https://course.acciojob.com/idle?question=bc51d934-d08b-4b7d-a852-2dec91e07bce

- EASY
- Max Score: 30 Points

DFS Implementation

Given a connected undirected graph consisting of `n` vertices and `e` edges. All the edges are given in form of a 2-D matrix `edges` of size `e * 2`, where there is an undirected edge between `edges[i][0]` and `edges[i][1]` for all `i` ∈ {`0`, `e-1`}.

Perform a Depth First Traversal of the graph.

Note: When traversing the vertices of a graph, if vertex u is connected to a set of neighboring vertices {n1, n2, n3, ...}, then the traversal should proceed in the order of n1, n2, n3, ...; where n1 < n2 < n3 < ...

For example if vertex u has neighbours {3, 1, 4} then first traverse neighbouring node 1, then 3, and lastly 4 as 1 < 3 < 4.

Input Format

The First line contains two integers `n` and `e` which denotes the no of vertices and no of edges respectively.

Next `e` lines contains two integers `u` and `v` where there is an edge between `u` and `v`

Output Format

Complete the function DFSTraversal() which prints the vertices. Start the traversal for vertex with the smallest index.

Example 1

Input

```
3 4
1 2
```

```
0 2
0 1
0 1
```

Output

```
0 1 2
```

Explanation

```
The Graph looks like:
   0
 /   \
1 --- 2
```

So starting traversal from node 0, we first print 0. Then we go to the neighbours of 0 which are {1, 2}. Since 1 < 2, we first traverse node 1, then node 2 printing them both respectively

Example 2

Input

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

Output

```
0 2 1 3
```

Constraints

0 <= n <= 10^3

1 <= e <= 10^4

Topic Tags

- Graphs

My code

```
// n java
import java.util.*;
// class Solution {
//      public static void dfs(ArrayList<ArrayList<Integer>> g, int x,
boolean[] vis) {
//              if(vis[x] == true) return;

//              vis[x] = true;
//              System.out.print(x + " ");

//              for(int i = 0; i < g.get(x).size(); i++) {
//                      dfs(g, g.get(x).get(i), vis);
//              }
//      }

//     public static void DFSTraversal(List<List<Integer>> edges, int
n) {
//         //Write your code here
//              ArrayList<ArrayList<Integer>> g = new ArrayList<>();

//              for(int i = 0; i < n; i++) {
//                      g.add(new ArrayList<Integer>());
//              }

//              for(int i = 0; i < edges.size(); i++) {
```

```
//                  int x = edges.get(i).get(0), y = edges.get(i).get(1);

//                  g.get(x).add(y);
//                  g.get(y).add(x);
//          }

//          for(int i = 0; i < n; i++) {
//                  Collections.sort(g.get(i));
//          }

//          boolean[] vis = new boolean[n];
//          for(int i = 0; i < n; i++) vis[i] = false;

//          dfs(g, 0, vis);
//      }
// }
```

```
class Solution {
public static void dfs(ArrayList<ArrayList<Integer>>g, int x,boolean
vis[])
    {
        if(vis[x]==true) return;
        vis[x]=true;
        System.out.print(x+" ");
        for(int i=0;i<g.get(x).size();i++)
            dfs(g,g.get(x).get(i),vis);
    }
```

```
    public static void DFSTraversal(List<List<Integer>> edges, int n)
{
        //Write your code here
        //      ArrayList<ArrayList<Integer>>g=new
ArrayList<Integer>();
                ArrayList<ArrayList<Integer>> g = new ArrayList<>();
                for(int i=0;i<n;i++)
                        g.add(new ArrayList<Integer>());
                for(int i=0;i<edges.size();i++)
                        {
                                int x=edges.get(i).get(0);
                                int y=edges.get(i).get(1);
                                g.get(x).add(y);
                                g.get(y).add(x);
                        }
                //sort for dfs
                for(int i=0;i<n;i++)
                Collections.sort(g.get(i));

                boolean vis[]=new boolean[n];
                        for(int i=0;i<n;i++)
                                vis[i]=false;//no meter default is also false

                  dfs(g,0,vis);//graph ,starting point ,array flags
    }
}

public class Main {
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int e = sc.nextInt();
        List<List<Integer>> ed = new ArrayList<>();
        for (int i = 0; i < e; i++) {
            List<Integer> l = new ArrayList<>();
            l.add(sc.nextInt());
            l.add(sc.nextInt());
            ed.add(l);
        }

        Solution ob = new Solution();
        ob.DFSTraversal(ed, n);
    }
}
```