

Задача А. Точки сочленения

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

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 ($1 \leq n \leq 20\,000$, $1 \leq m \leq 200\,000$).

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

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.

Пример

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

Задача В. Компоненты реберной двусвязности

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Компонентой реберной двусвязности графа $\langle V, E \rangle$ называется подмножество вершин $S \subset V$, такое что для любых различных u и v из этого множества существует не менее двух реберно не пересекающихся путей из u в v .

Дан неориентированный граф. Требуется выделить компоненты реберной двусвязности в нем.

Формат входных данных

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

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

Формат выходных данных

В первой строке выходного файла выведите целое число k — количество компонент реберной двусвязности графа.

Во второй строке выведите n натуральных чисел a_1, a_2, \dots, a_n , не превосходящих k , где a_i — номер компоненты реберной двусвязности, которой принадлежит i -я вершина.

Компоненты требуется нумеровать в порядке возрастания минимального номера вершины, входящей в компоненту.

Пример

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

Задача С. Кратчайший путь

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

Дан ориентированный ациклический взвешенный граф. Требуется найти в нем кратчайший путь из вершины s в вершину t .

Формат входных данных

Первая строка входного файла содержит четыре целых числа n , m , s и t — количество вершин, дуг графа, начальная и конечная вершина соответственно.

Следующие m строк содержат описания дуг по одной на строке. Ребро номер i описывается тремя натуральными числами b_i , e_i и w_i — началом, концом и длиной дуги соответственно ($1 \leq b_i, e_i \leq n$, $|w_i| \leq 1000$).

Входной граф не содержит циклов и петель.

$1 \leq n \leq 100\,000$, $0 \leq m \leq 200\,000$.

Формат выходных данных

Первая строка выходного файла должна содержать одно целое число — длину кратчайшего пути из s в t .

Если пути из s в t не существует, выведите «Unreachable».

Пример

стандартный ввод	стандартный вывод
2 1 1 2 1 2 -10	-10

Задача D. Конденсация графа

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

Вам задан ориентированный граф с N вершинами и M ребрами ($1 \leq N \leq 200\,000$, $1 \leq M \leq 200\,000$). Найдите компоненты сильной связности заданного графа и топологически отсортируйте его конденсацию.

Формат входных данных

Граф задан во входном файле следующим образом: первая строка содержит числа N и M . Каждая из следующих M строк содержит описание ребра – два целых числа из диапазона от 1 до N – номера начала и конца ребра.

Формат выходных данных

На первой строке выведите число K – количество компонент сильной связности в заданном графе. На следующей строке выведите N чисел – для каждой вершины выведите номер компоненты сильной связности, которой принадлежит эта вершина. Компоненты сильной связности должны быть занумерованы таким образом, чтобы для любого ребра номер компоненты сильной связности его начала не превышал номера компоненты сильной связности его конца.

Пример

стандартный ввод	стандартный вывод
10 19	2
1 4	1 2 2 1 1 2 2 2 2 1
7 8	
5 10	
8 9	
9 6	
2 6	
6 2	
3 8	
9 2	
7 2	
9 7	
4 5	
3 6	
7 3	
6 7	
10 8	
10 1	
2 9	
2 7	

Задача Е. Правильная скорая помощь

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

В связи с напряженной эпидемиологической обстановкой было решено пристраивать к уже существующим домам станции скорой помощи. Город задан в виде графа, в котором ребра — это односторонние дороги, а вершины — дома. Разумеется, во время вызова скорая может игнорировать ПДД (и даже направление движения), но вот возвращаться обратно по встрече уже не получится: больной уже под контролем врачей, да и рискованно это слишком.

Экономическая обстановка тоже не самая спокойная, поэтому требуется определить минимальное количество станций, которое нужно построить, чтобы скорая могла доехать обратно до станции от любого дома города.

Формат входных данных

В первой строке входного файла задано число n ($1 \leq n \leq 3000$) — количество домов. Во второй строке записано количество дорог m ($1 \leq m \leq 10^5$). Далее следует описание дорог в формате $a_i b_i$, означающее, что по i -й дороге разрешается движение от дома a_i к дому b_i ($1 \leq a_i, b_i \leq n$).

Формат выходных данных

Выведите одно число — минимальное количество станций, которое нужно построить, чтобы скорая могла доехать от любого дома до станции, соблюдая ПДД. Если к дому пристроена станция, то от этого дома до станции, очевидно, можно доехать.

