Графы, их хранение.
Обход в глубину (Depth-First-Search) и в ширину (Breadth-First-Search).

A. *Стоки и истоки графа*

Вершина ориентированного графа называется истоком, если в неё не входит ни одно ребро и стоком, если из неё не выходит ни одного ребра.

Ориентированный граф задан матрицей смежности. Найдите все вершины графа, которые являются истоками, и все его вершины, которые являются стоками.

Сначала вводится число N ($1 \leqslant N \leqslant 100$) — количество вершин в графе, а затем N строк по N чисел, каждое из которых равно 0 или 1, — его матрица смежности: в i-ой строке на j-ом месте стоит 1, если в графе есть ребро из вершины i в вершину j, и 0 в противном случае.

В первой строке выведите k — число истоков в графе и затем k чисел — номера вершин, которые являются истоками, в возрастающем порядке. Затем выведите информацию о стоках в том же порядке.

Input	Output
5 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0	2 3 4 3 1 4 5

B. *Турнирный граф*

Ориентированный граф называется турниром, если между любой парой его различных вершин существует ровно одно ребро. Для заданного списком рёбер графа проверьте, является ли он турниром.

Сначала вводятся числа N ($1 \leqslant N \leqslant 100$) — количество вершин в графе и M ($1 \leqslant M \leqslant N(N-1)$) — количество ребер. Затем следует M пар чисел — ребра графа (ребро из первой вершины во вторую).

Выведите `YES`, если граф является турниром, и `NO` в противном случае.

Input	Output
5 10 1 2 1 3 1 5 2 3 2 5 4 1 4 2 4 3 4 5 5 3	YES

C. *Транзитивность неориентированного графа*

Напомним, что граф называется транзитивным, если для любых попарно различных вершин u, v, w из того, что вершины u и v соединены ребром и вершины v и w соединены ребром следует, что вершины u и w соединены ребром.

Проверьте, что заданный неориентированный граф является транзитивным.

Сначала вводятся числа N ($1 \leqslant N \leqslant 100$) — количество вершин в графе и M ($1 \leqslant M \leqslant \dfrac{n(n-1)}{2}$) — количество рёбер. Затем следует M пар чисел — рёбра графа.

Выведите `YES`, если граф является транзитивным, и `NO` в противном случае.

Input	Output
5 4 1 2 2 3 3 1 4 5	YES

D. *Транзитивность ориентированного графа*

Напомним, что ориентированный граф называется транзитивным, если для любых трёх различных вершин u, v и w из того, что из u в вершину v ведет ребро и из вершины v в вершину w ведет ребро, следует, что из вершины u в вершину w ведет ребро.

Проверьте, что заданный ориентированный граф является транзитивным.

Сначала вводятся числа N ($1 \leqslant N \leqslant 100$) — количество вершин в графе, а затем N строк по N чисел, каждое из которых равно 0 или 1, — его матрица смежности.

Выведите `YES`, если граф является транзитивным, и `NO` в противном случае.

Input	Output
5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	YES

E. *Количество вершин в компоненте связности*

Дан неориентированный граф. Требуется найти количество вершин, лежащих в одной компоненте связности с данной вершиной (считая эту вершину).

В первой строке входных данных содержатся два числа: N и S ($1 \leqslant N \leqslant 100; 1 \leqslant S \leqslant N$), где N — количество вершин графа, а S — заданная вершина. В следующих N строках записано по N чисел — матрица смежности графа, в которой 0 означает отсутствие ребра между вершинами, а 1 — его наличие. Гарантируется, что на главной диагонали матрицы всегда стоят нули.

Выведите одно целое число — искомое количество вершин.

Input	Output
3 1 0 1 1 1 0 0 1 0 0	3

F. *Проверить является ли граф деревом*

Дан неориентированный граф. Необходимо определить, является ли он деревом.

В первой строке входного файла содержится одно натуральное число N ($N \leqslant 100$) — количество вершин в графе. Далее в N строках по N чисел — матрица смежности графа.

