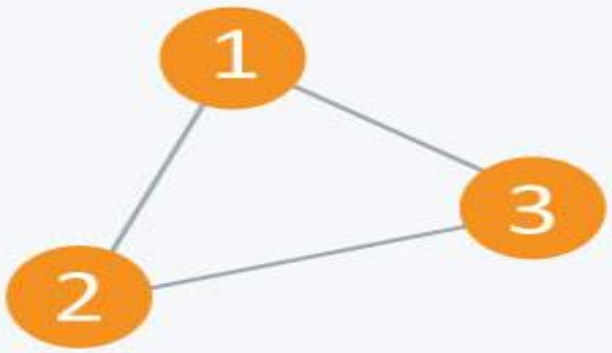
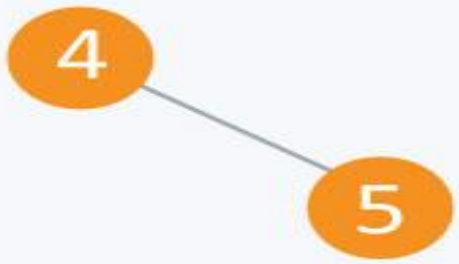
DFS Implementation in C++

Presented by: Adarsh Singh

Graph G



First Connected Component



Second Connected Component

Third Connected Component

File Edit Selection Find View Goto Tools Project Preferences Help

adja.cpp

```cpp
#include <iostream>
#include <vector>
using namespace std;

vector <int> adj[10];
bool visited[10];

void dfs(int s) {
    visited[s] = true;
    for(int i = 0;i < adj[s].size();++i)    {
     if(visited[adj[s][i]] == false)
         dfs(adj[s][i]);
    }
}

void initialize() {
    for(int i = 0;i < 10;++i)
     visited[i] = false;
}

int main() {
    int nodes, edges, x, y, connectedComponents = 0;
    cin >> nodes;                          // Number of nodes
    cin >> edges;                          // Number of edges
    for(int i = 0;i < edges;++i) {
     cin >> x >> y;
 // Undirected Graph
     adj[x].push_back(y);                  // Edge from vertex x to vertex y
     adj[y].push_back(x);                  // Edge from vertex y to vertex x
    }
```

adja.cpp

```
    for(int i = 0;i < 10;++i)
     visited[i] = false;
}

int main() {
    int nodes, edges, x, y, connectedComponents = 0;
    cin >> nodes;                          // Number of nodes
    cin >> edges;                          // Number of edges
    for(int i = 0;i < edges;++i) {
     cin >> x >> y;
 // Undirected Graph
     adj[x].push_back(y);                  // Edge from vertex x to vertex y
     adj[y].push_back(x);                  // Edge from vertex y to vertex x
    }

    initialize();                          // Initialize all nodes as not visited

    for(int i = 1;i <= nodes;++i) {
     if(visited[i] == false)     {
         dfs(i);
         connectedComponents++;
     }
    }
    cout << "Number of connected components: " << connectedComponents << endl;
    return 0;
}
```

Input/Output

Input

6
4
1 2
2 3
1 3
4 5

Output

3

Time Complexity

- Time complexity of DFS is **O(V + E)** , where V is the number of nodes and E is the number of Edges.

Thanks