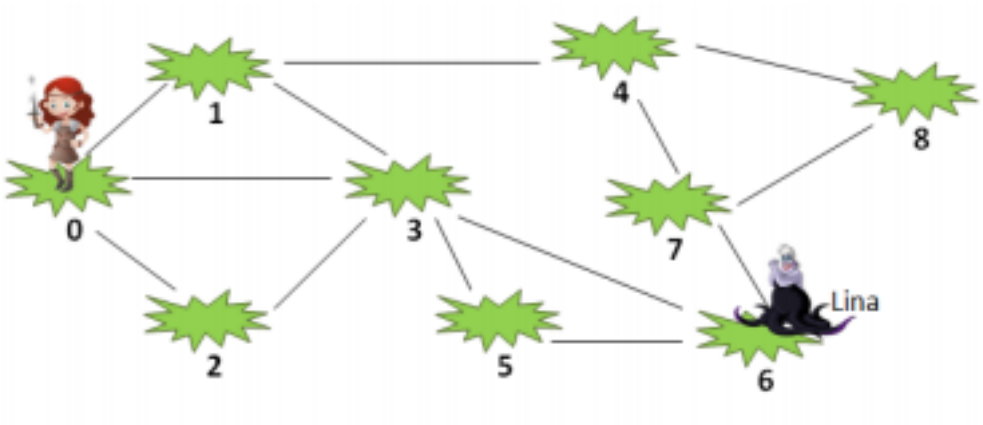
Graph Theory

Nora is a teenage girl living in Planet 'X'. A competition called "Fight with Brain" takes place on that planet yearly. There are several levels in that competition, one who completes all the levels wins the competition. This year the winner of "Fight with Brain" will get a ticket to ``Planet Earth". Nora has been excited about Planet "Earth" from her childhood, hence she has signed up for this competition. The organizers are now briefing Nora about the first Level.

Level 1: [1 Point]

Nora has been taken to "HighLand" for the first Level of the competition. Here there are n warriors with no weapons in different fixed positions. Nora has been given with a weapon and placed in position 0, other positions are 1, 2, 3,…, n-1. There are many connections among these positions. If there is a connection between position u and position v, Nora can go to position v from u (or vice versa) with 1 move. She can kill the warrior after reaching that position. Now the task of Nora is to kill warrior "Lina" who is in position 'x' Your task is to help Nora with the minimum number of
moves Nora needs to make to go to Lina.

Nora



Sample Input

9 //number of different fixed positions (including Nora's one)
13 //number of connections
0 1 //position 0 is connected with position 1
0 2
0 3
1 3
1 4
2 3
3 5

3 6
4 8
4 7
5 6
6 7
7 8
6 //'x'-Lina's position

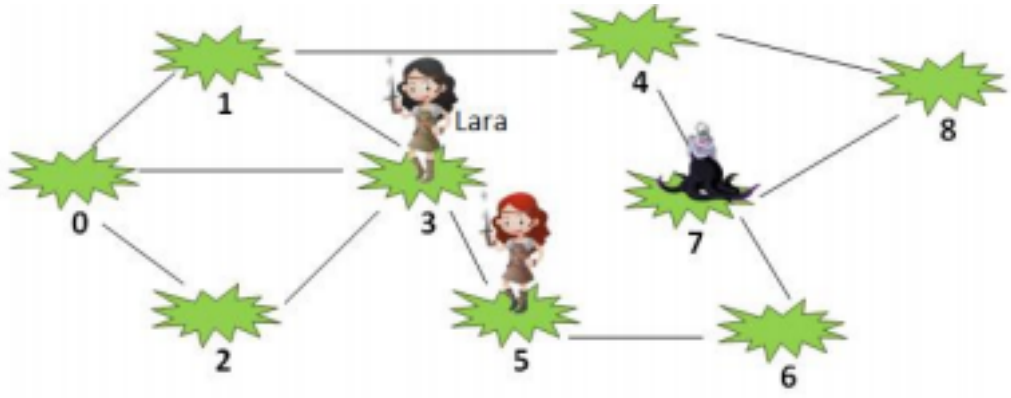
Sample Output

2 //Minimum number of moves Nora needs to go to 'x'

Nora has successfully completed Level 1 with your help and got 10 points as reward. She is proceeding to Level 2 now.

