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 ребер. Затем следует M пар чисел — ребра 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. Треубется 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 невзвешенный 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 невзвешенный 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.

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 ($2 \leqslant N \leqslant 100$) и E — количество пар станций, расстояния между которыми заданы ($0 \leqslant E \leqslant 10000$). 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