CSE221
Lab Practice Sheet 04
Fall 2024

A useful tool for making graphs:

https://csacademy.com/app/graph_editor/

You can find all the Input Files of this lab from here:

https://drive.google.com/drive/folders/10grRR3iRIfVn6kFLyD86-n8xQewUNU9n?usp=sharing

Task 01 (Graph Representation):

A) You are given a **directed weighted** Graph, G. The graph consists of N vertices and M edges. In this problem, you have to store the graph using the Adjacency Matrix and print the matrix.

Input:

The first line contains two integers N and M ($1 <= N, M <= 100$) — the number of vertices and the total number of edges.
The next M lines will contain three integers u_i, v_i, and w_i ($1 <= u_i, v_i <= N$, $1 <= w_i <= 1000$) — denoting there is an edge between node u_i and v_i with cost w_i.

Output:

You have to make the graph using the Adjacency Matrix and print the Adjacency Matrix in the output.

Sample Input/Output:

Sample Input 1	Sample Output 1

4 3
1 3 8
3 2 5
1 4 2
	0 0 0 0 0
	0 0 0 8 2
	0 0 0 0 0
	0 0 5 0 0
	0 0 0 0 0

Sample Input 2	Sample Output 2

6 7
1 5 6
6 3 5
1 3 9
3 4 7
4 6 1
5 6 8
6 1 6
	0 0 0 0 0 0 0
	0 0 0 9 0 6 0
	0 0 0 0 0 0 0
	0 0 0 0 7 0 0
	0 0 0 0 0 0 1
	0 0 0 0 0 0 8
	0 6 0 5 0 0 0

B) Previously, we have learned how to represent a graph using the Adjacency matrix. Now in this problem, using the information in Task 01 (a), you have to build the graph using the Adjacency List.

Sample Input/Output:

Sample Input 1	Sample Output 1

4 3
1 3 8
3 2 5
1 4 2
	0 :
	1 : (3,8) (4,2)
	2 :
	3 : (2,5)
	4 :

Sample Input 2	Sample Output 2

6 7
1 5 6
6 3 5
1 3 9
3 4 7
4 6 1
5 6 8
6 1 6
	0 :
	1 : (5,6) (3,9)
	2 :
	3 : (4,7)
	4 : (6,1)
	5 : (6,8)
	6 : (3,5) (1,6)

Sample Input 3	Sample Output 3

9 15
1 9 4
8 7 5
5 4 7
4 3 2
	0 :
	1 : (9,4)
	2 : (1,10)
	3 : (1,15) (9,8) (8,4) (7,3)
	4 : (3,2)

2 1 10
3 1 15
7 1 13
3 9 8
3 8 4
3 7 3
5 3 1
5 8 2
9 4 8
9 2 10
9 2 12
	5 : (4,7) (3,1) (8,2)
	6 :
	7 : (1,13)
	8 : (7,5)
	9 : (4,8) (2,10) (2,12)

Brainstorming: [No need to submit the following questions]

a) What if it was an **undirected** graph? Can you think of where you have to modify your code for an undirected graph?

b) In the last input of Task 01 (B), you notice there are two edges from node ‘9’ to node ‘2’ of cost 10 and 12, known as **parallel edges**. If a graph contains parallel edges, then can we represent the graph using the Adjacency Matrix?

<u>Task 02 (Graph Traversal Using BFS):</u>

You have most probably heard of Dora The Explorer. She likes to travel from one country to another.

Currently, Dora is at Doraland. Doraland is a beautiful country, consisting of N cities and M **bidirectional** roads.

Recently, Dora has started to learn Graph Algorithms. She knows about BFS and DFS. However, since Dora is still learning, she is not feeling so confident and asks for your help.

Dora contacts you and gives you all the necessary information about Doraland. **Dora will start her journey from City 1.** Now, your task will be to help Dora find a path that is similar to the path of BFS graph traversal.

Input

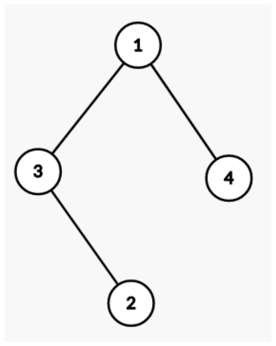
The given map will be an undirected and unweighted graph.

