

Задача А. Цветные волшебники – 2

Имя входного файла: a.in
Имя выходного файла: a.out
Ограничение по времени: 2 с
Ограничение по памяти: 64 Мб

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 ($1 \leq n \leq 100\,000$) и m ($1 \leq m \leq 500\,000$) — количество городов и дорог в волшебной стране соответственно. Третья строка содержит одно целое число f ($1 \leq f \leq n$) — 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 ($1 \leq k \leq 100\,000$) — 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.

Пример

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

Задача В. Граф операций

Имя входного файла: `b.in`
 Имя выходного файла: `b.out`
 Ограничение по времени: 2 с
 Ограничение по памяти: 256 Мб

Вам задан неориентированный граф. Каждому ребру (u, v) в этом графе приписана некоторая бинарная операция и некоторое число c . Число c может иметь значение 0 или 1, а операции могут быть такие: логическое умножение (AND), логическое сложение (OR) и сложение по модулю два (XOR). Ваша задача выяснить, можно ли присвоить каждой вершине u число 0 или 1 (обозначим это значение как $A(u)$) таким образом, чтобы для каждого ребра (u, v) результат приписанной ему бинарной операции над величинами $A(u)$ и $A(v)$ был равен написанному на нем числу c .

Правила вычисления указанных операций такие:

- $(0 \text{ AND } 1) = (1 \text{ AND } 0) = (0 \text{ AND } 0) = 0, (1 \text{ AND } 1) = 1$
- $(0 \text{ OR } 1) = (1 \text{ OR } 0) = (1 \text{ OR } 1) = 1, (0 \text{ OR } 0) = 0$
- $(0 \text{ XOR } 1) = (1 \text{ XOR } 0) = 1, (1 \text{ XOR } 1) = (0 \text{ XOR } 0) = 0$

Формат входного файла

В первой строке записано два целых числа n и m ($1 \leq n \leq 1000, 0 \leq m \leq 300000$) — количество вершин и ребер в графе. Следующие m строк содержат описание ребер. Ребро задается номерами соединяемых вершин u_i, v_i ($1 \leq u_i, v_i \leq n, u_i \neq v_i$), числом c_i и названием операции (AND, OR или XOR). Никакое ребро не дано более одного раза.

Формат выходного файла

Выведите “YES”, если можно найти требуемый набор значений для вершин, и “NO” — в противном случае.

Примеры

<code>b.in</code>	<code>b.out</code>
3 3 1 2 1 OR 2 3 1 AND 1 3 1 XOR	YES
3 2 1 2 1 AND 2 3 0 OR	NO

Задача С. Регулярное паросочетание

Имя входного файла: c.in
 Имя выходного файла: c.out
 Ограничение по времени: 3 с
 Ограничение по памяти: 256 Мб

Максим с детства увлекается теорией графов. Сейчас его интересует все, что связано с двудольными графами. Двудольным графом называют граф, вершины которого можно разбить на две части так, что не существует ребер, соединяющих вершины из одной части. В одной замечательной книжке Максим прочитал, что в регулярном двудольном графе существует совершенное паросочетание. В этой же книге было написано, что регулярный граф — это граф, у которого степени всех вершин одинаковые, а совершенное паросочетание — разбиение **всех** вершин графа на пары соединенных ребром вершин. Однако, в этой книге ничего не было написано как найти это паросочетание. Всю ночь Максим провел в поиске алгоритма, однако нашел только маленькую сноску в огромной книге, что эта задача легко решается для двудольных регулярных графов с степенью 2^k . Помогите ему найти совершенное паросочетание.

Формат входного файла

В первой строке задано число n ($1 \leq n \leq 50000$) — количество вершин в каждой доле графа. Во второй строке записана степень каждой вершины d ($d = 2^k$, где $0 \leq k \leq 5$). В последующих n строках для каждой вершины первой доли записано по d чисел — номера смежных вершин второй доли. Вершины первой и второй доли нумеруются независимо от 1 до n . В графе могут быть кратные ребра. Гарантируется, что степени всех вершин, как первой, так и второй доли, равны d . В графе допускаются кратные ребра, то есть пару вершин могут соединять более одного ребра.

Формат выходного файла

Выведите ровно n чисел — для каждой вершины первой доли номер вершины второй доли, с которой она будет объединена в совершенном паросочетании. В выводе должна быть некоторая перестановка чисел от 1 до n . Если решений несколько, выведите любое.

Примеры

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

Задача D. Авиаперелеты

Имя входного файла: d.in
Имя выходного файла: d.out
Ограничение по времени: 2 с
Ограничение по памяти: 256 Мб

Главного конструктора Петю попросили разработать новую модель самолета для компании “Air Бубундия”. Оказалось, что самая сложная часть заключается в подборе оптимального размера топливного бака.

