```
/*
    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.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class Solution {
        static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        public static void DFS(int[][] edges, int sv, boolean[] visited) {
                visited[sv] = true;
                for(int i = 0; i < edges.length; i++) {
                        if(edges[sv][i] == 1 && !visited[i]) {
                                DFS(edges, i, visited);
                                visited[i] = true;
                        }
                }
        }

        public static boolean isConnected(int[][] edges) {
                boolean[] visited = new boolean[edges.length];

                DFS(edges, 0, visited);

                for(int i = 0; i < visited.length; i++) {
                        if(!visited[i]) {
                                return false;
                        }
                }

                return true;
        }

        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]);
                if (n==0){
            System.out.println("true");
            return;
        }
```

```
                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;
        }

                System.out.println(isConnected(edges));
        }

}
```