```
/*
    Time complexity: O(V + E)
    Space complexity: O(V^2)

    where V is the number of vertices in the input graph and
    E is the number of edges in the input graph
*/

import java.util.LinkedList;
import java.util.Queue;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
public class Solution {
        static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        public static boolean BFS(int edges[][], int sv, int ev, boolean visited[]) {

                // Check for invalid input of sv or ev
        if (sv > (edges.length - 1) || ev > (edges.length - 1) ){
            return false;
        }

                if(edges[sv][ev] == 1) {
                        return true;
                }

                Queue<Integer> queue = new LinkedList<>();
                visited[sv] = true;
                queue.add(sv);

                while(!queue.isEmpty()) {
                        int front = queue.remove();

                        for(int i = 0; i < edges.length; i++) {
                                if(edges[front][i] == 1 && !visited[i]) {
                                        if(i == ev)
                                                return true;
                                        else {
                                                visited[i] = true;
                                                queue.add(i);
                                        }
                                }
                        }
                }
                return false;
        }

        private static boolean hasPath(int[][] edges, int sv, int ev) {
                boolean visited[] = new boolean[edges.length];
                return BFS(edges, sv, ev, visited);
        }

        public static void main(String[] args) throws NumberFormatException, IOException {

                String[] strNums;
        strNums = br.readLine().split("\\s");
        int n = Integer.parseInt(strNums[0]);
        int e = Integer.parseInt(strNums[1]);


                int edges[][] = new int[n][n];

        for (int i = 0; i < e; i++) {
            String[] strNums1;
            strNums1 = br.readLine().split("\\s");
            int fv = Integer.parseInt(strNums1[0]);
            int sv = Integer.parseInt(strNums1[1]);
            edges[fv][sv] = 1;
            edges[sv][fv] = 1;
        }

                String[] strNums1;
        strNums1 = br.readLine().split("\\s");
        int sv = Integer.parseInt(strNums1[0]);
        int ev = Integer.parseInt(strNums1[1]);

                System.out.println(hasPath(edges, sv, ev));
        }

}
```