

Experiment No. 1.3

Student Name: Jesu Yuvraj

UID: 22MCI10074

Branch: MCI – AIML

Section/Group: 22MCI -1 / Grp B

Semester: I

Date of Performance: 12th Nov 22

Subject Name: DAA Lab

Subject Code: 22CAP-636

1. Aim/Overview of the practical:

Using DFS traversal check the whether the directed graph is connected. Print all the nodes.

2. Task to be done:

To write a program which uses DFS to search if all the vertex are connected or not. If not, then graph is not connected, then answer will be “NO.” And to print all the nodes/vertex.

3. Algorithm/Flowchart:

- **Step 1)** Take two bool arrays vis1 and vis2 of size N (number of nodes of a graph) and keep false in all indexes.
- **Step 2)** Start at a random vertex v of the graph G, and run a DFS(G, v).
- **Step 3)** Make all visited vertices v as vis1[v] = true.
- **Step 4)** Now reverse the direction of all the edges.
- **Step 5)** Start DFS at the vertex which was chosen at step 2.
- **Step 6)** Make all visited vertices v as vis2[v] = true.
- **Step 7)** If any vertex v has vis1[v] = false and vis2[v] = false then the graph is not connected.

4. Code for experiment/practical:

```
#include <bits/stdc++.h>
using namespace std;
#define N 100000

// To keep correct and reverse direction
vector<int> gr1[N], gr2[N];

bool vis1[N], vis2[N];

// Function to add edges
void Add_edge(int u, int v)
{
    gr1[u].push_back(v);
    gr2[v].push_back(u);
    cout<<"Two Nodes of this edge is: "<<u<<" and "<<v<<endl;
}

// DFS function
void dfs1(int x)
{
    vis1[x] = true;

    for (auto i : gr1[x]){
        if (!vis1[i]){
            dfs1(i);
        }
    }
}

// DFS function
void dfs2(int x)
{
    vis2[x] = true;

    for (auto i : gr2[x]){
        if (!vis2[i]){
            dfs2(i);
        }
    }
}

bool Is_Connected(int n)
{
    // Call for correct direction
    memset(vis1, false, sizeof vis1);
    dfs1(1);

    // Call for reverse direction
    memset(vis2, false, sizeof vis2);
    dfs2(1);

    for (int i = 1; i <= n; i++) {

        // If any vertex it not visited in any direction
        // Then graph is not connected
        if (!vis1[i] and !vis2[i])
```

```
        return false;
    }

    // If graph is connected
    return true;
}

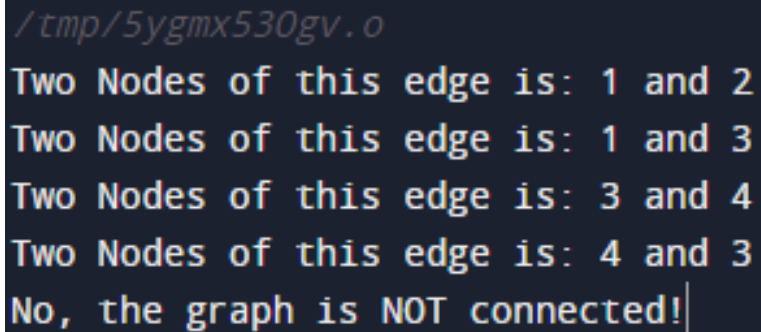
// Driver code
int main()
{
    int n = 5;

    // Add edges
    Add_edge(1, 2);
    Add_edge(1, 3);
    Add_edge(3, 4);
    Add_edge(4, 3);

    // Function call
    if (Is_Connected(n))
        cout << "Yes, it is a connected graph!";
    else
        cout << "No, the graph is NOT connected!";

    return 0;
}
```

5. Result/Output/Writing Summary:



```
/tmp/5ygm530gv.o
Two Nodes of this edge is: 1 and 2
Two Nodes of this edge is: 1 and 3
Two Nodes of this edge is: 3 and 4
Two Nodes of this edge is: 4 and 3
No, the graph is NOT connected!
```

Learning outcomes (What I have learned):

1. Learned to implement the DFS in a graph.
2. Learned to check whether a graph is connected or not.