CSE221_LabAssignment_04_Fall2022

Submission Guidelines :

1. You can code all of them either in Python, CPP, or Java. But you should choose a specific language for all tasks.
2. For each task write separate python files like task2.py, task3.py, and so on. For problems that have subproblems, name those like task1A.py, task1B.py, and so on.
3. For each problem, take input from files called **"inputX_Y.txt"** and output at **"outputX_Y.txt"**, where X is the task number and Y is the sample i/o number. For example, for problem 2 sample 1, the input file is this, "input2_1.txt". For problems that have subproblems, name the files like "input1a_1.txt", "output1a_1.txt" and so on. Same for output.
4. For each task include the input files (if any) in the submission folder.
5. Finally zip all the files and rename this zip file as per this format:**LabSectionNo_ID_CSE221LabAssignmentNo_Fall2022.zip** [Example:**LabSection01_21101XXX_CSE221LabAssignment02_Fall2022.zip**]
6. Don't copy from your friends.
7. You MUST follow all the guidelines, naming/file/zipping convention stated above.
   Failure to follow instructions will result in a straight 50% mark deduction.

A useful tool for making graphs:

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

You can find all the Input Files of Lab04 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 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 Adjacency Matrix, and print the Adjacency Matrix in 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 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 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 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	0 : 1 : (9,4) 2 : (1,10) 3 : (1,15) (9,8) (8,4) (7,3) 4 : (3,2) 5 : (4,7) (3,1) (8,2) 6 : 7 : (1,13) 8 : (7,5) 9 : (4,8) (2,10) (2,12)

Brainstorming:

   a) What if it was an **undirected** graph? Can you think of where
      you have to modify in 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 Adjacency Matrix?

Task 02 (Graph Traversal Using BFS):

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 to find a path which is similar
to the path of BFS graph traversal.

Input
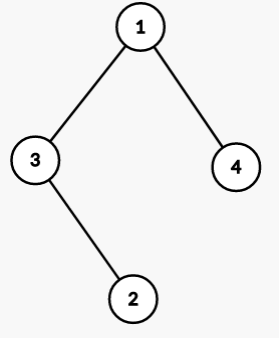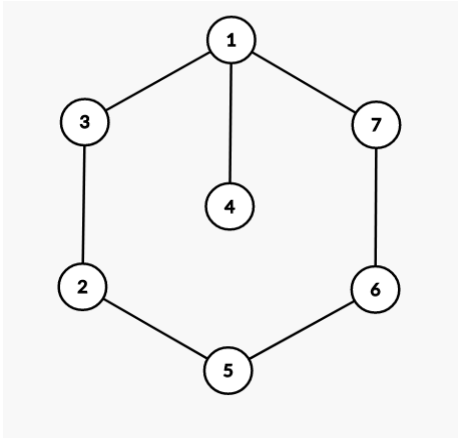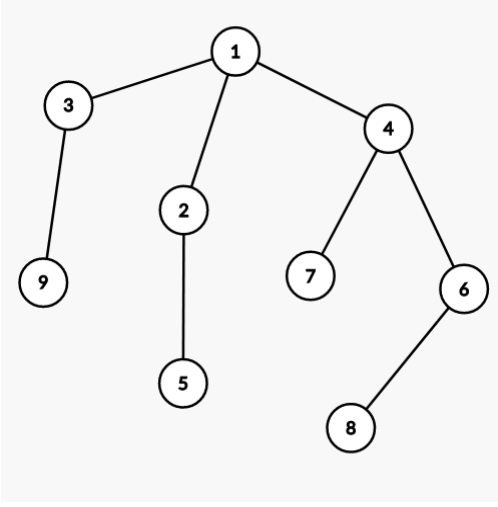
The given map will be an undirected and unweighted graph.
The first line contains two integers N and M (1 <= N, M <= 100)
— the number of cities and the total number of roads.
The next M lines will contain two integers u_i, v_i (1 <= u_i, v_i <=
N) — denoting there is a bidirectional road between u_i 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	1 2 3 8 10 5 9 7 6 4 11	

```
2 5
4 6
4 7
6 8
3 9
3 7
1 8
4 8
1 10
9 11
10 11
```
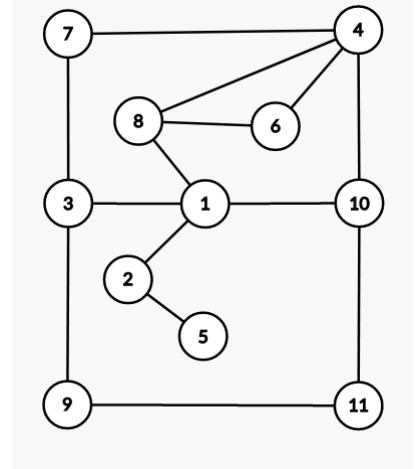
Pseudocode of BFS:

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:
2     u.colour = 0
3 Q = Ø ;
4 s.colour = 1
5 ENQUEUE(Q,s)
6 while Q ≠ Ø;
7     u = DEQUEUE(Q)
8     for each v in G.Adj[u]:
9             if v.colour == 0:
10                v.colour = 1
11                ENQUEUE(Q,v)
```

Task 03 (Graph Traversal Using DFS):

After successful BFS traversal, Dora wants to visit the whole
country again. However, this time Dora wants to traverse the
country following the path which 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
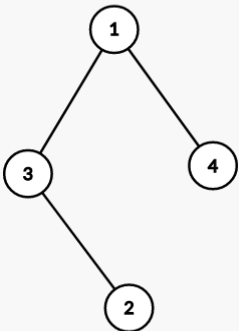her journey from city 1 again**.

Input
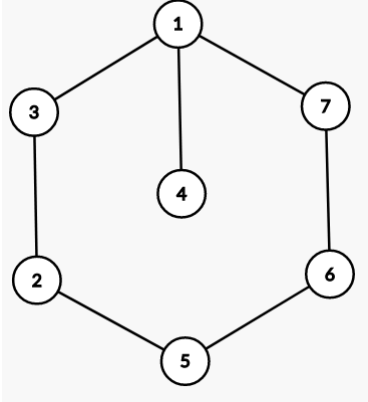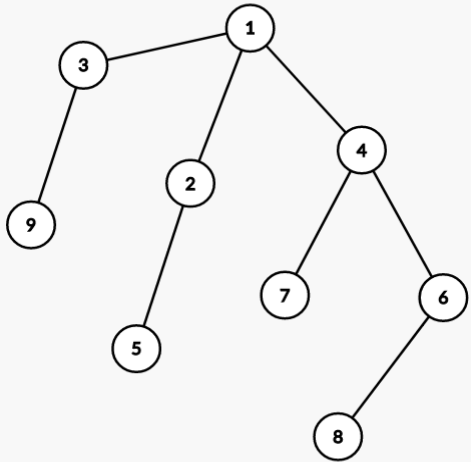
The given map will be an undirected and unweighted graph.
The first line contains two integers N and M (1 <= N, M <= 100)
— the number of cities and the total number of roads.
The next M lines will contain two integers u_i, v_i (1 <= u_i, v_i <=
N) — denoting there is a bidirectional road between u_i 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	1 3 2 5 6 7 4	

7 6		
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	
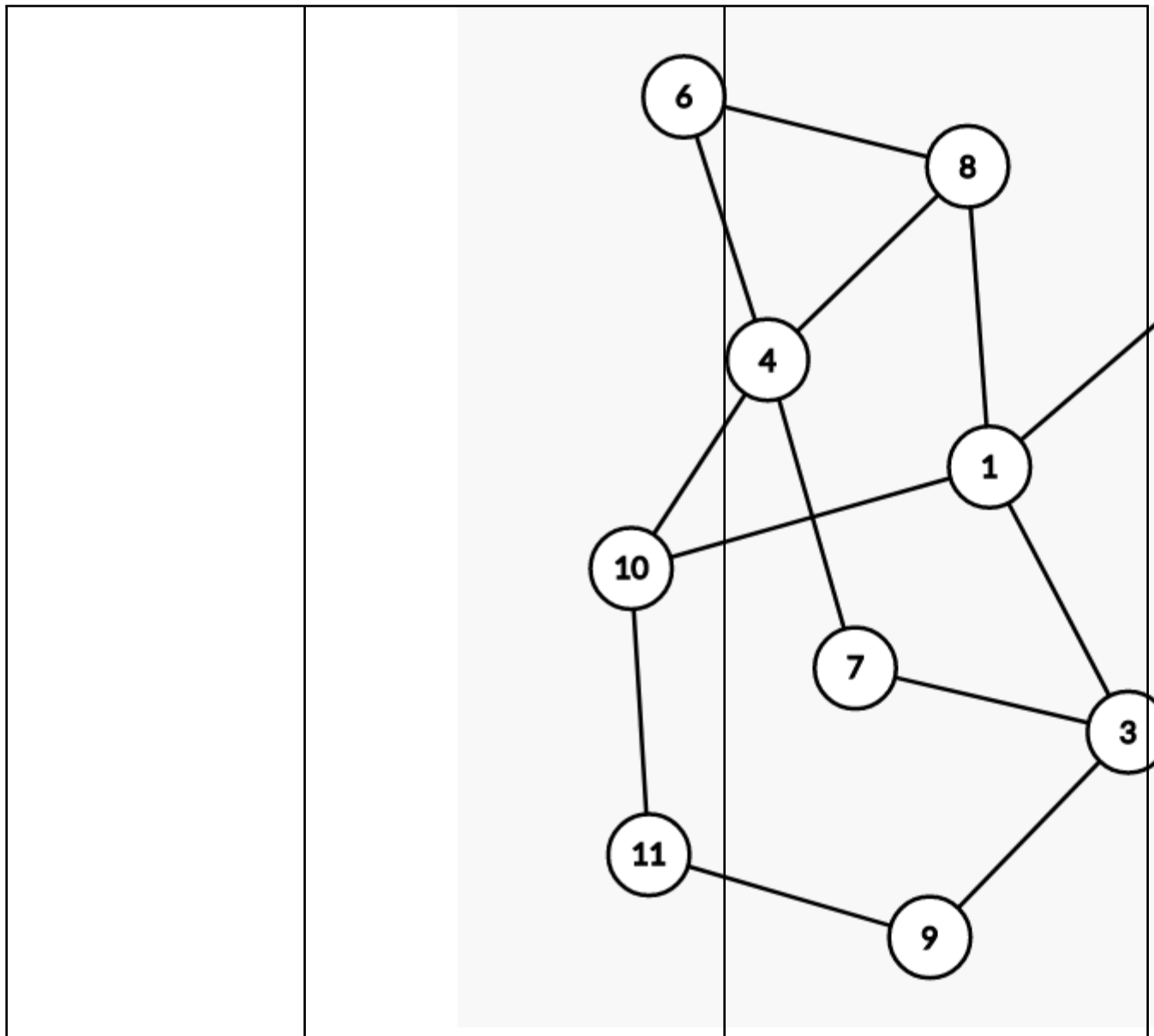
Pseudocode of DFS:

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