The first line contains two integers N and M (1 <= N, M <= 100) — the number of cities and the total number of roads.
The next M lines will contain two integers u_i, v_i (1 <= u_i, v_i <= N) — denoting there is a bidirectional road between u_i and v_i.

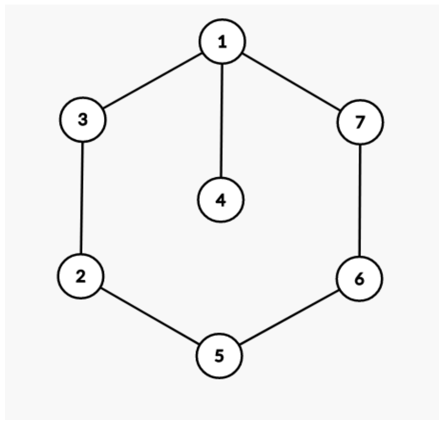
Output

Print the BFS path traversal which Dora will follow to explore the city, starting from city 1.

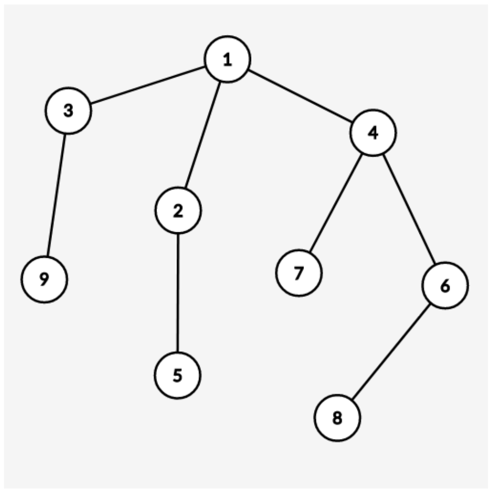
Sample Input 1	Sample Output 1	Sample Graph 1

4 3
1 3
3 2
1 4
	1 3 4 2
	
	(Another valid path:
	1 4 3 2)
		

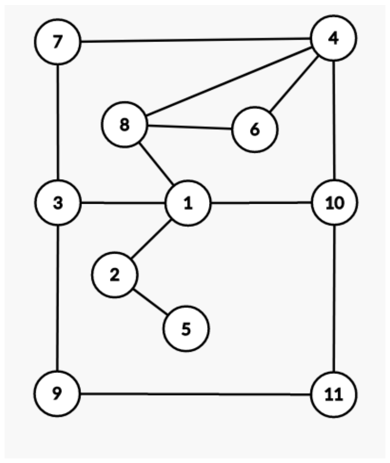
Sample Input 2	**Sample Output 2**	**Sample Graph 2**

7 7
1 3
3 2
1 4
2 5
5 6
1 7
7 6
	1 3 4 7 2 6 5	

Sample Input 3	**Sample Output 3**	**Sample Graph 3**

9 8
1 2
1 3
1 4
2 5
4 6
4 7
6 8
3 9
	1 2 3 4 5 9 6 7 8	

Sample Input 4	Sample Output 4	Sample Graph 4

11 14
1 2
1 3
10 4
2 5
4 6
4 7
6 8
3 9
3 7
1 8
4 8
1 10
9 11
10 11
	1 2 3 8 10 5 9 7 6 4 11	

This problem may have multiple valid solutions, however, your solution should follow the BFS level order traversal.

Pseudocode of BFS:

The breadth-first-search procedure BFS below assumes that the input graph G = (V, E) is represented using adjacency lists.

```
BFS(G,s):
1 for each vertex u in G.V:
    u.colour = 0
Q = Ø ;
s.colour = 1
ENQUEUE(Q,s)
while Q ≠ Ø;
    u = DEQUEUE(Q)
    for each v in G.Adj[u]:
        if v.colour == 0:
            v.colour = 1
            ENQUEUE(Q,v)
```

<u>Task 03 (Graph Traversal Using DFS):</u>

After successful BFS traversal, Dora wants to visit the whole country again. However, this time Dora wants to traverse the country following the path that is similar to the path of DFS graph traversal.

Yes, you guessed it correctly. You have to help Dora this time also.