На главной диагонали матрицы стоят нули. Матрица симметрична относительно главной диагонали.

Вывести `YES`, если граф является деревом, и `NO` в противном случае.

Input	Output
6 0 1 1 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0	NO
3 0 1 0 1 0 1 0 1 0	YES

G. *Компоненты связности*

Дан неориентированный граф. Необходимо посчитать количество его компонент связности и вывести их.

Во входном файле записано два числа N и M ($0 < N \leqslant 100000, 0 \leqslant M \leqslant 100000$). В следующих M строках записаны по два числа i и j ($1 \leqslant i, j \leqslant N$), которые означают, что вершины i и j соединены ребром.

В первой строчке выходного файла выведите количество компонент связности. Далее выведите сами компоненты связности в следующем формате: в первой строке количество вершин в компоненте, во второй — сами вершины в произвольном порядке.

Input	Output
6 4 3 1 1 2 5 4 2 3	3 3 1 2 3 2 4 5 1 6

H. *Есть ли цикл*

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

В первой строке входного файла содержится одно натуральное число N ($N \leqslant 50$) — количество вершин в графе. Далее в N строках по N чисел — матрица смежности графа. Гарантируется, что на диагонали матрицы будут стоять нули.

Выведите NO, если в заданном графе цикла нет, и `YES`, если он есть.

Input	Output
3 0 1 0 0 0 1 0 0 0	NO
3 0 1 0 0 0 1 1 0 0	YES

I. *Найти цикл*

Дан неориентированный граф. Требуется определить, есть ли в нём цикл, и, если есть, вывести его.

Циклом называется простой путь $v_1 \to v_2, v_2 \to v_3, \ldots, v_{n-1} \to v_n$, такой, что существует также ребро $v_n \to v_1$.

В первой строке дано одно число N ($1 \leqslant N \leqslant 500$) — количество вершин в графе. Далее в N строках задан сам граф матрицей смежности.

Если в исходном графе нет цикла, то выведите `NO`. Иначе, в первой строке выведите `YES`, во второй строке выведите число K — количество вершин в цикле, а в третьей строке выведите K различных чисел — номера вершин, которые принадлежат циклу в порядке обхода (обход можно начинать с любой вершины цикла). Если циклов несколько, то выведите любой.

Input	Output
3 0 1 1 1 0 1 1 1 0	YES 3 3 2 1
4 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0	NO

J. *Рассадка (проверка графа на двудольность)*

На банкет были приглашены N Очень Важных Персон (ОВП). Были поставлены 2 стола. Столы достаточно большие, чтобы все посетители банкета могли сесть за любой из них. Проблема заключается в том, что некоторые ОВП не ладят друг с другом и не могут сидеть за одним столом. Вас попросили определить, возможно ли всех ОВП рассадить за двумя столами.

В первой строке входных данных содержатся два числа: N и M ($1 \leqslant N, M \leqslant 100$), где N — количество ОВП, а M — количество пар ОВП, которые не могут сидеть за одним столом. В следующих M строках записано по 2 числа — пары ОВП, которые не могут сидеть за одним столом.

Если способ рассадить ОВП существует, то выведите `YES` в первой строке и номера ОВП, которых необходимо посадить за первый стол, во второй строке. В противном случае в первой и единственной строке выведите `NO`.

Input	Output
3 2 1 2 1 3	YES 1

K. *Построение (топологическая сортировка)*

Группа солдат-новобранцев прибыла в армейскую часть.

Прапорщик пронумеровал новобранцев числами от 1 до N. После этого он велел им построиться по росту (начиная с самого высокого). При этом прапорщик уверил себя, что знает про некоторых солдат, кто из них кого выше, и это далеко не всегда соответствует истине.

После трёх 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.

Сначала на вход программы поступают числа N и M ($1 < N \leqslant 100, 1 \leqslant M \leqslant 5000$) — 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.

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.

