены и датчика звука, который её включает. Сирена, включённая в комнате i, активирует датчик звука в каждой комнате, расстояние до которой не превосходит расстояние d_i, определяемое мощностью этой сирены. Другими словами, включение сирены в комнате i автоматически включает сирену во всех комнатах j, таких что $\rho(i, j) \leqslant d_i$. Эта сирена, в свою очередь, может вызвать автоматическое включение других сирен и так далее.

В случае возникновения чрезвычайной ситуации некоторые сирены необходимо включить вручную, после чего звук от них автоматически включит сирены в других комнатах. Правила безопасности предписывают выбор такого набора сирен для ручного включения, который в конце концов приведёт к автоматическому включению сирен во всех комнатах.

Требуется написать программу, которая определяет минимальное количество сирен в наборе, удовлетворяющем правилам безопасности.

Формат входных данных

Первая строка входных данных содержит единственное число n — количество комнат ($1 \leqslant n \leqslant 3000$).

Вторая строка содержит последовательность из n целых чисел d_i, i-е из них равно максимальному расстоянию, на котором расположенная в комнате i сирена активирует датчики ($0 \leqslant d_i \leqslant 10^9$).

Последующие $n-1$ строк описывают коридоры бункера. В i-й из них находятся три целых числа: u_i, v_i, l_i, где u_i, v_i — номера различных комнат, соединённых коридором i, а l_i — длина этого коридора ($1 \leqslant u_i, v_i \leqslant n$; $1 \leqslant l_i \leqslant 10^9$).

Формат выходных данных

Выходные данные должны состоять из единственного числа — минимального количества сирен, которые необходимо включить вручную.

Примеры

alarm.in	alarm.out
10 1 2 2 2 6 3 4 5 4 3 1 2 5 2 3 1 2 4 5 4 5 2 4 6 4 4 7 3 1 8 1 8 9 5 8 10 4	3