The input format remains the same as Task 02. **Dora will start her journey from City 1 again.**

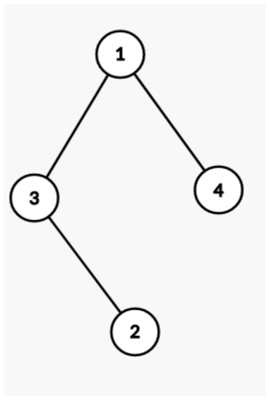
Input

The given map will be an undirected and unweighted graph.
The first line contains two integers N and M (1 <= N, M <= 100) — the number of cities and the total number of roads.
The next M lines will contain two integers u_i, v_i (1 <= u_i, v_i <= N) — denoting there is a bidirectional road between u_i and v_i.

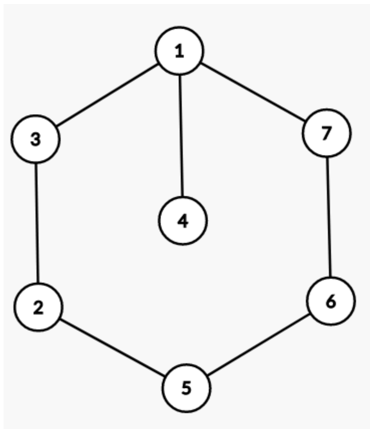
Output

Print the DFS path traversal which Dora will follow to explore the city, starting from city 1.

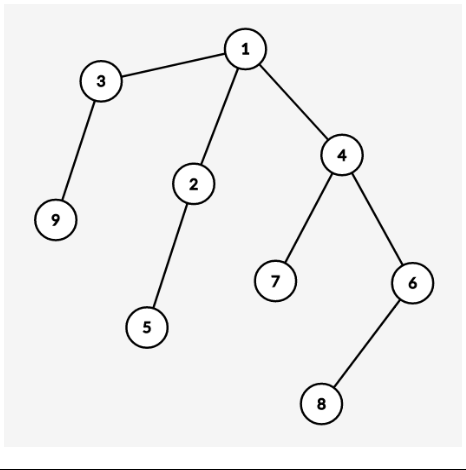
Sample Input 1	Sample Output 1	Sample Graph 1

4 3
1 3
3 2
1 4
	1 3 2 4
	
	(Another valid path:
	1 4 3 2)
		

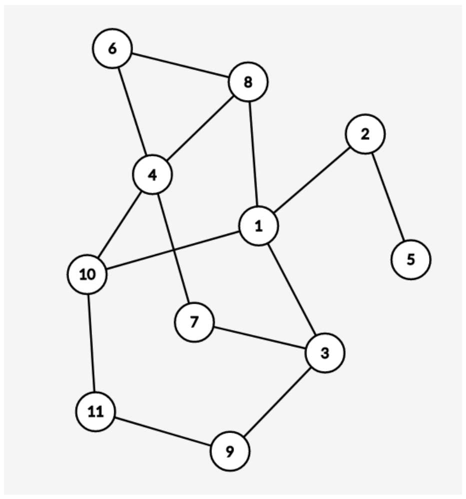
Sample Input 2	Sample Output 2	Sample Graph 2

7 7
1 3
3 2
1 4
2 5
5 6
1 7
7 6
	1 3 2 5 6 7 4	

Sample Input 3	Sample Output 3	Sample Graph 3

9 8
1 2
1 3
1 4
2 5
4 6
4 7
6 8
3 9
	1 2 5 3 9 4 6 8 7	

Sample Input 4	Sample Output 4	Sample Graph 4

11 14
1 2
1 3
10 4
2 5
4 6
4 7
6 8
3 9
3 7
1 8
4 8
1 10
9 11
10 11
	1 2 5 3 9 11 10 4 6 8 7	

This problem may have multiple valid solutions. However, your solution should follow the DFS path traversal order.

Pseudocode of DFS:

The depth-first-search procedure DFS below assumes that the input graph G = (V, E) is represented using adjacency lists.

```
colourInitializing(G):
for each vertex u in G.V:
   u.colour = 0

DFS(G,u):
   u.colour = 1
   for each v in G.Adj[u]:
        if v.colour == 0:
            DFS(G,v)
```

Task 04 (Cycle Finding):

It was a very hectic day for Dora. Before going to sleep, Dora wants to find out if there is any Cycle in the map of the city.