Главный картограф “Air Бубундия” Вася составил подробную карту Бубундии. На этой карте он отметил расход топлива для перелета между каждой парой городов.

Петя хочет сделать размер бака минимально возможным, для которого самолет сможет долететь от любого города в любой другой (возможно, с дозаправками в пути).

Формат входного файла

Первая строка входного файла содержит натуральное число n ($1 \leq n \leq 1000$) — число городов в Бубундии. Далее идут n строк по n чисел каждая. j -ое число в i -ой строке равно расходу топлива при перелете из i -ого города в j -ый. Все числа не меньше нуля и меньше 10^9 . Гарантируется, что для любого i в i -ой строчке i -ое число равно нулю.

Формат выходного файла

Первая строка выходного файла должна содержать одно число — оптимальный размер бака.

Пример

d.in	d.out
4 0 10 12 16 11 0 8 9 10 13 0 22 13 10 17 0	10

Задача Е. Противопожарная безопасность

Имя входного файла: e.in
Имя выходного файла: e.out
Ограничение по времени: 2 с
Ограничение по памяти: 64 Мб

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

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

Ясно, что где бы ни оказалась пожарная машина, у неё должна быть возможность вернуться на ту пожарную станцию, с которой выехала. Но строительство станций стоит больших денег, поэтому на совете города 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 входного файла

В первой строке входного файла задано число n ($1 \leq n \leq 3000$). Во второй строке записано количество дорог m ($1 \leq m \leq 100\,000$). 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 ($1 \leq a_i, b_i \leq n$).

Формат выходного файла

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

Пример

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

Задача F. Островные государства

Имя входного файла: `f.in`
 Имя выходного файла: `f.out`
 Ограничение по времени: 2 с
 Ограничение по памяти: 256 Мб

Суровые феодальные времена переживала некогда великая островная страна Байтландия. За главенство над всем островом борются два самых сильных барона. Таким образом, каждый город страны контролируется одним из правителей. Как водится издревле, некоторые из городов соединены двусторонними дорогами. Бароны очень не любят друг друга и стараются делать как можно больше пакостей. В частности, теперь для того, чтобы пройти по дороге, соединяющей города различных правителей, надо заплатить пошлину — один байтландский рубль. Кроме этого, за выезд из городов с четными номерами берется удвоенная пошлина.

Программист Вася живет в городе номер 1. С наступлением лета он собирается съездить в город N на Всебайтландское сборище программистов. Разумеется, он хочет затратить при этом как можно меньше денег и помочь ему здесь, как обычно, предлагается Вам.

Формат входного файла

В первой строке входного файла записано два числа N и M ($1 \leq N, M \leq 100\,000$) — количество городов и количество дорог соответственно.

В следующей строке содержится информация о городах — N чисел 1 или 2 — какому из баронов принадлежит соответствующий город.

В последних M строках записаны пары $1 \leq a, b \leq N$, $a \neq b$. Каждая пара означает наличие дороги из города a в город b . По дорогам Байтландии можно двигаться в любом направлении.

Формат выходного файла

Если искомого пути не существует, выведите единственное слово `impossible`. В противном случае, в первой строке напишите минимальную стоимость и количество посещенных городов, а во вторую выведите эти города в порядке посещения. Если минимальных путей несколько, выведите любой.

Пример

<code>f.in</code>	<code>f.out</code>
7 8 1 1 1 1 2 2 1 1 2 2 5 2 3 5 4 4 3 4 7 1 6 6 7	0 5 1 2 3 4 7
5 5 1 1 1 2 1 1 2 2 3 3 4 4 5 2 4	3 5 1 2 3 4 5

Задача G. Король

Имя входного файла: g.in
 Имя выходного файла: g.out
 Ограничение по времени: 2 с
 Ограничение по памяти: 256 Мб

В Тридесятом царстве, Тридевятиом государстве жил-был король. И было у короля n сыновей. В Тридесятом царстве жили n прекрасных девушек, и король знал, какие девушки нравятся каждому сыну (поскольку сыновья были молодыми и безшабашными, то им могли нравиться несколько девушек одновременно).

Однажды король приказал своему советнику подобрать для каждого сына прекрасную девушку, на которой тот сможет жениться. Советник выполнил приказ и подобрал для каждого сына для женитьбы прекрасную девушку, которая ему нравилась. Разумеется, каждая девушка может выйти замуж только за одного из сыновей.

Посмотрев на список невест, король сказал: “Мне нравится этот список, но я хочу знать для каждого сына список всех девушек, на которых он может жениться. Разумеется, при этом все сыновья также должны иметь возможность жениться на девушках, которые им нравятся”.

Эта задача оказалась для советника слишком сложной. Помогите ему избежать казни, решив ее.

