6. Example: Full Code Implementation

This chapter combines **all the concepts** into a **single Graph class** using the C++ STL for the adjacency list.

```
#include <iostream>
#include <vector>
#include <list>     // For Adjacency List
#include <queue>    // For BFS
#include <stack>    // For Iterative DFS
#include <vector>   // For visited tracker

class Graph {
private:
    int V; // Number of vertices
    // Adjacency List: A vector of lists
    std::vector<std::list<int>> adj;

    // Helper for recursive DFS
    void DFSUtil(int u, std::vector<bool>& visited) {
        visited[u] = true;
        std::cout << u << " ";
        for (auto& neighbor : adj[u]) {
            if (!visited[neighbor]) {
                DFSUtil(neighbor, visited);
            }
        }
    }

public:
    // Constructor
    Graph(int V) {
        this->V = V;
        adj.resize(V); // Resize the vector to hold V lists
    }
```

```
// Add an edge (for an undirected graph)
void addEdge(int u, int v) {
    adj[u].push_back(v);
    adj[v].push_back(u); // Comment this line for a directed graph
}

// Breadth-First Search (Iterative)
void BFS(int s) {
    std::vector<bool> visited(V, false);
    std::queue<int> q;
    visited[s] = true;
    q.push(s);

    std::cout << "BFS Traversal: ";
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        std::cout << u << " ";

        for (auto& neighbor : adj[u]) {
            if (!visited[neighbor]) {
                visited[neighbor] = true;
                q.push(neighbor);
            }
        }
    }
    std::cout << std::endl;
}

// Depth-First Search (Iterative)
void DFS(int s) {
    std::vector<bool> visited(V, false);
    std::stack<int> stack;
    stack.push(s);

    std::cout << "DFS Traversal (Iterative): ";
    while (!stack.empty()) {
        int u = stack.top();
        stack.pop();

        if (!visited[u]) {
```

```
                visited[u] = true;
                std::cout << u << " ";
            }

            for (auto& neighbor : adj[u]) {
                if (!visited[neighbor]) {
                    stack.push(neighbor);
                }
            }
        }
        std::cout << std::endl;
    }

    // Depth-First Search (Recursive)
    void DFSRecursive(int s) {
        std::vector<bool> visited(V, false);
        std::cout << "DFS Traversal (Recursive): ";
        DFSUtil(s, visited);
        std::cout << std::endl;
    }
};
```

Revision #1
Created 2025-11-11 11:29:54 UTC by RE
Updated 2025-11-11 11:32:18 UTC by RE