Since Dora has traveled the whole country twice, she is feeling very exhausted and asks you to solve the problem. However, Dora has made the roads of the cities **directed** in her map, so that you don’t get bored.

In graph theory, a cycle in a graph is a non-empty trail in which only the first and last vertices are equal. [Wikipedia]

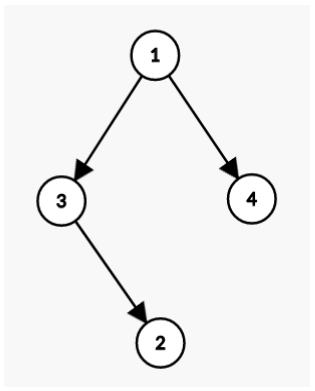
Input:

The given map will be a directed and unweighted graph.
The first line contains two integers N and M (1 <= N, M <= 100) — the number of cities and the total number of roads.
The next M lines will contain two integers u_i, v_i (1 <= u_i, v_i <= N) — denoting there is a bidirectional road between u_i and v_i.

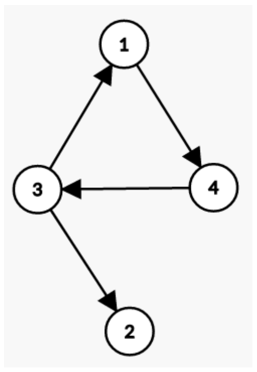
Output:

Print “YES” if the map contains any Cycle, otherwise print “NO”.

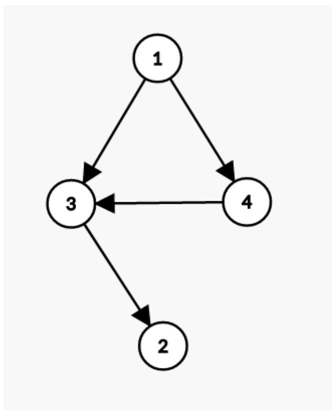
Sample Input 1	Sample Output 1	Sample Graph 1

4 3
1 3
3 2
1 4
	NO	

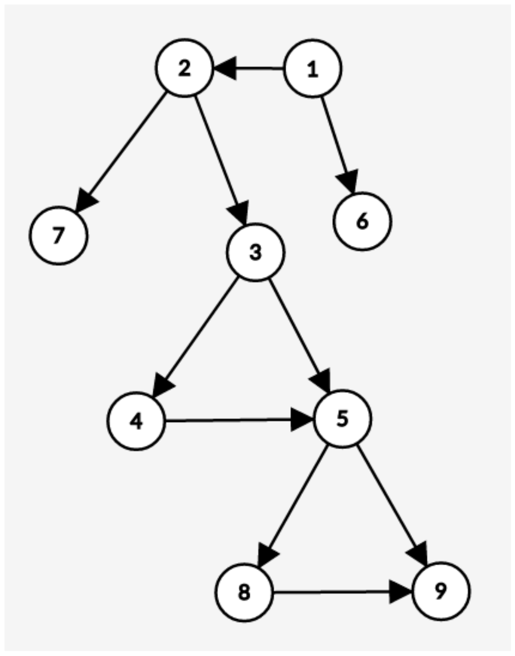
Sample Input 2	Sample Output 2	Sample Graph 2

4 4
3 1
3 2
1 4
4 3
	YES	

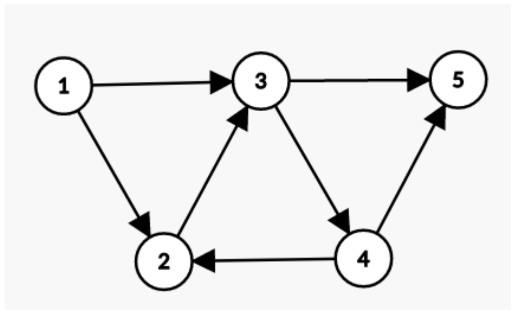
Sample Input 3	Sample Output 3	Sample Graph 3

4 4
1 3
3 2
1 4
4 3
	NO	

Sample Input 4	Sample Input 4	Sample Graph 4

9 10
1 2
2 3
3 4
4 5
3 5
1 6
2 7
5 8
5 9
8 9
	NO	

Sample Input 5	Sample Input 5	Sample Graph 5

5 7
1 2
2 3
1 3
3 4
4 5
3 5
4 2
	YES	

Brainstorming: [No need to submit the following questions]

a) Suppose that you have an undirected and unweighted graph, can you find out if the given graph is a tree or not?

b) Suppose that you have an undirected and unweighted graph. Can you find out if the graph contains an Odd-length Cycle? An odd-length cycle means the cycle will contain odd numbers of vertices.

