National University of Computer and Emerging Sciences



# Lab Manual

for

# Data Structure

Department of Computer Science FAST-NU, Lahore, Pakistan

**Objectives:**

After performing this lab, students shall be able to revise:

* DFS

**Question 1) Recursive Implementation of DFS Traversal and Cycle Detection**

#### ****Objective:****

### Implement Depth First Search (DFS) recursively for graph traversal and spanning tree construction. Additionally, implement an **cycle detection function** to determine if the graph contains cycles.

### ****Problem Statement:****

You are provided with a graph represented as an adjacency list. Your task is to:

1. Implement a **recursive DFS traversal** to visit all vertices reachable from a given vertex v.
2. Implement a **cycle detection** function using DFS.
3. Implement a function to **construct a spanning tree** using DFS traversal and output the edges of the spanning tree.

### ****Tasks:****

#### ****Part 1: Recursive DFS Traversal****

* Implement the Graph::DFS() method to perform DFS traversal for all unvisited vertices.
* Use the provided recursive DFSHelp(int v, bool visited[]) function to explore adjacent vertices.

**Implementation Details:**

* Write a recursive function to mark the current vertex as visited and recursively visit all adjacent vertices.

**Input:**

* Number of vertices (V).
* List of edges defining the graph.
* Starting vertex v for traversal.

**Output:**

* Print the vertices in the order they are visited during DFS.

**Part 2: Cycle Detection**

* Implement the Graph::isCyclic() method to detect if the graph contains any cycles using a helper function isCyclicUtil(int v, bool visited[], int parent).

**Input:**

* Number of vertices (V).
* List of edges defining the graph.

**Output:**

* Print true if the graph contains a cycle; otherwise, print false.

### ****Steps to Follow:****

1. **Graph Representation:**  
   Represent the graph using an adjacency list. Create a Graph class that includes methods DFS(), DFSHelp(), isCyclic(), and isCyclicUtil().
2. **Recursive DFS Implementation:**  
   a. Implement the recursive function DFSHelp() to traverse vertices.  
   b. Use the DFS() method to start DFS from all unvisited vertices in the graph.
3. **Cycle Detection:**  
   a. Implement the isCyclicUtil() function to detect cycles.  
   b. Use the isCyclic() method to call the helper function for each connected component.

#### ****Part 3: Spanning Tree Construction****

* Implement a Graph::spanningTree() method to construct a spanning tree using DFS traversal.
* During traversal, track and store edges used to visit unvisited vertices.

**Implementation Details:**

* Use a vector or list to store the spanning tree edges.
* Ensure the spanning tree has exactly V-1 edges for a connected graph.

**Input:**

* Number of vertices (V).
* List of edges defining the graph.

**Output:**

* Print the edges of the spanning tree.