valid path

Naveen is interested in graph theory and wants to determine if there exists a valid path from a given starting vertex to an ending vertex in a graph. He decided to use Depth-First Search (DFS) to solve this problem. Given a graph with vertices and edges, your task is to help Parthi write a program that determines if there is a valid path from a specified starting vertex to an ending vertex. A valid path is a sequence of vertices such that there is an edge between consecutive vertices in the sequence.

Input Format

The first line of input consists of the two integers n and m, representing the number of vertices and edges in the graph, respectively, separated by a space. The next m lines consist of two integers u and v, representing an undirected edge between vertices u and v. The last two lines of input consist of the two integers start and end, representing the starting and ending vertices, respectively.

Constraints

The test cases will fall under the following constraints: $1 \leq n \leq 10$, $0 \leq m \leq n*(n-1)/2$ $1 \leq u, v \leq n$, $u \neq v$ $1 \leq start, end \leq n$, $start \neq end$

Output Format

The output consists of the following format: If there is a valid path from the starting vertex to the ending vertex, print: "There is a path from [start] to [end]". If there is no valid path from the starting vertex to the ending vertex, print: "There is no path from [start] to [end]". Refer to the sample output for the formatting specifications

Sample Input 0

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

Sample Output 0

```
There is no path from 0 to 5
```

Contest ends in 16 days

Submissions: 52
Max Score: 10
Difficulty: Medium

Rate This Challenge:

More

Java 15

```
1 import java.io.*;
2 import java.util.*;
3
4 public class Solution {
5
```

```java
    static boolean dfs(int st,int ed,int arr[][],int v){
        Stack<Integer> s = new Stack();
        s.push(st);
        boolean vis[]=new boolean[v];
        while(!s.isEmpty()){
            int ele=s.pop();
            if(!vis[ele]){
                if(ele==ed){
                    return true;
                }
                vis[ele]=true;
            }
            for(int i=v-1;i>=0;i--){
                if(arr[ele][i]==1 && !vis[i]){
                    s.push(i);
                }
            }
        }
        return false;
    }

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. */
        Scanner sc=new Scanner(System.in);
        int v=sc.nextInt();
        int e=sc.nextInt();
        int arr[][]=new int[v][v];
        for(int i=0;i<e;i++){
            int st=sc.nextInt();
            int ed=sc.nextInt();
            arr[st][ed]=1;
            arr[ed][st]=1;
        }
        int st=sc.nextInt();
        int ed=sc.nextInt();
        if(!dfs(st,ed,arr,v)){
            System.out.printf("There is no path from %d to %d",st,ed);
```

```
43          }else{
44              System.out.printf("There is a path from %d to %d",st,ed);
45          }
46      }
47  }
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Compile Message

```
Note: Solution.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Compile Time

Input (stdin)

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

Run Time

Your Output (stdout)

```
There is no path from 0 to 5
```

Expected Output

```
There is no path from 0 to 5
```