Level 2: [2 Points]

Nora has now been taken to "HigherLand" for Level 2 of the competition. Here she discovered that she will be competing with another participant "Lara". Here also there are n warriors in different fixed positions (position 0 through position n-1). Nora is now placed in position 'p' and Lara is placed in position 'q'. Both of them have to kill the warrior Lina in position 'x'. The one who goes to position 'x' first, wins Level 2.



For n different positions and m connections (u<->v) among the positions, your task is to determine who can go to 'x' first –Nora or Lara? Where Nora is in position 'p' and Lara is in position 'q'. If both win, print both.

Sample Input

9 //number of different fixed positions (including Nora's one)
12 //number of connections
0 1 //position 0 is connected with position 1
0 2
0 3
1 3
1 4
2 3

3 5
4 8
4 7
5 6
6 7
7 8
7 //'x'-Lina's position
5 // 'p'-Nora's position
3 // 'q'-Lara's position

Sample Output

Nora //winner, since Nora can go to 'x' with 2 moves and Lara can go with 3 moves

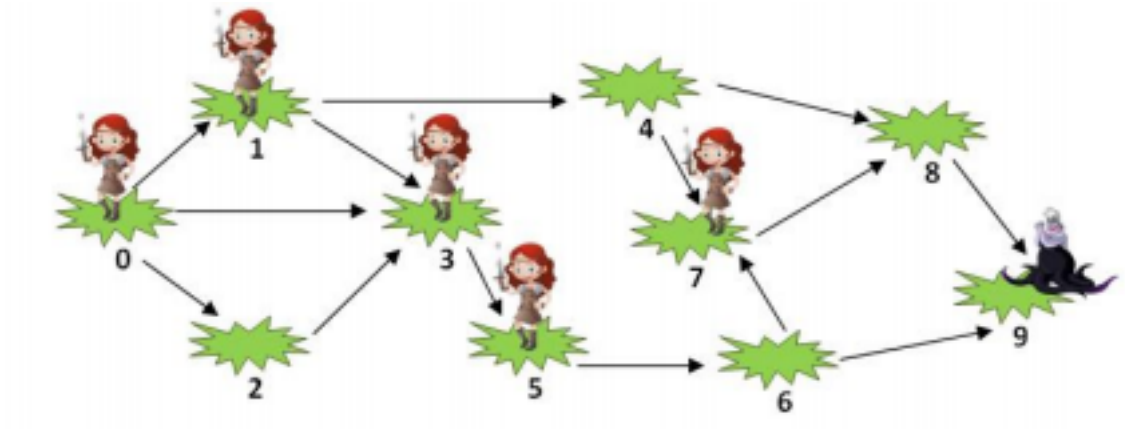
The one who completes Level 2 of "Fight with Brain" will be rewarded with 30 points and a "WildCard" which can be used to bypass the next Level (Level 3) and proceed to Level 4 without losing any point. But if decided to take Level 3 and completed successfully, participant will be rewarded with 30 bonus points.

Level 3: [2 Point]

Now, in this level, the participant is taken to "HighestLand" and is given a big challenge. He/she discovers that there are a total of k competitors now (including him/her). All of their target is to kill the warrior Lina in position 'x'. But now, if there's a connection from u to v, participants can go from u to v but not vice versa.

For n different fixed positions and m connections(u->v) among them, k different positions will be given along with position 'x' and your task will be to find the participant who reaches 'x' first. This time, you may just print the minimum number of moves the participant needed to reach 'x'.

You are thinking of applying the BFS algorithm, right? Go on! But the main challenge is yet to come, you may apply the algorithm only once. So the complexity of your algorithm should be no less than O(n+m).



Sample Input

10 //n - number of different fixed positions (including Nora's one)

14 //m - number of connections
0 1 //position 0 is connected with position 1
0 2
0 3
1 3
1 4
2 3
3 5
4 7
4 8
5 6
6 7
6 9
7 8
8 9
9 //'x'-Lina's position
5 // 'k'- number of participants
0 // position of k_1 participant
1 // position of k_2 participant
3 // position of k_3 participant
5 // position of k_4 participant
7 // position of k_5 participant

Sample Output

2 //minimum number of moves the winner(k_5) needed to go to 'x'