Пример

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

Задача F. Расписание электричек

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

Рассмотрим расписание движения электричек на некоторой железнодорожной линии. Нас будут интересовать только электрички, идущие в одном направлении.

Каждая электричка отправляется с некоторой станции и следует до некоторой другой станции со всеми остановками. При этом средняя маршрутная скорость у каждой электрички своя (будем считать, что весь маршрут электричка проходит с этой скоростью, временем стоянки на станциях пренебрежем). Поскольку на участке только один путь в данном направлении — электрички в процессе следования друг друга не обгоняют.

Требуется выпустить книжку-расписание электричек. Обычно такая книжка представляет собой таблицу, где в первом столбце перечислены все станции, а каждый следующий столбец соответствует электричке: если электричка проходит через станцию, то в соответствующей клетке указывается время прохождения этой электрички через эту станцию, и прочерк, если электричка через эту станцию не проходит.

Естественно, что в книжке-расписании нужно расположить электрички так, чтобы они были указаны в хронологическом порядке. А именно, если две электрички имеют хотя бы одну общую станцию (даже если она является начальной станцией для одной, и конечной — для другой электрички), электрички в расписании должны идти в том порядке, в каком они проходят через эту станцию (поскольку электрички не обгоняют друг друга, то это же будет справедливо для всех общих станций этих двух электричек). Если же электрички не имеют ни одной общей станции, то они могут быть указаны в любом порядке.

По данному расписанию движения электричек определите порядок, в котором электрички должны идти в книжке-расписании.

Формат входных данных

Сначала вводится целое число N ($1 \leq N \leq 1000$) — количество электричек.

Далее идёт описание электричек: каждая электричка задается четырьмя числами A_i, B_i, C_i, D_i ($0 \leq A_i < B_i \leq 10^6$, $1 \leq C_i \leq 100$, $0 \leq D_i \leq 10000$), которые обозначают, что данная электричка отправляется со станции « A_i -й километр» и следует до станции « B_i -й километр». Электричка отправляется с начальной станции в момент C_i . Один километр электричка проезжает за D_i секунд.

Гарантируется, что расписание можно составить корректно, в частности, никакая электричка не обгоняет другую.

Формат выходных данных

Выведите последовательность из N номеров от 1 до N — номера электричек в том порядке, в котором они должны идти в книжке-расписании. Если возможных ответов несколько, выведите любой.

Пример

стандартный ввод	стандартный вывод
3	3 2 1
1 10 3 4	
3 5 3 4	
10 11 10 1	

Замечание

Ответ 2 3 1 также будет верным.

Задача G. Компоненты вершинной двусвязности

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Компонентой вершинной двусвязности графа $\langle V, E \rangle$ называется максимальный по включению подграф (состоящий из вершин и ребер), такой что любые два ребра из него лежат на вершинно простом цикле.

Дан неориентированный граф без петель. Требуется выделить компоненты вершинной двусвязности в нем.

Формат входных данных

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

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

Формат выходных данных

В первой строке выходного файла выведите целое число k — количество компонент вершинной двусвязности графа.

Во второй строке выведите m натуральных чисел a_1, a_2, \dots, a_m , не превосходящих k , где a_i — номер компоненты вершинной двусвязности, которой принадлежит i -е ребро. Ребра нумеруются с единицы в том порядке, в котором они заданы во входном файле.

Компоненты требуется нумеровать в порядке возрастания минимального номера ребра, входящего в компоненту.

Пример

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

Задача Н. Водостоки

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

Карту местности условно разбили на квадраты, и посчитали среднюю высоту над уровнем моря для каждого квадрата.

Когда идет дождь, вода равномерно выпадает на все квадраты. Если один из четырех соседних с данным квадратом квадратов имеет меньшую высоту над уровнем моря, то вода с текущего квадрата стекает туда (и, если есть возможность, то дальше), если же все соседние квадраты имеют большую высоту, то вода скапливается в этом квадрате.

Разрешается в некоторых квадратах построить водостоки. Когда на каком-то квадрате строят водосток, то вся вода, которая раньше скапливалась в этом квадрате, будет утекать в водосток.

Если есть группа квадратов, имеющих одинаковую высоту и образующих связную область, то если хотя бы рядом с одним из этих квадратов есть квадрат, имеющий меньшую высоту, то вся вода утекает туда, если же такого квадрата нет, то вода стоит во всех этих квадратах. При этом достаточно построить водосток в любом из этих квадратов, и вся вода с них будет утекать в этот водосток.