Input	Output
4 5 1 2 2 3 3 4 1 4 4 1	NO

L. *Перегоны*

На некоторой железнодорожной ветке расположено N станций, которые последовательно пронумерованы числами от 1 до N. Известны расстояния между некоторыми станциями. Требуется точно вычислить длины всех перегонов между соседними станциями или указать, что это сделать невозможно (то есть приведенная информация является противоречивой или ее недостаточно).

Во входном файле записаны сначала числа N — количество станций ($2 \leqslant N \leqslant 10^5$) и E — количество пар станций, расстояния между которыми заданы ($0 \leqslant E \leqslant 10^5$). Далее идет E троек чисел, первые два числа каждой тройки задают номера станций (это числа из диапазона от 1 до N), а третье — расстояние между этими станциями (все эти расстояния заданы точно и выражаются вещественными неотрицательными числами не более чем с 3-я знаками после десятичной точки).

В случае, когда восстановить длины перегонов можно однозначно, в выходной файл выведите сначала число 1, а затем $N-1$ вещественное число. Первое из этих чисел должно соответствовать расстоянию от 1-й станции до 2-й, второе — от 2-й до 3-й, и так далее. Все числа должны быть выведены с точностью до 3-х знаков после десятичной точки.

Если приведенная информация о расстояниях между станциями является противоречивой или не позволяет однозначно точно восстановить длины перегонов, выведите в выходной файл одно число 2.

Input	Output
2 3 1 1 0 2 2 0 2 1 2.5	1 2.500
15 13 1 2 1 1 3 2 1 4 3 1 5 4 1 6 5 1 7 6 1 8 7 1 9 8 1 10 9 1 11 10 1 12 11 1 13 12 15 14 13	2

M. *Поиск эйлерова пути*

В городе есть N площадей, соединённых улицами. При этом количество улиц не превышает 100000 и существует не более трёх площадей, на которые выходит нечётное количество улиц. Для каждой улицы известна её длина. По каждой улице разрешено движение в обе стороны. В городе есть хотя бы одна улица. От каждой площади до любой другой можно дойти по улицам. Почтальону требуется пройти хотя бы один раз по каждой улице. Почтальон хочет, чтобы длина его пути была минимальна. Он может начать движение на любой площади и закончить также на любой (в том числе и на начальной).

Помогите почтальону составить такой маршрут.

Сначала записано число N — количество площадей в городе ($2 \leqslant N \leqslant 1000$). 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 натуральных 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 улицы с площади на нее саму.

Все числа во входном файле не превосходят 100000.

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

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

Если решений несколько, выведите любое.

Input	Output
4 2 2 1 2 2 4 1 2 4 4 3 5 1 1 2 2 5 4 8 2 3 8 2 4	5 1 2 3 4 2 1

N. *Кратчайший путь в невзвешенном графе*

В неориентированном графе требуется найти минимальный путь между двумя вершинами.

Первым на вход поступает число N — количество вершин в графе ($1 \leqslant N \leqslant 100$). Затем записана матрица смежности (0 обозначает отсутствие ребра, 1 — наличие ребра). Далее задаются номера двух вершин — начальной и конечной.

Выведите сначала L — длину кратчайшего пути (количество ребер, которые нужно пройти), а потом сам путь — номера всех вершин в правильном порядке. Если путь имеет длину 0, то его выводить не нужно, достаточно вывести длину. Если пути нет, нужно вывести -1.

Input	Output
5 0 1 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 3 5	3 3 2 1 5

O. *Два коня*

На стандартной шахматной доске (8×8) живут 2 шахматных коня. Им нужно оказаться на одной клетке. Но наши шахматные кони ходят не по очереди, а одновременно, и если оказываются на одной клетке, никто никого не съедает. Сколько ходов им потребуется, чтобы оказаться в одной клетке?

На вход программы поступают координаты коней, записанные по стандартным шахматным правилам (т.е. двумя символами — маленькая латинская буква (от `a` до `h`) и цифра (от 1 до 8), задающие столбец и строку соответственно).