Task 05 (Find the shortest path):

The next day, Dora again called you to help her. Among all the cities she likes city 'D' the most.

Since you are becoming very much annoyed with Dora, you have planned to reach city D spending the minimum amount of time.

In the belugaland, moving between two cities connected by a road takes one second. You have to find the minimum time to reach D and show the path in the output.

Dora and you will start your journey from City 1.

Input

The given map will be an undirected and unweighted graph.
The first line contains three integers N, M, and D (1 <= N, M, D <= 100) — the number of cities, the total number of roads, and the destination city.

The next M lines will contain two integers u_i, v_i ($1 <= u_i$, $v_i <= N$) — denoting there is a bidirectional road between u_i and v_i.

Output

Print the minimum amount of time to reach city D from city 1. Next, print the shortest path you will follow.

Sample Input 1	Sample Output 1

4 3 2
1 3
3 2
1 4
	Time: 2
	Shortest Path: 1 3 2

Sample Input 2	**Sample Output 2**

6 6 5
1 3
3 2
1 4
2 6
5 6
4 6
	Time: 3
	Shortest Path: 1 4 6 5

Sample Input 3	**Sample Output 3**

9 8 7
1 2
1 3
1 4
2 5
4 6
4 7
6 8
3 9
	Time: 2
	Shortest Path: 1 4 7

Sample Input 4	**Sample Output 4**

11 14 5
4 2
1 3
10 4
2 5
4 6
4 7
6 8
	Time: 4
	Shortest Path: 1 8 4 2 5

3 9
3 7
1 8
4 8
1 10
9 11
10 11	

Sample Input 5	Sample Output 5

4 3 1
1 3
3 2
1 4
	Time: 0
	Shortest Path: 1

This problem may have multiple valid shortest paths. However, the minimum time required to reach the destination should be matched exactly with the output.

Task 06 (Flood Fill):

Dora is leaving the country. Before leaving the country, as a token of gesture, Dora gifted you with a map of the jungle "Jumanji".

The map is very interesting. The map is a 2D grid. Some of those cells are occupied by diamonds and obstacles. [See the Sample input for better understanding.]

Now, you are on a journey to Jumanji in search of Diamonds. However, since the jungle is full of ferocious creatures, you can't collect all the diamonds. **You will choose only one start position such that you collect the maximum amount of diamonds. Please note that you can not move to any cell which contains obstacles.**

Input

The first line contains two integers R and H (1 <= R, H <= 100) — the number of rows and columns respectively in the grid.

The next R line will contain H characters. Each character represents the status of a cell as follows.

1) '.': Empty Cell → You can move here.

2) 'D': Cell with a Diamond → If you move to the cell containing ‘D’, you will collect that Diamond.
3) '#': Cell with an obstacle → You can’t move to this cell.

Output

Print a single integer that denotes the maximum amount of Diamonds you can collect.

[The diamonds are coloured red to demonstrate which diamonds have been collected.]

Sample Input 1	Sample Output 1
4 3 ..D D.. .D. ##.	3
Sample Input 2	**Sample Output 2**
10 7 ...#..D ...#D.. D..#.D. D..#... DDD#### ....####.. .#D.#DD .####.. DDD...D	11
Sample Input 3	**Sample Output 3**
9 11 .#..D...D.. .#.#######. D#.#..D..#. D#D#.###.#D .#.#..D#.#. .#.#####.#D D#..D...D#. .#########. ...D..D...D	15
Sample Input 4	**Sample Output 4**

5 5 ..... ####. #D.#. ####.	1
Sample Input 5	Sample Output 5
5 5 ..... ####. #..#. ####.	0
Sample Input 6	Sample Output 6
1 5 D....	1
Sample Input 7	Sample Output 7
12 12 ....####....... .#D.#....... .####....... ....###. ...D....#D#. ....#D#. .########D#. .#D....##.#. .#.D..D##D#. .##########.	4

