DSA SERIES

- Learn Coding

Topic to be Covered today

Graph

547. Number of Provinces

```
class Solution {
public:


void dfs(unordered_map<int,vector<int>> &adj , int u ,  vector<bool> &visited){
    visited[u] = true;


    for(int v : adj[u]){
        if(!visited[v]){
            dfs(adj,v,visited);
        }
    }
}
    int findCircleNum(vector<vector<int>>& isConnected) {
        int n = isConnected.size();


        vector<bool> visited(n,false);
```

```
        unordered_map<int,vector<int>> adj;


        for(int i = 0;i<n;i++){
            for(int j = 0;j<n;j++){
                if(isConnected[i][j] == 1){
                    adj[i].push_back(j);
                    adj[j].push_back(i);
                }
            }
        }
        int count = 0;
        for(int i =0;i<n;i++){
            if(!visited[i]){
                dfs(adj,i,visited);
                count++;
            }
        }

        return count;
    }
};
```

1061. Lexicographically Smallest Equivalent String

```
class Solution {
public:
    char dfs(unordered_map<char, vector<char>>& adj, char curr_ch,
             vector<int>& visited) {
        visited[curr_ch - 'a'] = 1;


        char minChar = curr_ch;


        for (char& v : adj[curr_ch]) {
            if (visited[v - 'a'] == 0) {
                minChar = min(minChar, dfs(adj, v, visited));
            }
        }


        return minChar;
    }
    string smallestEquivalentString(string s1, string s2, string baseStr) {
        int n = s1.length();
        int m = baseStr.length();
```

```
        unordered_map<char, vector<char>> adj;


        for (int i = 0; i < n; i++) {
            char u = s1[i];
            char v = s2[i];

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

        for (int i = 0; i < m; i++) {
            char ch = baseStr[i];

            vector<int> visited(26, 0);
            char minChar = dfs(adj, ch, visited);

            result.push_back(minChar);
        }
        return result;
    }
};
```

2359. Find Closest Node to Given Two Nodes

```
class Solution {
public:


void dfs(vector<int> &edges, int node, vector<int> &dist, vector<bool> &visited){
    visited[node] = true;


    int v = edges[node];


    if((v !=-1)  && !visited[v]){
        visited[v] =true;
        dist[v] = 1 + dist[node];
        dfs(edges,v,dist,visited);
    }
}
    int closestMeetingNode(vector<int>& edges, int node1, int node2) {
        int n = edges.size();
```

```
vector<int> dist1(n,INT_MAX);
        vector<int> dist2(n,INT_MAX);


        vector<bool> visited1(n,false);
        vector<bool> visited2(n,false);


        dist1[node1]=0;
        dist2[node2]=0;


        dfs(edges,node1, dist1,visited1);
        dfs(edges,node2, dist2,visited2);



        int minNode = -1;
        int minDist = INT_MAX;


        for(int  i = 0;i<n;i++){
            int maxD = max(dist1[i],dist2[i]);
```

```
  if(minDist > maxD){
                minDist = maxD;
                minNode = i;
            }
        }


        return minNode;
    }
};
```

Learn coding

THANK YOU