Формат входного файла

Первая строка входного файла содержит число n — количество сыновей ($1 \leq n \leq 2000$). Следующие n строк содержат списки прекрасных девушек, которые нравятся сыновьям. В начале идет k_i — количество девушек, которые нравятся i -му сыну. Затем идут k_i чисел — номера девушек. Сумма k_i не превышает 200 000.

Последняя строка входного файла содержит список, составленный советником — n различных чисел от 1 до n : для каждого сына — номер прекрасной девушки, на которой он может жениться. Гарантируется, что список корректен, то есть каждому сыну нравится выбранная для него девушка.

Формат выходного файла

Выходной файл должен содержать n строк. Для каждого сына выведите l_i — количество различных девушек, на которых он может жениться. После этого выведите l_i чисел — номера девушек в произвольном порядке.

Пример

g.in	g.out
4	2 1 2
2 1 2	2 1 2
2 1 2	1 3
2 2 3	1 4
2 3 4	
1 2 3 4	

Задача Н. Каток

Имя входного файла: h.in
 Имя выходного файла: h.out
 Ограничение по времени: 2 с
 Ограничение по памяти: 64 Мб

Бизнесмен Николай очень любит кататься на коньках. Каждый год 31-го декабря он собирает компанию друзей, арендует каток и весело встречает там Новый Год. Недавно Николаю в голову пришла мысль, что гораздо приятнее было бы иметь собственный каток и бесплатно кататься на нем в любое удобное время. Недолго думая, он решил реализовать эту идею, то есть построить для себя каток, и нанял архитектора Дмитрия.

Архитектор узнал, что любимое число Николая — число P , а любимый способ катания на коньках — катание вдоль бортика. Чтобы угодить клиенту, Дмитрий решил построить такой каток, на котором Николай, прокатившись вдоль всех бортиков, проедет ровно P сотен метров. Ему уже почти удалось нарисовать план такого катка, но Николай неожиданно предъявил новое требование — каток должен иметь площадь ровно S десятков тысяч квадратных метров (на меньшей площади не поместятся все друзья Николая, а большая слишком дорого обойдется).

Если некоторая площадь катка граничит с площадью, не принадлежащей катку, то между ними обязательно необходимо построить бортик. Таким образом, запрещается устанавливать бортики в произвольном месте катка, а разрешается лишь на его границе.

Новый каток будет построен недалеко от коттеджа Николая. В данный момент под этот проект выделили квадратный участок со стороной 400 метров. Проект катка должен быть таким, чтобы все углы катка были прямые, длины бортиков кратны 100 метрам, а сами бортики параллельны сторонам участка, на котором будет вестись строительство.

Помогите Дмитрию придумать план катка, который удовлетворял бы всем требованиям.

Формат входного файла

В единственной строке входного файла записаны два целых числа P и S ($1 \leq P \leq 40$, $1 \leq S \leq 16$) — периметр и площадь будущего катка, соответственно.

Формат выходного файла

В выходной файл выведите “Impossible”, если спроектировать каток, удовлетворяющий указанным требованиям невозможно.

В противном случае, выведите 4 строки по 4 символа в каждой — план нового катка. Каждый символ будет описанием квадрата 100×100 метров участка, на котором будет вестись строительство. Если соответствующий квадрат участка находится внутри катка, то выведите “*”, иначе — “.”.

Примеры

h.in	h.out
18 8	**** *.*. **..
3 1	Impossible

Задача I. Предок

Имя входного файла: `i.in`
Имя выходного файла: `i.out`
Ограничение по времени: 2 с
Ограничение по памяти: 256 Мб

Напишите программу, которая для двух вершин дерева определяет, является ли одна из них предком другой.

Формат входного файла

Первая строка входного файла содержит натуральное число n ($1 \leq n \leq 100\,000$) — количество вершин в дереве. Во второй строке находится n чисел, i -ое из которых определяет номер непосредственного родителя вершины с номером i . Если это число равно нулю, то вершина является корнем дерева.

В третьей строке находится число m ($1 \leq m \leq 100\,000$) — количество запросов. Каждая из следующих m строк содержит два различных числа a и b ($1 \leq a, b \leq n$).

Формат выходного файла

Для каждого из m запросов выведите на отдельной строке число 1, если вершина a является одним из предков вершины b , и 0 — в противном случае.

Пример

<code>i.in</code>	<code>i.out</code>
6	0
0 1 1 2 3 3	1
5	1
4 1	0
1 4	0
3 6	
2 6	
6 5	

Задача J. Неизбежность

Имя входного файла: j.in
 Имя выходного файла: j.out
 Ограничение по времени: 2 с
 Ограничение по памяти: 256 Мб

Вася живет в первой вершине связного неориентированного графа, состоящего из n вершин и m ребер. Каждый день он ходит в школу, находящуюся в вершине с номером n . Вася старается каждый день ходить в школу новым маршрутом, однако однажды он заметил, что некоторые ребра он проходит каждый день, независимо от того, каким маршрутом идет. Помогите Васе найти все такие ребра.