Требуется определить, какое минимальное количество водостоков нужно построить, чтобы после дождя вся вода утекала в водостоки.

Формат входных данных

Во входном файле записаны сначала числа N и M , задающие размеры карты — натуральные числа, не превышающие 100. Далее идет N строк, по M чисел в каждой, задающих высоту квадратов карты над уровнем моря. Высота задается натуральным числом, не превышающим 10000. Считается, что квадраты, расположенные за пределами карты, имеют высоту 10001 (то есть вода никогда не утекает за пределы карты).

Формат выходных данных

В выходной файл выведите минимальное количество водостоков, которое необходимо построить.

Пример

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

Задача I. Обновление дата-центров

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

У компании BigData Inc. есть n дата-центров, пронумерованных от 1 до n , расположенных по всему миру. В этих дата-центрах хранятся данные клиентов компании (как можно догадаться из названия — большие данные!)

Основой предлагаемых компанией BigData Inc. услуг является гарантия возможности работы с пользовательскими данными даже при условии выхода какого-либо из дата-центров компании из доступности. Подобная гарантия достигается путём использования *двойной репликации* данных. Двойная репликация — это подход, при котором любые данные хранятся в двух идентичных копиях в двух различных дата-центрах.

Про каждого из m клиентов компании известны номера двух различных дата-центров $c_{i,1}$ и $c_{i,2}$, в которых хранятся его данные.

Для поддержания работоспособности дата-центра и безопасности данных программное обеспечение каждого дата-центра требует регулярного обновления. Релизный цикл в компании BigData Inc. составляет один день, то есть новая версия программного обеспечения выкладывается на каждый компьютер дата-центра каждый день.

Обновление дата-центра, состоящего из множества компьютеров, является сложной и длительной задачей, поэтому для каждого дата-центра выделен временной интервал длиной в час, в течение которого компьютеры дата-центра обновляются и, как следствие, могут быть недоступны. Будем считать, что в сутках h часов. Таким образом, для каждого дата-центра зафиксировано целое число u_j ($0 \leq u_j \leq h-1$), обозначающее номер часа в сутках, в течение которого j -й дата-центр недоступен в связи с плановым обновлением.

Из всего вышесказанного следует, что для любого клиента должны выполняться условия $u_{c_{i,1}} \neq u_{c_{i,2}}$, так как иначе во время одновременного обновления обоих дата-центров, компания будет не в состоянии обеспечить клиенту доступ к его данным.

В связи с переводом часов в разных странах и городах мира, время обновления в некоторых дата-центрах может сдвинуться на один час вперёд. Для подготовки к непредвиденным ситуациям руководство компании хочет провести учения, в ходе которых будет выбрано некоторое непустое подмножество дата-центров, и время обновления каждого из них будет сдвинуто на один час позже внутри суток (то есть, если $u_j = h-1$, то новым часом обновления будет 0, иначе новым часом обновления станет $u_j + 1$). При этом учения не должны нарушать гарантии доступности, то есть, после смены графика обновления должно по-прежнему выполняться условие, что данные любого клиента доступны хотя бы в одном экземпляре в любой час.

Учения — полезное мероприятие, но трудоёмкое и затратное, поэтому руководство компании обратилось к вам за помощью в определении минимального по размеру непустого подходящего подмножества дата-центров, чтобы провести учения только на этом подмножестве.

Формат входных данных

В первой строке находятся три целых числа n , m и h ($2 \leq n \leq 100\,000$, $1 \leq m \leq 100\,000$, $2 \leq h \leq 100\,000$) — число дата-центров компании, число клиентов компании и количество часов в сутках.

Во второй строке вам даны n чисел u_1, u_2, \dots, u_n ($0 \leq u_j < h$), j -е из которых задаёт номер часа, в который происходит плановое обновление программного обеспечения на компьютерах дата-центра j .

Далее в m строках находятся пары чисел $c_{i,1}$ и $c_{i,2}$ ($1 \leq c_{i,1}, c_{i,2} \leq n$, $c_{i,1} \neq c_{i,2}$), задающие номера дата-центров, на которых находятся данные клиента i .

Гарантируется, что при заданном расписании обновлений в дата-центрах любому клиенту в любой момент доступна хотя бы одна копия его данных.

Формат выходных данных

В первой строке выведите минимальное количество дата-центров k ($1 \leq k \leq n$), которые должны затронуть учения, чтобы не потерять гарантию доступности. Во второй строке выведите k различных целых чисел — номера кластеров x_1, x_2, \dots, x_k ($1 \leq x_i \leq n$), на которых в рамках учений обновления станут проводиться на час позже. Номера кластеров можно выводить в любом порядке.