Требуется вывести наименьшее необходимое количество ходов, либо число -1, если кони не могут встретиться.

Input	Output
a1 a3	1

P. *Важные вершины*

Дан неориентированный граф с N вершинами и M рёбрами, не содержащий петель и кратных рёбер. Так же вам дано несколько пар вершин (a_i, b_i). Для каждой пары требуется найти количество вершин c ($c \neq a_i; c \neq b_i$), таких, что любой путь из a_i в b_i, если такой существует, проходит через вершину c.

Путь из a в b — это последовательность вершин, начинающаяся в a и заканчивающаяся в b, такая, что каждые две соседние вершины этой последовательности соединены ребром. Обратите внимание, что путь может проходить по одной и той же вершине более одного раза.

В первой строке через пробел записаны два целых числа N и M — количество вершин и ребер графа, где $1 \leqslant N \leqslant 400; 0 \leqslant M \leqslant \dfrac{n(n-1)}{2}$.

В следующих M строках записано по два целых числа u и v ($1 \leqslant u, v \leqslant n; u \neq v$) — номера вершин, которые соединяет описываемое ребро.

В следующей строке записано единственное целое число Q ($1 \leqslant Q \leqslant 10000$) — количество пар (a_i, b_i). В следующих Q строках описываются пары. В i-й из этих строк через пробел записаны целые числа a_i и b_i ($a_i \neq b_i; 1 \leqslant a_i, b_i \leqslant n$).

Для каждой описанной пары выведите на отдельной строке единственное число — количество вершин c, таких, что любой путь из a_i в b_i, проходит через c.

Input	Output
4 3 1 2 2 3 3 4 2 1 2 1 3	0 1

Q. *Вершины на путях*

Дан неориентированный граф с N вершинами и M рёбрами, не содержащий петель и кратных рёбер. Кроме того дано несколько пар вершин (a_i, b_i). Для каждой пары требуется найти количество вершин c, таких, что существует путь из a_i в b_i, проходящий через вершину c.

Путь из a в b — это последовательность вершин, начинающаяся в a и заканчивающаяся в b, такая, что каждые две соседние вершины этой последовательности соединены ребром. Обратите внимание, что путь может проходить по одной и той же вершине более одного раза.

В первой строке через пробел записаны два целых числа n и m ($1 \leqslant n \leqslant 100; 0 \leqslant m \leqslant n(n-1)/2$) — количество вершин и рёбер графа. В следующих m строках записано по два целых числа u и v ($1 \leqslant u, v \leqslant n; u \neq v$) — номера вершин, которые соединяет описываемое ребро.

В следующей строке записано единственное целое число Q ($1 \leqslant Q \leqslant 10000$) — количество пар (a_i, b_i). В следующих Q строках описываются пары. В i-й из этих строк через пробел записаны целые числа a_i и b_i ($a_i \neq b_i; 1 \leqslant a_i, b_i \leqslant n$).

Для каждой описанной пары выведите на отдельной строке единственное число — количество вершин c, таких, что существует путь из a_i в b_i, проходящий через c.

Input	Output
4 3 1 2 2 3 3 4 2 1 2 1 3	4 4

R. *Циклы через рёбра*

Дан неориентированный граф с N вершинами и M ребрами, не содержащий петель и кратных рёбер. Найдите количество рёбер графа, через которые можно провести цикл.

В первой строке через пробел записаны два целых числа N и M ($1 \leqslant n \leqslant 100; 0 \leqslant m \leqslant n(n-1)/2$) — количество вершин и ребер графа. В следующих M строчках записано по два целых числа a и b ($1 \leqslant a, b \leqslant n; a \neq b$) — номера вершин, которые соединяет описываемое ребро.

Выведите единственное число — количество ребер графа, через которые можно провести цикл.

