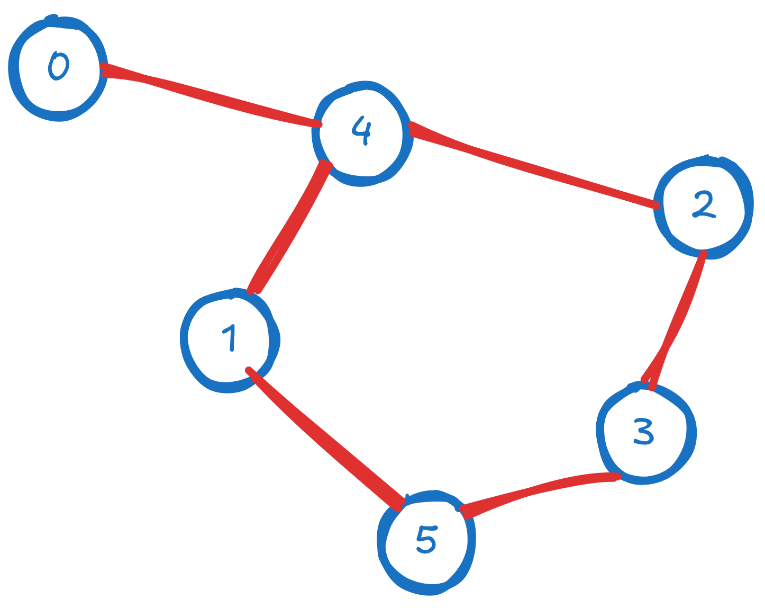
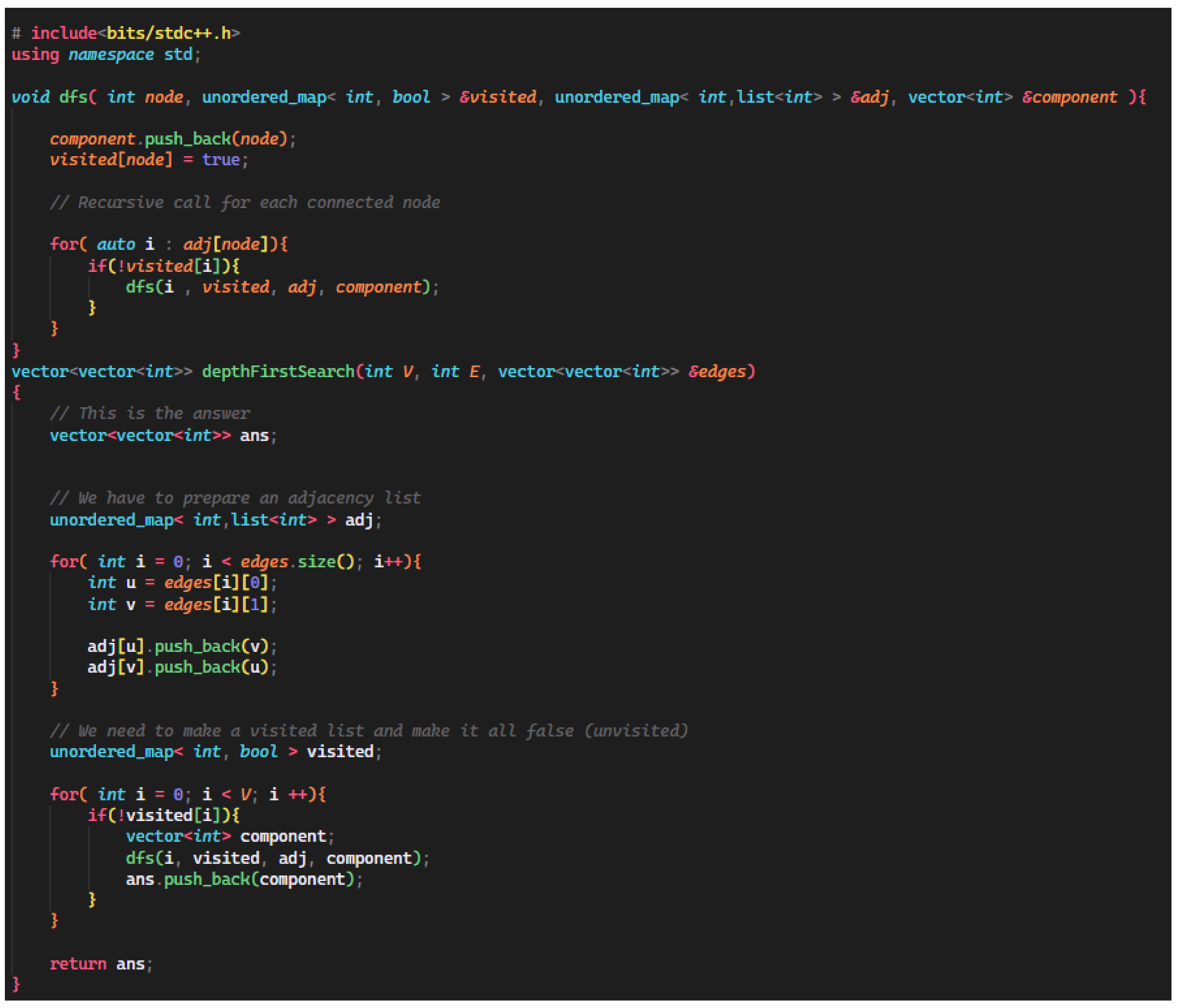
Depth First Search



