```
/*
    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.HashMap;
import java.util.LinkedList;
import java.util.Map;
import java.util.Queue;
import java.util.ArrayList;
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 ArrayList<Integer> getPathBFSHelper(int[][] edges, int sv, int ev, boolean[] visited) {
                int n = edges.length;

                Map<Integer, Integer> map = new HashMap<>();
                Queue<Integer> queue = new LinkedList<>();

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

                if(edges[sv][ev] == 1 && sv == ev) {
                        ArrayList<Integer> ans = new ArrayList<>();
                        ans.add(sv);
                        return ans;
                }

                queue.add(sv);
                visited[sv] = true;

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

                        for(int i = 0; i < n; i++) {
                                if(edges[front][i] == 1 && !visited[i]) {
                                        map.put(i, front);
                                        queue.add(i);

                                        visited[i] = true;

                                        if(i == ev) {
```

```
                                        ArrayList<Integer> ans = new ArrayList<>();
                                        ans.add(ev);
                                        int value = map.get(ev);

                                        while(value != sv) {
                                                ans.add(value);
                                                value = map.get(value);
                                        }
                                        ans.add(value);

                                        return ans;
                                }
                        }
                }
        }
        return null;
}

public static ArrayList<Integer> getPathBFS(int[][] edges, int sv, int ev) {
        boolean[] visited = new boolean[edges.length];
        return getPathBFSHelper(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]);

        ArrayList<Integer> ans = getPathBFS(edges, sv, ev);
        if(ans != null) {
                for(int elem: ans) {
```

```
                System.out.print(elem + " ");
            }
        }
    }
}
```