Task 07:

Dora returns to Doraland once again. She finds that the city has undergone significant improvements since her last visit. The city structure has been reconstructed to make it more beautiful. Currently, **Doraland consists of N cities, and all the cities are connected with N-1 bidirectional roads.**

However, this time, Dora will have a very short trip, and as usual, she seeks your help. According to Dora's wish, you will choose two cities, let's say city A and city B in such a way that while traveling from city A to city B Dora can pass through a maximum number of cities. You have to keep in mind that it is not possible to visit any roads twice in the journey since Dora is on a short trip.

Can you help Dora find such two cities?

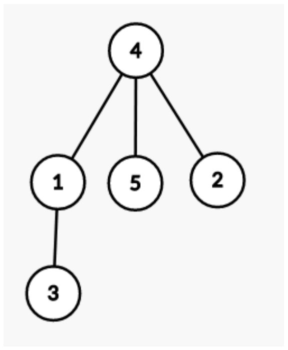
Input

The given map will be an undirected and unweighted graph.
The first line contains one integer N (2 <= N <= 100) — the number of cities.
The next N-1 lines will contain two integers u_i, v_i (1 <= u_i, v_i <= N) — denoting there is a bidirectional road between u_i and v_i.

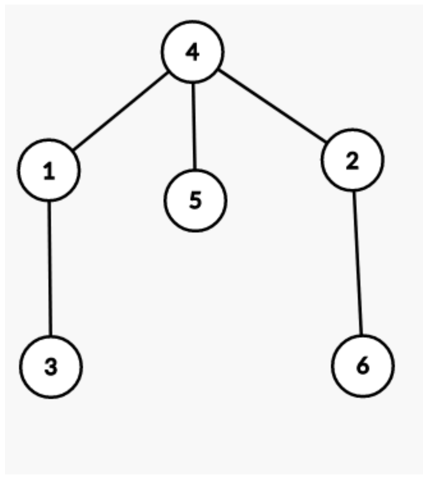
Output

Print two cities A and B which satisfy the conditions stated in the problem statement.

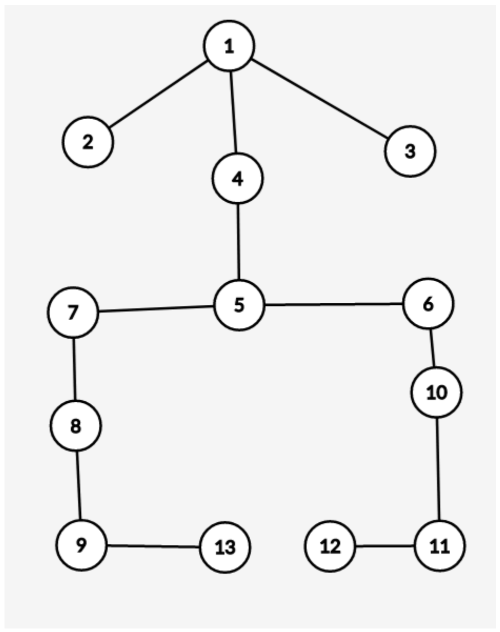
Sample Input 1	Sample Output 1	Sample Graph 1

5
4 1
4 2
1 3
4 5
	2 3
	
	Or (3 5)
	
	[You may print only one valid output]
		

Sample Input 2	Sample Output 2	Sample Graph 2

6
4 1
4 2
1 3
4 5
2 6
	3 6	

Sample Input 3	Sample Output 3	Sample Graph 3

13
1 2
1 3
1 4
4 5
5 6
5 7
7 8
8 9
9 13
6 10
10 11
11 12
	12 13	

This problem may have multiple valid answers. However, make sure it satisfies the given condition.

Explanation:

In test case 1, if you choose city 2 and city 3, then the path will be
2 → 4 → 1 → 3. If you choose city 3 and city 5, then the path will be
3 → 1 → 4 → 5.

Since, in both paths you can visit a maximum of four cities by using the roads only one, both the answers are valid.

