,

Задача A. Есть ли цикл?

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дан ориентированный граф. Требуется определить, есть ли в нем цикл.

Формат входных данных

Сначала вводятся два числа: N и M ($1 \leqslant N \leqslant 10^5, 0 \leqslant M \leqslant 10^5$) – количество вершин и ребер графа соответственно. Далее идет M пар чисел, задающих ребра.

Формат выходных данных

Выведите «-1» без кавычек, если нет цикла. В противном случае выведите в первую строку количество вершин в цикле, а во вторую — номера вершин в цикле в порядке обхода. Если ответов несколько выведите любой.

Примеры

стандартный ввод	стандартный вывод
4 4 1 2 2 3 3 4 4 1	4 2 3 4 1

,

Задача B. 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 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. Построение

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Группа солдат-новобранцев прибыла в армейскую часть №666. После знакомства с прапорщиком стало очевидно, что от работ на кухне по очистке картофеля спасти солдат может только чудо.

Прапорщик, будучи не в состоянии запомнить фамилии, пронумеровал новобранцев от 1 до N. После этого он велел им построиться по росту (начиная с самого высокого). С этой несложной задачей могут справиться даже совсем необученные новобранцы, да вот беда, прапорщик уверил себя, что знает про некоторых солдат, кто из них кого выше, и это далеко не всегда соответствует истине.

После трех дней обучения новобранцам удалось выяснить, что знает (а точнее, думает, что знает) прапорщик. Помогите им, используя эти знания, построиться так, чтобы товарищ прапорщик остался доволен.

Формат входных данных

Сначала на вход программы поступают числа N и M ($2 \leqslant N \leqslant 10^5$, $1 \leqslant M \leqslant 2 \cdot 10^5$) — количество солдат в роте и количество пар солдат, про которых прапорщик знает, кто из них выше.

Далее идут эти пары чисел A и B по одной на строке ($1 \leqslant A, B \leqslant N$), что означает, что, по мнению прапорщика, солдат A выше, чем B.

Не гарантируется, что все пары чисел во входных данных различны.

Формат выходных данных

В первой строке выведите «`Yes`» (если можно построиться так, чтобы прапорщик остался доволен) или «`No`» (если нет).

После ответа «`Yes`» на следующей строке выведите N чисел, разделенных пробелами, — одно из возможных построений.

Примеры

стандартный ввод	стандартный вывод
2 2 1 2 2 1	No
3 7 1 2 2 3 1 3 2 3 1 2 1 2 1 3	Yes 1 2 3

,

Задача E. Премьер министр

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Новый премьер-министр решил проехать по России от Москвы до Владивостока по железной дороге, а затем вернуться обратно. Он поручил своим помощникам разработать маршрут так, чтобы не пришлось два раза проезжать через один и тот же город. Однако помощники сообщили, что для Российских железных дорог это невозможно. Определите, в каких городах премьер-министр будет вынужден побывать дважды.

Формат входных данных

В первой строке входных данных находятся числа n и m — количество российских городов, соединенных железными дорогами в единую сеть, и количество железнодорожных перегонов, соединяющих пары городов ($1 \leqslant n \leqslant 20\,000; 1 \leqslant m \leqslant 200\,000$).

Города имеют номера от 1 до n. В каждой из следующих m строк находится пара натуральных чисел, описывающая, между какими двумя городами проходит соответствующая железнодорожная ветка. В последней строке находятся два целых числа s и e ($1 \leqslant s \neq e \leqslant n$) — номера Москвы и Владивостока по версии РЖД.

Формат выходных данных

В первой строке выведите число b — количество городов, которые премьер-министру придется посетить дважды. В следующих b строках выведите b целых чисел — номера этих городов в возрастающем порядке.

Примеры

стандартный ввод	стандартный вывод
3 2 1 2 2 3 3 1	1 2

,

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 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

,

Задача G. Противопожарная безопасность

Имя входного файла:	`firesafe.in`
Имя выходного файла:	`firesafe.out`
Ограничение по времени:	0.5 секунда
Ограничение по памяти:	256 мегабайт

В Судиславле n домов. Некоторые из них соединены дорогами с односторонним движением.

В последнее время в Судиславле участились случаи пожаров. В связи с этим жители решили построить в посёлке несколько пожарных станций. Но возникла проблема: едущая по вызову пожарная машина, конечно, может игнорировать направление движения текущей дороги, однако возвращающаяся с задания машина обязана следовать правилам дорожного движения (жители Судиславля свято чтут эти правила!).

Ясно, что, где бы ни оказалась пожарная машина, у неё должна быть возможность вернуться на ту пожарную станцию, с которой она выехала. Но строительство станций стоит больших денег, поэтому на совете посёлка было решено построить минимальное количество станций таким образом, чтобы это условие выполнялось. Кроме того, для экономии было решено строить станции в виде пристроек к уже существующим домам.

Ваша задача — написать программу, рассчитывающую оптимальное положение станций.

Формат входных данных

В первой строке входного файла задано число n ($1 \leqslant n \leqslant 3\,000$). Во второй строке записано количество дорог m ($1 \leqslant m \leqslant 100\,000$). Далее следует описание дорог в формате a_i b_i, означающее, что по i-й дороге разрешается движение автотранспорта от дома a_i к дому b_i ($1 \leqslant a_i, b_i \leqslant n$).

Формат выходных данных

В первой строке выведите минимальное количество пожарных станций K, которое необходимо построить. Во второй строке выведите K чисел в произвольном порядке — дома, к которым необходимо пристроить станции. Если оптимальных решений несколько, выведите любое.

Примеры

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

,

Задача H. 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 E. 2-SAT

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

Формулировка `2-SAT`: нужно подобрать значения n булевых переменных так, чтобы все m утверждений вида $x_{i_1} = e_1 \vee x_{i_2} = e_2$ обратились в истину. В данной задаче вам гарантируется, что решение существует.

Формат входных данных

Входной файл состоит из одного или нескольких тестов.

Каждый тест описывается следующим образом. На первой строке число переменных n и число утверждений m. Каждая из следующих m строк содержит числа i_1, e_1, i_2, e_2, задает утверждение $x_{i_1} = e_1 \vee x_{i_2} = e_2$ ($0 \leqslant i_j < n$, $0 \leqslant e_j \leqslant 1$). Ограничения: сумма всех n не больше 100 000, сумма всех m не больше 300 000.

Формат выходных данных

Для каждого теста выведите строку из n нулей и единиц — значения переменных. Если у данной задачи `2-SAT` есть несколько решений, выведите любое.

Примеры

2sat.in	2sat.out
1 0 2 2 0 0 1 0 0 1 1 1 3 4 0 1 1 0 0 0 2 1 1 1 2 0 0 0 0 1	0 01 000