Input	Output
4 3 1 2 2 3 3 4	0
4 4 1 2 2 3 1 3 3 4	3

S. *Проверка обхода в глубину*

Дан неориентированный граф и некоторая последовательность номеров его вершин. Определить — может ли эта последовательность вершин быть результатом работы программы, перечисляющей вершины графа в порядке его обхода в глубину?

В первой строке через пробел записаны два целых числа N и M ($1 \leqslant n \leqslant 10^5; 0 \leqslant m \leqslant 10^6$ — количество вершин и ребер графа. В следующих M строчках записано по два целых числа a и b ($1 \leqslant a, b \leqslant n; a \neq b$) — номера вершин, которые соединяет описываемое ребро.

В следующей строке записано число Q — количество запросов. Наконец, в каждой из следующих Q строк записана последовательность чисел a_i ($1 \leqslant a_i \leqslant N$).

Программа должна вывести Q строк: в i-й строке следует вывести `YES`, если последовательность a_i может быть получена при выполнении обхода в глубину на данном графе начиная с какой-то вершины в каком-то порядке обхода и `NO`, если не может быть получена.

Input	Output
3 2 1 2 3 2 2 1 2 3 3 1 2	YES NO
4 2 1 2 3 4 1 1 3 2 4	NO

T. *Табличка*

Дана таблица, состоящая из N строк и M столбцов. В каждой клетке таблицы записано одно из чисел: 0 или 1. Расстоянием между клетками (x_1, y_1) и (x_2, y_2) назовем сумму $|x1-x2|+|y1-y2|$. Вам необходимо построить таблицу, в клетке (i, j) которой будет записано минимальное расстояние между клеткой (i, j) начальной таблицы и клеткой, в которой записана 1. Гарантируется, что хотя бы одна 1 в таблице есть.

В первой строке вводятся два натуральных числа N и M, не превосходящих 5000. Далее идут N строк по M чисел — элементы таблицы.

Требуется вывести N строк по M чисел — элементы искомой таблицы.

Input	Output
2 3 0 0 1 1 0 0	1 1 0 0 1 1

U. *Игрушечный лабиринт*

Игрушечный лабиринт представляет собой прозрачную плоскую прямоугольную коробку, внутри которой есть препятствия и перемещается шарик. Лабиринт можно наклонять влево, вправо, к себе или от себя, после каждого наклона шарик перемещается в заданном направлении до ближайшего препятствия или до стенки лабиринта, после чего останавливается. Целью игры является загнать шарик в одно из специальных отверстий — выходов. Шарик проваливается в отверстие, если оно встречается на его пути (шарик не обязан останавливаться в отверстии).

Входные данные В первой строке входного файла записаны числа N и M — размеры лабиринта (целые положительные числа, не превышающие 100). Затем идет N строк по M чисел в каждой — описание лабиринта. Число 0 в описании означает свободное место, число 1 — препятствие, число 2 — отверстие.

Выведите единственное число — минимальное количество наклонов, которые необходимо сделать, чтобы шарик покинул лабиринт через одно из отверстий.

Input	Output
4 5 0 0 0 0 1 0 1 1 0 2 0 2 1 0 0 0 0 1 0 0	3

V. *Только направо*

Змей Горыныч оказался в лабиринте и хочет выбраться из него как можно скорее. Но Змей плохо соображает, поэтому может поворачивать направо и идти прямо, но не может поворачивать налево и разворачиваться на месте. Помогите Змею Горынычу определить длину кратчайшего пути до выхода из лабиринта.

В первой строке через пробел записаны числа r и c ($4 \leqslant r, c \leqslant 20$) — количество строк и столбцов в карте лабиринта. В каждой из следующих r строк записано по c символов, задающих эту карту. Символ S обозначает положение Змея Горыныча, символ F — точку выхода из лабиринта, символ X — стенку. Пробелами обозначены проходимые клетки. Гарантируется, что лабиринт окружен стенами. Перед началом движения Змей Горыныч может сориентироваться по любому из 4 направлений (вверх, вниз, влево или направо).

