DSA SERIES

- Learn Coding

Topic to be Covered today

Graph

841. Keys and Rooms

```
class Solution {
public:


    void dfs(vector<vector<int>>& rooms , int source , vector<bool> &visited){
        visited[source]= true;


        for(int &node : rooms[source]){
            if(!visited[node]){
                dfs(rooms , node ,visited);
            }
        }
    }
    bool canVisitAllRooms(vector<vector<int>>& rooms) {
        int n = rooms.size();
        vector<bool>  visited(n,false);


        dfs(rooms , 0 ,visited);
```

```
for(bool x : visited ){
            if(x == false){
                return false;
            }
        }


        return true;
    }
};
```

797. All Paths From Source to Target

```
class Solution {
public:
    void dfs(vector<vector<int>>& graph, int u, int target,
             vector<vector<int>>& result, vector<int> temp) {
        temp.push_back(u);


        if (u == target) {
            result.push_back(temp);
        } else {
            for (int& v : graph[u]) {
                dfs(graph, v, target, result, temp);
            }
        }


        temp.pop_back();
    }
    vector<vector<int>> allPathsSourceTarget(vector<vector<int>>& graph) {
        int n = graph.size();
        int source = 0;
        int target = n - 1;
```

```
        vector<vector<int>> result;
        vector<int> temp;


        dfs(graph, source, target, result, temp);


        return result;
    }
};
```

2492. Minimum Score of a Path Between Two Cities

```
class Solution {
public:
    void dfs(int u, unordered_map<int, vector<pair<int, int>>>& adj,
             vector<bool>& visited, int& result) {
        visited[u] = true;


        for (auto& P : adj[u]) {
            int v = P.first;
            int d = P.second;


            result = min(result, d);
            if (!visited[v]) {
                dfs(v, adj, visited, result);
            }
        }
    }
    int minScore(int n, vector<vector<int>>& roads) {
        unordered_map<int, vector<pair<int, int>>> adj;
```

```
for (auto& vec : roads) {
            int u = vec[0];
            int v = vec[1];
            int d = vec[2];


            adj[u].push_back({v, d});
            adj[v].push_back({u, d});
        }


        vector<bool> visited(n, false);
        int result = INT_MAX;


        dfs(1, adj, visited, result);


        return result;
    }
};
```

Learn coding

THANK YOU