**CSE-2212: Design and Analysis of Algorithms-I Lab**

**Practice Lab 2– April 8, 2025**

**Experiment:** Create a Graph Class with Depth-First Search (DFS) and Topological Sort in Java.

1. Create a Graph class where the constructor takes a filename as input and graph is initialized using input from the file and stored in an adjacency list or matrix. Here’s an example input for 5 vertices, 3 edges:

5 3

1 2

2 5

3 4

1. There should be functions to
2. add vertex(n): Add n new vertices to the graph
3. add edge(u,v): Add an edge to the graph
4. get the number of vertices
5. get adjacent vertices of a given vertex
6. display the graph’s adjacency list

Then you should implement the following features:

**1. Depth-First Search (DFS):** Perform DFS traversal on the graph and print the order of vertices visited.

**2. Topological Sort:** Implement a topological sorting algorithm using DFS, and return the vertices in topologically sorted order. [ Check Lecture 4 in [classroom](https://docs.google.com/presentation/d/1rArG7yiIm0ssqp57KsrWsdGbw9rh8Njq/edit?usp=sharing&ouid=114013689365888328149&rtpof=true&sd=true)]

The graph should be initialized using input from a file. For example, given the following file: **[ Notice that the vertices start from 0 today ]**

6 6

5 2

5 0

4 0

4 1

2 3

3 1

This input represents a directed graph with 6 vertices and 6 edges.

**New Methods to Implement:**

- **DFS(v):** Perform DFS starting from vertex `v` and print the order of traversal.

- **topologicalSort():** Perform Topological Sorting and return the list of vertices in topologically sorted order.

**DFS(v):**

- Implement DFS traversal starting from a given vertex.

- Print the order of vertices visited during the DFS traversal.

**topologicalSort():**

- Implement Topological Sort using DFS for Directed Acyclic Graphs (DAG) only.

- The method should return a list of vertices in topologically sorted order.

**This should be your main class:**

public class Lab2 {

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

Graph graph = new Graph("input.txt");

System.out.println("Graph adjacency list:");

graph.displayGraph();

System.out.println("\nPerforming DFS starting from vertex 5:");

graph.DFS(5);

System.out.println("\nPerforming Topological Sort:");

List<Integer> topoOrder = graph.topologicalSort();

System.out.println("Topological Sort order: " + topoOrder);

}

}

**Sample input.txt:** This represents a directed acyclic graph with 6 vertices and 6 edges:

6 6

5 2

5 0

4 0

4 1

2 3

3 1

**Expected Output:**

Graph adjacency list:

0 ->

1 ->

2 -> 3

3 -> 1

4 -> 0 1

5 -> 2 0

Performing DFS starting from vertex 5:

DFS Traversal starting from vertex 5: 5 2 3 1 0

Performing Topological Sort:

Topological Sort order: [5, 4, 2, 3, 1, 0]

**Bonus Challenges:**

1. Find the path between a source vertex and a destination vertex
2. Run DFS in lexicographic order and print the order of vertices visited

Practice Problems:

1. <https://www.codechef.com/practice-old/tags/dfs>
2. <https://www.hackerearth.com/practice/algorithms/graphs/depth-first-search/practice-problems/>
3. https://leetcode.com/problem-list/depth-first-search/