# Assignment 6

Mayur Zope SE Comp A 75

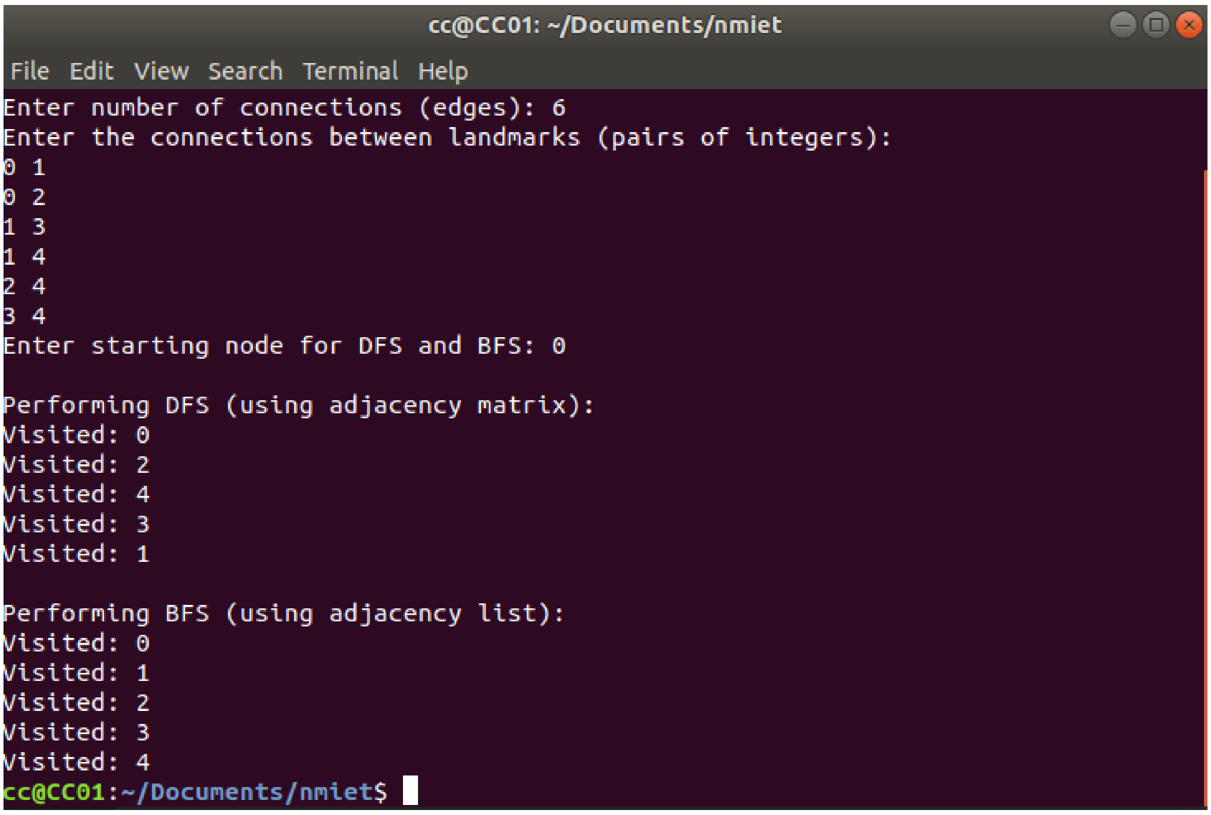
Represent a given graph using adjacency matrix/list to perform DFS and using adjacency list to perform BFS. Use the map of the area around the college as the graph. Identify the prominent land marks as nodes and perform DFS and BFS on that.

```
#include <iostream>
#include <vector>
#include <queue>
#include <stack>

using namespace std;

class Graph {
public:
    vector<vector<int>> adjMatrix;
    vector<vector<int>> adjList;
    int V;

    Graph(int V) {
        this->V = V;
        adjMatrix.resize(V, vector<int>(V, 0));
        adjList.resize(V);
    }

    void addEdgeMatrix(int u, int v) {
        adjMatrix[u][v] = 1;
        adjMatrix[v][u] = 1;
    }

    void addEdgeList(int u, int v) {
        adjList[u].push_back(v);
        adjList[v].push_back(u);
```

```
}

void DFSMatrix(int start, vector<bool>& visited) {
    stack<int> s;
    s.push(start);
    visited[start] = true;

    while (!s.empty()) {
        int node = s.top();
        s.pop();
        cout << "Visited: " << node << endl;

        for (int i = 0; i < V; i++) {
            if (adjMatrix[node][i] == 1 && !visited[i]) {
                visited[i] = true;
                s.push(i);
            }
        }
    }
}

void BFSList(int start) {
    vector<bool> visited(V, false);
    queue<int> q;
    q.push(start);
    visited[start] = true;

    while (!q.empty()) {
        int node = q.front();
        q.pop();
```

```
            cout << "Visited: " << node << endl;

            for (int neighbor : adjList[node]) {
                if (!visited[neighbor]) {
                    visited[neighbor] = true;
                    q.push(neighbor);
                }
            }
        }
    }
};

int main() {
    int V, E;
    cout << "Enter number of landmarks (nodes): ";
    cin >> V;

    Graph g(V);

    cout << "Enter number of connections (edges): ";
    cin >> E;

    cout << "Enter the connections between landmarks (pairs of integers):" << endl;
    for (int i = 0; i < E; i++) {
        int u, v;
        cin >> u >> v;
        g.addEdgeMatrix(u, v);
        g.addEdgeList(u, v);
    }
    int start;
    cout << "Enter starting node for DFS and BFS: ";
    cin >> start;

    cout << "\nPerforming DFS (using adjacency matrix):" << endl;
    vector<bool> visitedDFS(V, false);
    g.DFSMatrix(start, visitedDFS);

    cout << "\nPerforming BFS (using adjacency list):" << endl;
    g.BFSList(start);

    return 0;
}
```

//OUTPUT



cc@CC01: ~/Documents/nmiet
File Edit View Search Terminal Help

```
Enter number of connections (edges): 6
Enter the connections between landmarks (pairs of integers):
0 1
0 2
1 3
1 4
2 4
3 4
Enter starting node for DFS and BFS: 0

Performing DFS (using adjacency matrix):
Visited: 0
Visited: 2
Visited: 4
Visited: 3
Visited: 1

Performing BFS (using adjacency list):
Visited: 0
Visited: 1
Visited: 2
Visited: 3
Visited: 4
cc@CC01:~/Documents/nmiet$
```