Формат входного файла

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

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 выходного файла

Первая строка выходного файла должна содержать одно натуральное число b — количество ребер, которые неизбежно встречаются на пути Васи. На следующей строке выведите b целых чисел — номера этих ребер в возрастающем порядке. Ребра нумеруются с единицы в том порядке, в котором они заданы во входном файле.

Пример

j.in	j.out
7 8 1 2 2 3 3 1 4 3 5 4 5 6 4 6 6 7	2 4 8
4 4 1 2 2 4 1 4 2 3	0

Задача К. Мосты

Имя входного файла: k.in
Имя выходного файла: k.out
Ограничение по времени: 2 с
Ограничение по памяти: 256 Мб

Дан неориентированный граф. Требуется найти все мосты в нем.

Формат входного файла

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

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 выходного файла

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

Пример

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

Задача L. Цветные волшебники

Имя входного файла: 1.in
Имя выходного файла: 1.out
Ограничение по времени: 2 с
Ограничение по памяти: 64 Мб

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 ($1 \leq n \leq 100$) и m ($1 \leq m \leq 500$) — количество городов и дорог в волшебной стране соответственно. Третья строка содержит одно целое число f ($1 \leq f \leq n$) — 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 ($1 \leq k \leq 100$) — 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.

Пример

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

Задача М. Точки сочленения

Имя входного файла: `m.in`
Имя выходного файла: `m.out`
Ограничение по времени: 2 с
Ограничение по памяти: 256 Мб

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

Формат входного файла

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

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 выходного файла

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

<code>m.in</code>	<code>m.out</code>
9 12	3
1 2	1
2 3	2
4 5	3
2 6	
2 7	
8 9	
1 3	
1 4	
1 5	
6 7	
3 8	
3 9	

Задача N. Почтальон

Имя входного файла: `n.in`
 Имя выходного файла: `n.out`
 Ограничение по времени: 2 с
 Ограничение по памяти: 256 Мб

В городе есть n площадей, соединенных улицами. При этом количество улиц не превышает ста тысяч и существует не более трех площадей, на которые выходит нечетное число улиц. Для каждой улицы известна ее длина. По улицам разрешено движение в обе стороны. В городе есть хотя бы одна улица. От любой площади до любой можно дойти по улицам.

Почтальону требуется пройти хотя бы один раз по каждой улице так, чтобы длина его пути была наименьшей. Он может начать движение на любой площади и закончить также на любой (в том числе и на начальной).

Формат входного файла

Первая строка входного файла содержит натуральное число n — количество площадей в городе ($1 \leq n \leq 1000$). Далее следуют n строк, задающих улицы. В i -ой из этих строк находится число m_i — количество улиц, выходящих из площади i . Далее следуют m_i пар положительных чисел. В j -ой паре первое число — номер площади, в которую идет j -ая улица с i -ой площади, а второе число — длина этой улицы.

Между двумя площадями может быть несколько улиц, но не может быть улиц с площади на нее саму.

Все числа во входном файле не превосходят 10^5 .

Формат выходного файла

Если решение существует, то в первую строку выходного файла выведите одно число — количество улиц в искомом маршруте (считая первую и последнюю), а во вторую — номера площадей в порядке их посещения.

Если решений нет, выведите в выходной файл одно число -1 .

Если решений несколько, выведите любое.

Пример

<code>n.in</code>	<code>n.out</code>
4	5
2 2 1 2 2	1 2 3 4 2 1
4 1 2 4 4 3 5 1 1	
2 2 5 4 8	
2 3 8 2 4	

Задача О. Конденсация графа

Имя входного файла: o.in
Имя выходного файла: o.out
Ограничение по времени: 2 с
Ограничение по памяти: 256 Мб

Требуется найти количество ребер в конденсации ориентированного графа. Примечание: конденсация графа не содержит кратных ребер.

Формат входного файла

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

Формат выходного файла

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

Пример

o.in	o.out
4 4 2 1 3 2 2 3 4 3	2

Задача Р. Топологическая сортировка

Имя входного файла: `p.in`
Имя выходного файла: `p.out`
Ограничение по времени: 2 с
Ограничение по памяти: 256 Мб

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

Формат входного файла

В первой строке входного файла даны два натуральных числа N и M ($1 \leq N \leq 100\,000$, $M \leq 100\,000$) — количество вершин и рёбер в графе соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

Формат выходного файла

Вывести любую топологическую сортировку графа в виде последовательности номеров вершин. Если граф невозможно топологически отсортировать, вывести -1.

Пример

<code>p.in</code>	<code>p.out</code>
6 6 1 2 3 2 4 2 2 5 6 5 4 6	4 6 3 1 2 5
3 3 1 2 2 3 3 1	-1