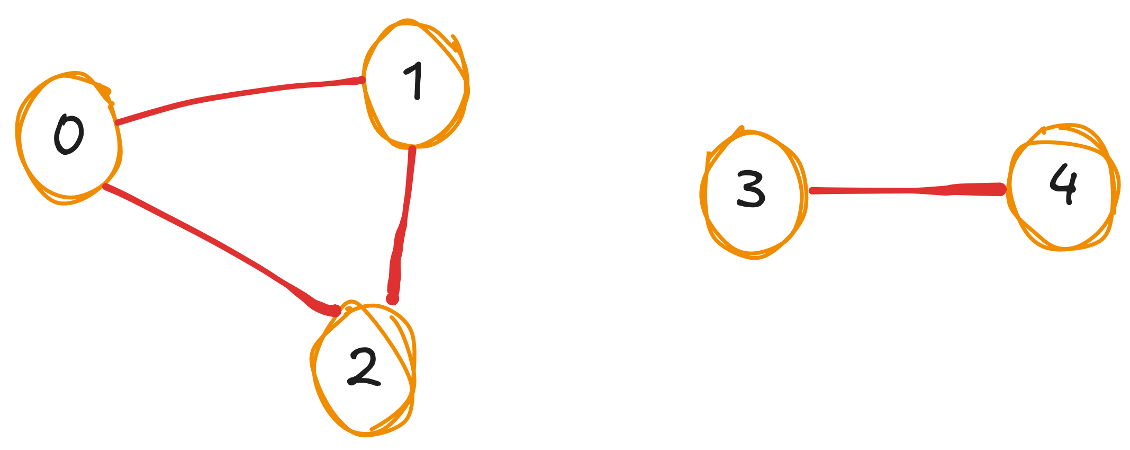
BFS : 0, 4, 2, 1, 3, 5

DFS : 0, 4, 2, 3, 5, 1 or 0, 4, 1, 5, 3, 2



```cpp
# include<bits/stdc++.h>
using namespace std;

void dfs( int node, unordered_map< int, bool > &visited, unordered_map< int,list<int> > &adj, vector<int> &component ){

    component.push_back(node);
    visited[node] = true;

    // Recursive call for each connected node

    for( auto i : adj[node]){
        if(!visited[i]){
            dfs(i , visited, adj, component);
        }
    }
}
vector<vector<int>> depthFirstSearch(int V, int E, vector<vector<int>> &edges)
{
    // This is the answer
    vector<vector<int>> ans;


    // We have to prepare an adjacency list
    unordered_map< int,list<int> > adj;

    for( int i = 0; i < edges.size(); i++){
        int u = edges[i][0];
        int v = edges[i][1];

        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    // We need to make a visited list and make it all false (unvisited)
    unordered_map< int, bool > visited;

    for( int i = 0; i < V; i ++){
        if(!visited[i]){
            vector<int> component;
            dfs(i, visited, adj, component);
            ans.push_back(component);
        }
    }

    return ans;
}
```

Explanation of the code and the logic for Depth-First-Search



Given are the two components of the graph, 0 1 2 are connected and 3 4 component is disconnected from 0 1 2

There is a catch in this visited list making in here, we just initializer this visited array as unoreder_map< int, bool > though we do not explicitly intialise all of the edges to false,

Initially all of the visited map is empty with no keys , when we ask for visited[1], we get false because there is no key of Node 1 and when we do visited[1] = true, a key of 1 is made mapped with true. So we actually no need to make map with all node initialised to false.

Adjacency list

0 -> 1 ,2
1 -> 0, 2
2 -> 1,0
3 -> 4
4 -> 3

Visited List

0 -> True
1 -> True
2 -> True
3 -> True
4 -> True

- We will be doing recursive call in the function.
- Start with source node, which is 0, this is not visited, we intialized a vector called as component and called the dfs function.
- We push this node into component, so component vector becomes 0
- We mark this 0 as visited.
- for adj[0] means, the adjacency of 0 which is 1 , 2
- 1 is not visited so we called df(1).
- df(1) , 1 was added to components 0 1
- 1 was marked as visited
- Look at the adjacency of 1 , is 0 and 2, 0 is marked as true, so now df(2) is called
- df(2) , adds 2 to the component vector 0 1 2
- 2 will be marked as true
- 2 has the adjacency of 0 and 1 but both 0 and 1 are marked as true, so the function ends
- we have the component vector , which is added to the ans as 0 1 2

- Now for loop in main runs for i = 1, which is marked as visited, nothing happens.
- Now for loop in main runs for i = 2, which is marked as visited nothing happens.

- Now for loop goes for i = 3 and it is not visited so, component vector is initialised and df(3) is called 3 is added to the component and 3 is marked as true 3
- Now we look at the adjacency of 3 which is 4, 4 is not visited so df(4) is called
- df(4) adds 4 to the component and 4 is marked true 3 4
- 4 has the adjacency has 3 which is marked true so nothing happens, we go the main loop and add this component to the answer

final answer : 1 0 2 3 4