Если возможных ответов несколько, разрешается вывести любой из них. Гарантируется, что хотя бы один ответ, удовлетворяющий условиям задачи, существует.

Примеры

стандартный ввод	стандартный вывод
3 3 5 4 4 0 1 3 3 2 3 1	1 3
4 5 4 2 1 0 3 4 3 3 2 1 2 1 4 1 3	4 1 2 3 4

Замечание

Рассмотрим первый тест из условия. Приведённый ответ является единственным способом провести учения, затронув только один дата-центр. В таком сценарии третий сервер начинает обновляться в первый час дня, и никакие два сервера, хранящие данные одного и того же пользователя, не обновляются в один и тот же час.

С другой стороны, например, сдвинуть только время обновления первого сервера на один час вперёд нельзя — в таком случае данные пользователей 1 и 3 будут недоступны в течение нулевого часа.

Задача J. Минимизация мостов

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

Добавить в граф $G = \langle V, E \rangle$ (возможно несвязный, с петлями и кратными рёбрами) ровно одно ребро, так чтобы количество мостов в данном графе стало минимально возможным.

Напомним, что мостом в графе называется такое ребро, удаление которого увеличивает число компонент связности графа.

Формат входных данных

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

Следующие m строк содержат описание рёбер по одному на строке. Ребро номер i описывается двумя натуральными числами v_i , u_i – номерами концов ребра ($1 \leq v_i, u_i \leq n$).

Формат выходных данных

Выведите наименьшее число мостов, которое можно получить добавлением ровно одного ребра.

Пример

стандартный ввод	стандартный вывод
6 7 1 2 2 3 3 4 1 3 4 5 4 6 5 6	0

Задача К. Авиалинии

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

В Берляндии скоро появятся свои авиалинии. Комитет по разработке берляндских авиалиний уже предложил свой вариант соединения городов авиарейсами. Каждый авиарейс задается парой различных городов. Рейсы односторонние. Президенту понравился план, однако он показался ему чересчур неэкономным. Требуется разработать новый план, который содержит наименьшее количество авиарейсов и удовлетворяет условию: если из города a можно было попасть в город b (возможно, с пересадками) согласно первоначальному плану, то и в новом плане это должно быть возможным. Если же это было сделать невозможно, то и согласно новому плану это не должно быть возможным. Очевидно, что из любого города можно попасть в него самого.

Формат входных данных

В первой строке входного файла записаны целые числа N и M ($1 \leq N \leq 10^3$, $0 \leq M \leq 10^4$), где N — количество городов в стране, а M — количество авиарейсов в первоначальном плане. Города нумеруются от 1 до N . Далее записано M пар различных чисел a_i, b_i обозначающих наличие рейса из a_i в b_i в первоначальном плане ($1 \leq a_i \leq N$, $1 \leq b_i \leq N$). Пары разделяются пробелами или переводами строк. Между парой городов может быть более одного авиарейса.

Формат выходных данных

В первую строку выходного файла выведите количество рейсов в новом плане. Далее выведите авиарейсы в формате, аналогичном формату входных данных. Пары разделяйте пробелами или переводами строк. Пары выводите в любом порядке. Если существует несколько решений, выведите любое.

Пример

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

Задача L. Магнитные подушки

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
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 ($3 \leq N \leq 100000$, $1 \leq M \leq 100000$). 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

стандартный ввод	стандартный вывод
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 М. Петербург?

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

– Это что за остановка –
Бологое или Поповка? –
А с платформы говорят:
– Это город Ленинград.

«Вот какой рассеянный», Самуил Маршак

Пытаясь спастись от мира спортивного программирования, Алина сбежала на вокзал и уехала прочь на ночной электричке. Минуты медленно уплывали в даль, и уставшую девочку клонило в сон. Ей снился город-сказка, где не надо программировать, а можно гулять, мечтать и наслаждаться жизнью. Внезапно дождь из **интерактивных** задач разрушил эту идиллию.

Проснувшись и открыв окно, Алина задалась вопросом весьма философского свойства: «Где я?». С перрона потерявшейся девочке сообщили, что этот город, не похожий ни на что вокруг, представляет собой неориентированный граф на n вершинах и m ребрах. Сей невероятный факт, однако, несколько не удивил Алину. Она давно мечтала побывать в одном таком городе — Петербурге. Его уникальной отличительной особенностью является то, что хотя бы **половина** его ребер — мосты (определение дано в конце условия). Так как никакие другие города Алине не интересны, она решила ограничиться расспросом находящихся на платформе эрудированных путешественников. Любой из них может по данной вершине v сообщить любое ещё не названное ребро, исходящее из нее, или же заявить об отсутствии таковых.