```
colourInitializing(G):
1 for each vertex u in G.V:
```

```
2      u.colour = 0

DFS(G,u):
1      u.colour = 1
2      for each v in G.Adj[u]:
3          if v.colour == 0:
4              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:
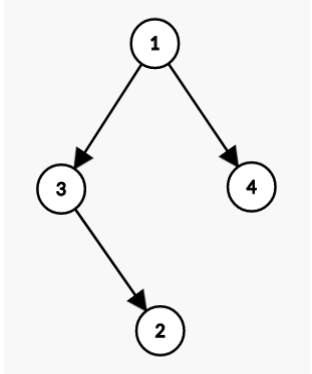
The given map will be a directed and unweighted graph.
The first line contains two integers N and M (1 <= N, M <= 100)
— the number of cities and the total number of roads.
The next M lines will contain two integers u_i, v_i (1 <= u_i, v_i <=
N) — denoting there is a bidirectional road between u_i and v_i.
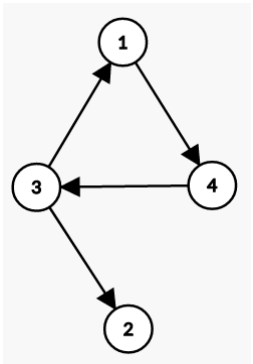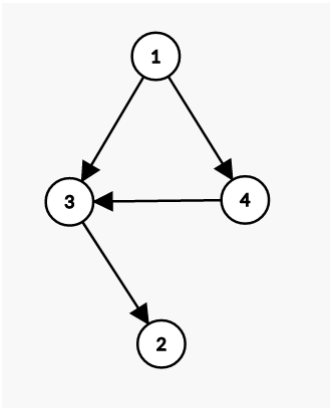
Output:

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	NO	

5 8 5 9 8 9		 Sample Graph 5
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:

    a) 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 for helping 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	Time: 2 Shortest Path: 1 3 2

1 4	
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 3 9 3 7 1 8 4 8 1 10 9 11 10 11	Time: 4 Shortest Path: 1 8 4 2 5
Sample Input 5	Sample Output 5
4 3 1 1 3 3 2 1 4	Time: 0 Shortest Path: 1

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