Выведите единственное число — расстояние, которое придется пройти Змею Горынычу. Гарантируется, что он всегда сможет выйти из лабиринта.

Input	Output
10 14 XXXXXXXXXXXXXX X XXX X XFXXXXX X XXX XX XX X X S X XX XXXXXX X X X X X X X X X X X X XXX XX X XXXXXXXXXXXXXX	29

W. *Наименьшее кратное*

Дано число X и множество цифр D. Требуется дописать к X минимальное количество цифр из D, чтобы получившееся число делилось на k. При этом получившееся число должно быть минимально возможным.

Первая строка входного файла содержит два натуральных числа X и k ($1 \leqslant X \leqslant 10^{1000}, 2 \leqslant k \leqslant 10^5$). Во второй строке записано количество цифр во множестве D. В третьей строке через пробел записаны эти цифры.

Единственная строка должна содержать минимальное число, полученное из X дописыванием цифр из D и кратное k. Если такого числа не существует, выведите -1.

Input	Output
102 101 3 1 0 3	29
202 101 3 1 0 3	202

X. *Алгоритм Дейкстры – кратчайшее расстояние*

Дан ориентированный взвешенный граф. Найдите кратчайшее расстояние от одной заданной вершины до другой.

В первой строке содержатся три числа: N, S и F ($1 \leqslant N \leqslant 100, 1 \leqslant S, F \leqslant N$), где N — количество вершин графа, S — начальная вершина, а F — конечная. В следующих N строках вводится по N чисел, не превосходящих 100, — матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое неотрицательное число — присутствие ребра данного веса. На главной диагонали матрицы записаны нули.

Требуется вывести искомое расстояние или -1, если пути между указанными вершинами не существует.

Input	Output
3 2 1 0 1 1 4 0 1 2 1 0	3

Y. *Алгоритм Дейкстры – кратчайший маршрут*

Дан ориентированный взвешенный граф. Найдите кратчайшее расстояние от одной заданной вершины до другой.

В первой строке содержатся три числа: N, S и F ($1 \leqslant N \leqslant 100, 1 \leqslant S, F \leqslant N$), где N — количество вершин графа, S — начальная вершина, а F — конечная. В следующих N строках вводится по N чисел, не превосходящих 100, — матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое неотрицательное число — присутствие ребра данного веса. На главной диагонали матрицы записаны нули.

Требуется вывести последовательно все вершины одного (любого) из кратчайших путей, или одно число -1, если пути между указанными вершинами не существует.

Input	Output
3 2 1 0 1 1 4 0 1 2 1 0	2 3 1

Z. *Военный поход*

В одном феодальном государстве средневековой Европы было n городов и m дорог, каждая из которых соединяла некоторые два города. Каждая дорога принадлежала правителю одного из городов (не обязательно одного из тех, которые она соединяла). Однажды правитель города S решил объявить войну правителю города T. Перед ним сразу же встала задача: как довести армию до города T.

Правитель каждого города требует плату за проход войск через его город. Кроме того, правитель города S может перемещать войска только по дорогам, которые принадлежат ему. При этом он может купить любую дорогу у её владельца за определенную плату (даже если владелец — правитель города T). К сожалению, все деньги из казны города S были потрачены на предыдущий неудачный военный поход, поэтому сначала правителю придется продать некоторые свои дороги (разумеется, после этого он не сможет провести по ним войска).

Помогите правителю выяснить, какие дороги следует продать, а какие купить, чтобы довести войска от города S до города T и оплатить проход через все промежуточные города. Все операции продажи и покупки дорог надо осуществить до начала похода, пока правители других городов не догадались о воинственных намерениях правителя города S.

В первой строке вводятся целые числа n и m — количество городов и дорог соответственно ($2 \leqslant n \leqslant 2000, 1 \leqslant m \leqslant 50000$). Города нумеруются от 1 до n, города S и T имеют номера 1 и n соответственно.