Алина неуверена в своих силах, поэтому попросила вас помочь ей определить, попала ли она в Петербург. Так как её поезд скоро продолжит свой путь, задать больше $3n$ вопросов не получится.

Обратите внимание, что в графе **могут** присутствовать петли и кратные ребра.

Протокол взаимодействия

В первой строке стандартного потока ввода даны два целых числа n и m ($1 \leq n, m \leq 100\,000$) — число вершин и ребер в графе соответственно.

Для того, чтобы узнать очередное ребро, исходящее из u -й вершины ($1 \leq u \leq n$), нужно вывести «? u ». После этого ваша программа на вход получит целое число v ($-2 \leq v \leq -1$ или $1 \leq v \leq n$) — $v = a + b - u$, если существует ребро ab , которое инцидентно вершине u и **ещё не было названо**, -1 , если такого ребра не существует и -2 , если вы превысили допустимое число запросов. В последнем случае ваша программа должна немедленно завершиться, в ином случае жюри не гарантирует корректность полученного вами вердикта.

Вам разрешается задать не более $3n$ вопросов.

Чтобы сообщить, что ответ найден, требуется вывести «! Yes» или «! No», в зависимости от того, является ли загаданный граф Петербургом. В случае положительного ответа выведите $\lceil \frac{m}{2} \rceil$ строк, по два целых числа u_i и v_i в каждой ($1 \leq u_i, v_i \leq n$), обозначающих, что ребро (u_i, v_i) является мостом. Любое ребро в приведенном списке должно встречаться не более одного раза (кратные ребра считаются различными).

Запрос на вывод ответа не входит в ограничение на $3n$ запросов.

Не забывайте сбрасывать буфер после каждого запроса. Например, на языке C++ надо использовать функцию `fflush(stdout)` или вызов `cout.flush()`, на Java вызов `System.out.flush()`, на Pascal `flush(output)` и `stdout.flush()` для языка Python.

Примеры

стандартный ввод	стандартный вывод
3 3 2 2 -1 3 -1 -1	? 3 ? ? 1 ? 2 ? 1 ? 1 ? 3 ! No
4 4 2 3 2 -1 4 -1 -1 -1	? 1 ? ? 2 ? 3 ? 1 ? 3 ? 3 ? 2 ? 4 ! Yes 1 2 3 4

Замечание

В условии в примере взаимодействия вводимые и выводимые данные расположены для удобства восприятия в хронологическом порядке, при реальном взаимодействии никакие «лишние» переводы строк возникать не должны.

Ввод-вывод в примерах демонстрирует пример взаимодействия вашей программы с проверяющей системой.

В первом примере был загадан граф на трех вершинах с ребрами (1, 2), (2, 3) и (3, 1).

Во втором примере была загадан граф на четырех вершинах с ребрами (1, 2), (2, 3), (3, 4) и (2, 3).

Ребро, соединяющее вершины u и v , называется мостом, если после его удаления между вершинами u и v не существует пути.

Задача N. Размещение данных

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
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 .

На рис. 1 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.

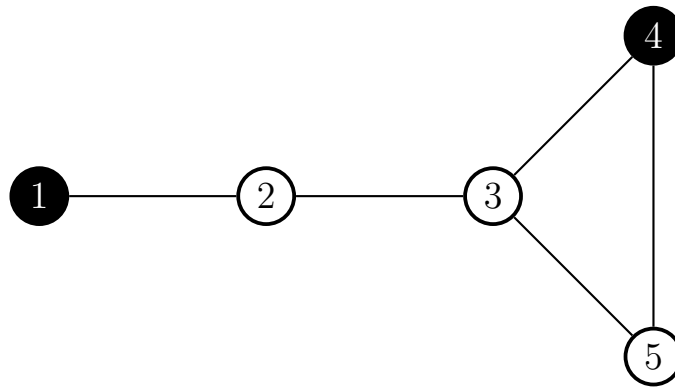


Рис. 1: Пример сети и отказоустойчивого множества серверов.

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, s — 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 ($2 \leq n \leq 200\,000$, $1 \leq m \leq 200\,000$). 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 $10^9 + 7$.

Пример

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