Course Name: DAA Lab **Course Code: 21ITH-311/21CSH-311**

Experiment 3.1

Aim: Develop a program and analyze complexity to do a depth-first search (DFS) on an undirected graph. Implementing an application of DFS such as to find the topological sort of a directed acyclic graph

Objectives: Code and analyze to do a depth-first search (DFS) on an undirected graph. Implementing an application of DFS such as (i) to find the topological sort of a directed acyclic graph

Input/Apparatus Used: Graph (G = (V,E)) is taken as input for this problem.

Procedure/Algorithm:

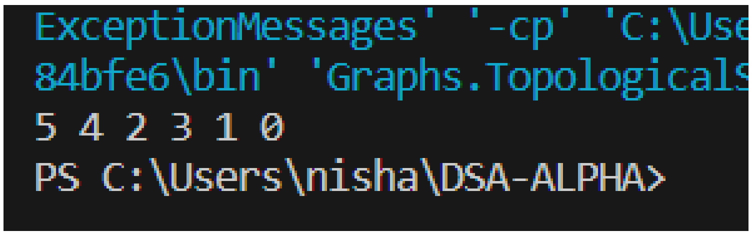
- Create a recursive function that takes the index of the node and a visited array.
- Mark the current node as visited and print the node.
- Traverse all the adjacent and unmarked nodes and call the recursive function with the index
- of the adjacent node.

Sample Code:

```
package Graphs;
import java.util.*;
public class TopologicalSorting {
    static class Edge{
        int src, dest;

        public Edge(int s, int d){
            this.src = s;
            this.dest = d;
        }
    }
    public static void createGraph(ArrayList<Edge>[] graph){
        for(int i=0; i<graph.length; i++){ // false
            graph[i] = new ArrayList<>();
        }
        graph[2].add(new Edge(2, 3));
        graph[3].add(new Edge(3, 1));
        graph[4].add(new Edge(4, 0));
        graph[4].add(new Edge(4, 1));
        graph[5].add(new Edge(5, 0));
        graph[5].add(new Edge(5, 2));
    }
    public static void topSort(ArrayList<Edge>[] graph){ // O(V+E)
        boolean vis[] = new boolean[graph.length];
        Stack<Integer> s = new Stack<>();
        for(int i=0; i<graph.length; i++){
            if(!vis[i]){
                topSortUtil(graph, i, vis, s);
            }
        }
        while(!s.isEmpty()){
            System.out.print(s.pop() + " ");
        }
    }
    public static void topSortUtil(ArrayList<Edge>[] graph, int curr, boolean vis[], Stack<Integer> s){
        vis[curr] = true;
        for(int i=0; i<graph[curr].size(); i++){
            Edge e = graph[curr].get(i);
            if(!vis[e.dest]){
                topSortUtil(graph, e.dest, vis, s);
            }
        }
        s.push(curr);
    }
    public static void main(String[] args) {
        int V = 6;
        ArrayList<Edge> graph[] = new ArrayList[V];
        createGraph(graph);
        topSort(graph);
    }
}
```

Observations/Outcome :



```
ExceptionMessages' '-cp' 'C:\Use
84bfe6\bin' 'Graphs.TopologicalS
5 4 2 3 1 0
PS C:\Users\nisha\DSA-ALPHA>
```

Time Complexity:

O(V + E)