Следующие n строк содержат под одному целому числу r_i — плату за проезд через город i ($0 \leqslant r_i \leqslant 10000, r_1 = r_n = 0$).

Далее идут m строк, задающих описания дорог. Дорога описывается четырьмя целыми числами: a_i, b_i, p_i и c_i. Здесь a_i и b_i — номера городов, которые соединяет дорога, p_i — номер города, правителю которого она принадлежит, c_i — её стоимость ($a_i \neq b_i, 1 \leqslant c_i \leqslant 10000$). По дороге можно перемещаться в обоих направлениях. Любые два города соединены не более, чем одной дорогой.

В первой строке выведите список дорог, которые нужно продать, в следующем формате: сначала число дорог, а затем их номера. Дороги нумеруются с единицы в том порядке, в котором они заданы во входных данных. Во второй строке выведите список дорог, которые нужно купить, в том же формате. В третьей строке выведите маршрут похода — номера городов в порядке следования войска. Если решений несколько, выведите любое.

Если решения нет, выведите число -1.

Input	Output
3 3 0 1 0 1 2 1 10 2 3 1 10 3 1 2 2	2 1 2 1 3 1 3

ZA. *Минимальное остовное дерево*

Требуется найти в связном графе остовное дерево минимально веса.

Первая строка входного файла содержит два натуральных числа N и M — количество вершин и рёбер графа соответственно ($1 \leqslant n \leqslant 20000, 0 \leqslant M \leqslant 100000$). Следующие M строк содержат описание рёбер по одному на строке. Ребро номер i описывается тремя натуральными числами b_i, e_i и w_i — номера концов ребра и его вес соответственно ($1 \leqslant b_i, e_i \leqslant n, 0 \leqslant w_i \leqslant 100000$).

Граф является связным.

Выведите единственное целое число — вес минимального остовного дерева.

Input	Output
4 4 1 2 1 2 3 2 3 4 5 4 1 4	7

ZB. *Конденсация графа*

Вам задан ориентированный граф с N вершинами и M ребрами ($1 \leqslant N \leqslant 20000, 1 \leqslant M \leqslant 200000$). Найдите компоненты сильной связности заданного графа и топологически отсортируйте его конденсацию.

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.

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.

Input	Output
10 14 4 9 7 8 2 5 1 4 9 2 10 6 6 5 8 3 5 9 4 3 8 7 5 1 2 1 6 10	4 3 3 4 3 3 2 1 1 3 2

ZC. *Алфавит*

Страна X использует при письме маленькие буквы латинского алфавита. Однако упорядочивают их в другом порядке.

Вам дан набор слов. Необходимо восстановить порядок букв в алфавите таким образом, чтобы данные слова оказались упорядоченными по возрастанию в лексикографическом порядке. Если порядок существует, но его нельзя восстановить однозначно — нужно вернуть любой подходящий порядок букв. Если не существует порядка букв, удовлетворящего условию задачи — вывести -1.

В первой строке входных данных указано количество слов ($N < 1000$). Затем N строк с одним словом на каждой. Суммарная длина слов не превосходит 10000.

Программа должна вывести одну строку: упорядоченную последовательность букв без разделителей, либо -1.

В последовательности букв должны встретиться обязательно все буквы из набора слов и никаких других букв.

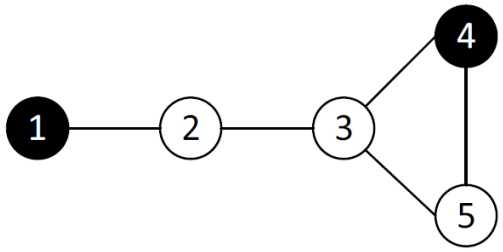
Input	Output
3 abc bc bb	acb

ZD. *Безопасность данных*

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 иллюстрации ниже 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 $10^9 + 7$.

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.

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 $10^9 + 7$.

Input	Output
5 5 1 2 2 3 3 4 3 5 4 5	2 3