In test case 2, one can see that visiting from city 3 to city 6 (or vice-versa) is only the valid answer. The path will be 3 → 1 → 4 → 2 → 6.

Task 08:

As mentioned earlier, Dora is on a short trip and she is leaving the country. Before leaving the country, as a token of gesture, Dora gifted you with a problem this time, which you will find in this link in the lightoj, an online judge.

To open the gift, you have to create an account at lightoj first. [https://lightoj.com/]

Problem Link: https://lightoj.com/problem/back-to-underworld

Task 09

We all know about the prerequisite course. For example, before taking CSE221, we have to complete CSE220. And prior to taking CSE220, we need to complete CSE111 and CSE110. The course sequence is as follows:

CSE110 → CSE111 → CSE220 → CSE221

In this problem, there are **N** courses in the curriculum. There are **M** prerequisite requirements of the form "Course **A** has to be completed before course **B**".

Your task is to find an order in which you can complete the courses.

A) Solve it using the DFS approach.
B) Solve it using the BFS approach.

Input:

The first input line has two integers **N** (1 <= N <= 1000) and **M** (1 <= M <= N^2) - the number of courses and prerequisite requirements. The courses are numbered 1,2,3,…, N.

Next, M lines describe the requirements. Each line has two integers **A**, **B** (1 <= A, B <= N)- course A has to be completed before course B.

Output:

Print an order in which you can complete the courses. Please note, that there could be multiple correct sequences. You can print any valid order that includes all the courses.

If there are no solutions, print "IMPOSSIBLE".

Sample Input 1	Sample Output 1
5 3	3 4 1 5 2
3 1	
1 2	
4 5	
Sample Input 2	**Sample Output 2**
6 6	IMPOSSIBLE
1 2	
2 3	
4 3	
4 5	
5 6	
6 4	
Sample Input 3	**Sample Output 3**
8 10	1 7 2 8 4 6 5 3
1 2	
1 4	
2 4	
2 5	
2 3	
4 6	
4 5	
6 5	
5 3	
7 8	

Task 10

The problem statement of this problem is the same as **Task 09.** The only difference is that in this task, you have to find the lexicographically smallest valid course sequence.

If you have two sequences, for example, A: 3 → 1 → 2 → 4 and B: 3 → 1 → 4 → 2. Then path A is lexicographically smaller than path B.

Input:

The first input line has two integers **N** (1 <= N <= 1000) and **M** (1 <= M <= N^2) - the number of courses and prerequisite requirements. The courses are numbered 1,2,3,…, N.

Next, there are M lines describing the requirements. Each line has two integers **A**, **B** (1 <= A, B <= N)- course A has to be completed before course B.

Output:

Print the lexicographically smallest valid course sequence in which you can complete the courses.

If there are no solutions, print "IMPOSSIBLE".

Sample Input 1	Sample Output 1

5 3
3 1
1 2
4 5
	3 1 2 4 5

Sample Input 2	**Sample Output 2**

6 6
1 2
2 3
4 3
4 5
5 6
6 4
	IMPOSSIBLE

Sample Input 3	Sample Output 3

8 10
1 2
1 4
2 4
2 5
2 3
4 6
4 5
6 5
5 3
7 8
	1 2 4 6 5 3 7 8

[Strongly Connected Component (SCC) has been excluded from the lab for this semester. We recommend that you self-study the topic and solve task 11.]

Task 11

You are given a Directed Graph consisting of **N** vertices and **M** edges. You have to find the strongly connected components of the given graph.

Input:

The given map will be a directed and unweighted graph.
The first line contains two integers N and M (1 <= N, M <= 100) — the number of vertices and the total number of edges.
The next M lines will contain two integers u_i, v_i (1 <= u_i, v_i <= N)— denoting there is a road between ui to vi.

Output

Print all the strongly connected components of the given graph. See the output for better understanding.

Sample Input 1	Sample Output 1

5 5
1 2
2 3
2 4
3 1
4 5
	1 2 3
	4
	5

Sample Input 2	Sample Output 2

8 9
1 2
1 6
2 3
3 4
4 5
4 6
5 7
6 2
7 5
	1
	2 3 4 6
	5 7
	8

Sample Input 3	Sample Output 3

4 3
1 2
2 3
2 4
	1
	2
	3
	4