Who is to Buy the Ticket

A big event is going to happen. Bangladesh is going to win against Australia (at least you think so). And you don't afford to miss the moment. So you don;t afford to miss it. You and your friends are planning to watch the match in person. So you will go to the stadium. Now all of your friend will start at the same time from your house. Those who will arrive first will buy the ticket for all. Now you, being geekiest of all your friends, want to predict who are going to buy the tickets. Interestingly all the roads of Dhaka city have equal length. And all your friends and the stadium is in a junction.

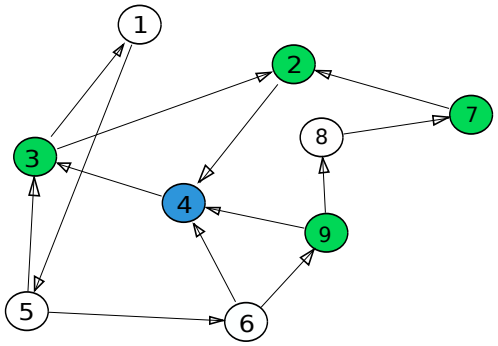

Figure 1: An example graph

The junction are numbered 1...n. In the example graph shown in figure 1, there are 6 junctions. Hence, $n = 6$. The stadium is at 4. Your friends live in $7, 3, 2, 9$. In this case, the friend who lives 2 and 9 will arrive first. So the answer will be $3, 4$ (3rd and 4th friend).

Input Specification

First line of input consists of two integers n and m. $1 \leq n \leq 300$ and $1 \leq m \leq n(n-1)$. First integer denotes number of junctions in the city, and the second integer denotes number of roads. The next m lines each contain two integers u, v which means there is a road from junction u to v in the graph. Note that the roads are one-way. Then contains a number $k \leq n$, number of people in your group. Next line contains k numbers, index of junction where your friend lives in. ID of first person in the list is 1, second person is 2 and so on. Last line of a case contains index of the stadium. The input file may contain description of multiple cases. Each of the case descriptions will be separated by a single blank line. You should stop taking inputs when 0 is input for both n and m.

Output Specification

For each of the cases in the input file, print one line which contains id of the friends who will arrive first in increasing order.

Sample Input

```
9 13
1 5
2 4
3 1
3 2
4 3
5 3
5 6
6 4
6 9
7 2
8 7
9 8
9 4
4
7 3 2 9
4

6 10
1 3
1 5
2 3
2 4
3 4
3 5
5 6
4 6
6 5
5 4
3
6 2 1
4

0 0
```

Sample Output

```
3 4
2
```

Remarks

Graph Representation You must use an adjacency list representation for the graph. Your graph should have an associated data structure, with which you will be able to find all the neighbors of a node u in $O(deg[u])$ time. Try to follow the adjacency represenstation style shown in the class. However, you are free to use `STL` data structures, if you like. Note that, unlike shown in class, the graph here is directed. So some modification in the process of building adjacency list will be needed.

Algorithm You have to use BFS(Breadth First Search) algorithm for this problem.

Input Output Explanation In the sample input given, there are two cases. The first case starts with the line `9 13` and ends with the line `4`. Output for that graph is the first line in the sample output. The second case starts with `6 10` in the sample input file, and the output for that graph is `2` as shown the second line in the sample output. However, the last line of input is `0 0`. Obviously, you should not process this as a graph, rather just exit from your program. So your code should take input of graphs in a loop and process them. The first sample is the example shown in figure 1.

```
while(scanf("%d%d",&n,&m)==2)
{
    if(n==0 && m==0) break;

    // some initialization here...

    for(i=0;i<m;i++)
    {
        scanf("%d%d",&u,&v);

        // insert the edge (u,v) in the adjacency list
        // ........
    }

    // .... process and print output
}
```

Tanaeem M Moosa
tanaeem.moosa@csebuet.org