Lab Report-04

(Depth_First_Search)

CSE-2212 (Design and Analysis of Algorithms Lab)

Submitted By:

Name: Eyasir Ahamed
Exam Roll: 413
Class Roll: 15
Registration No: 202004017

Submitted To:

Sharad Hasan
Ex. Lecturer
Dept. of CSE
Sheikh Hasina University, Netrokona

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SHEIKH HASINA UNIVERSITY
NETROKONA, BANGLADESH

#4_Depth First Search (DFS)

Problem Definition

Given a graph represented as an adjacency list and a starting vertex start, the problem is to traverse the graph using Depth First Search algorithm starting from the vertex start.

Formal Statement of the Algorithm

- Initialize a visited array to keep track of visited vertices and mark start as visited.
- Visit vertex start.
- For each adjacent vertex u of start:
    - If u is not visited, recursively call DFS on u.
- Repeat step 3 for each unvisited adjacent vertex of the current vertex.

Complexity Analysis

- Time Complexity:
    - O(V + E), where V is the number of vertices and E is the number of edges.
    - Similar to BFS, DFS also visits each vertex and edge once, so the time complexity is linear in terms of both vertices and edges.
- Space Complexity:
    - O(V) for the visited array.
    - O(V) for the recursion stack in the worst case when all vertices are part of the DFS traversal path.

- Total: O(V + V) = O(V), where V is the number of vertices.
- Additional space is required for maintaining the visited array and the recursion stack.

Actual Code and Output

```cpp
#include <iostream>
#include <vector>

using namespace std;

void dfs(vector<vector<int>>& graph, vector<bool>& visited, int v) {
    visited[v] = true;
    cout << v << " ";

    for (int u : graph[v]) {
        if (!visited[u]) {
            dfs(graph, visited, u);
        }
    }
}

void dfsTraversal(vector<vector<int>>& graph, int start) {
    int n = graph.size();
    vector<bool> visited(n, false);
    dfs(graph, visited, start);
}

int main() {
    vector<vector<int>> graph = {
        {1, 2},
        {0, 3, 4},
        {0, 4},
        {1},
        {1, 2}
    };

    cout << "DFS traversal starting from vertex 0: ";
    dfsTraversal(graph, 0);
    cout << endl;

    return 0;
}
```

```
DFS traversal starting from vertex 0: 0 1 3 4 2
[Finished in 373ms]
```