January 2023 CSE106

Offline Assignment on Graph

Deadline: Saturday, 19 August 2023, 11:45 PM

Changelog

Modification	Time
The solution of Task 3 must have a time complexity of `O(n*m*q)`	13 August

For **all the tasks** , you have to take input from a file named **`in.txt`** and give output to a file **`out.txt`** .

Task 1

Given a **uni-directional** graph `G=(V,E)` , you have to report, if given two vertices, the **second** vertex is reachable from the **first** vertex.

You may assume there is no `multi-edge` or `self-loop` in the graph. Please use **adjacency List** representation of graph for task **1** and **2** .

Solve the problem using **BFS**.

Input

The first line of input contains two integers `n and m` (`1≤n≤10^5` , `1≤m≤10^5`) — the number of vertices and the number of edges.

The following m line each has two integers `u and v` — there is a directed edge from u to v.

The final line contains two integers `s and d` — the source and the destination.

Ouput

If there is a path from `s` to `d` , print the path length. Finally the second line prints the path itself.

If no path exists print `-1` .

Sample I/O

Test case 1

in.txt

```
6 6
0 1
0 2
1 2
2 0
2 3
4 5
0 3
```

out.txt

```
2
0 2 3
```

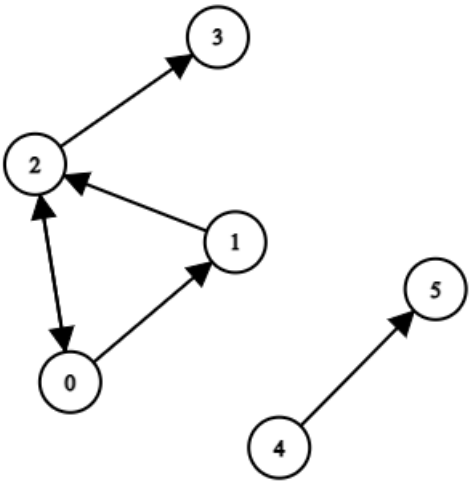
Test case 2

in.txt

```
6 6
0 1
0 2
1 2
2 0
2 3
4 5
3 0
```

out.txt

```
-1
```



You may result in a **different path** (path length **remains the same**) based on your implementation.

Task 2

Solve the same problem using **DFS** .

Sample I/O

Test case 1

in.txt

```
 6 6
0 1
0 2
1 2
2 0
2 3
4 5
0 3
```

out.txt

```
 3
0 1 2 3
```

Here you may have a **different path** with **different path length** .

Task 3

In this task, we will model a real-life problem using graphs and solve it efficiently.

Do you remember chess? Especially the piece **knight**. A traditional knight has up to 8 moves from a square with coordinates `(x, y)` to squares `(x+1, y+2)`, `(x+1, y-2)`, `(x+2, y+1)`, `(x+2, y-1)`, `(x-1, y+2)`, `(x-1, y-2)`, `(x-2, y+1)`, `(x-2, y-1)`, and can't move outside the chessboard. We build a new piece of our own and call it the **rider**. A rider can jump like a knight several times in **a single move**. A rider that can perform a maximum of K jumps during a single move is denoted as a K-rider. For example, a 2-rider can jump once or twice during a single move, and a 1-rider is a traditional knight.

There are **some riders of different types** on a chessboard. Find the **minimal total number of moves** necessary to move all the riders to the same square(can be any square on the chessboard). **Only one piece can move during each move** . Multiple riders can share the same squares during the process. Print `-1` if it is impossible.

Input

The first line of input contains three integers `n, m and q` (`1≤n≤100`, `1≤m≤100`, `1≤q≤n*m`) — the number of rows, the number of columns and the number of k-rider on the chessboard.

The following q line each has three integers `x, y and k` — `(x,y)` the coordinates of the rider and `k` is the number of **maximum jumps** this k-rider can make in **a single move**.

Here `(x,y)` are `0`-indexed. Please refer to the sample IO for a better understanding of the input.

Output

Output the total minimal number of moves, if we can move all k-rider to the same square. Else print -1.

Please notice, **The solution must have a time complexity of `O(n*m*q)`**

Sample I/O

Test case 1

in.txt

```
3 3 3
0 0 1
0 2 1
2 2 1
```

out.txt

```
4
```

Test case 2

in.txt

```
1 4 2
0 0 1
0 3 1
```

out.txt

```
-1
```

Submission guideline

1. Create a directory with your 7-digit student id as its name
2. item Put the source files only into the directory created in step 1
3. item Zip the directory (compress in .zip format; .rar, .7z or any other format is not acceptable)
4. item Upload the .zip file on Moodle.

For example, if your student id is 2105xxx, create a directory named 2105xxx. Put only your source files (.c, .cpp, .java, .h, etc.) into 2105xxx. Compress 2105xxx into 2105xxx.zip and upload the 2105xxx.zip on moodle.

For Queries

If you have any questions related to the assignment please first check the queries thread in Moodle. You should post your confusion in the thread. If your query goes unanswered for 24 hours, please mail me.

Mark Distribution

Task No.	Task Name	Mark
1	Proper implementation of BFS	25
2	Proper implementation of DFS	25
3	Detecting impossible case (tc03)	10
3	Properly works for knights (tc05)	15
3	Properly works for k-riders (tc04)	25