Domain Winter Camp DAY-6

Name :- Aman Pal

Section/Group:-901-Kpit-A

UID :- 22BCS10188

Problem 1 Minimum Height Trees

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

vector<int> findMinHeightTrees(int n, vector<vector<int>> &edges)
{
    // Edge case: Single node tree
    if (n == 1)
        return {0};

    // Step 1: Build the graph (adjacency list) and degree array
    vector<vector<int>> graph(n);
    vector<int> degree(n, 0);
    for (const auto &edge : edges)
    {
        graph[edge[0]].push_back(edge[1]);
        graph[edge[1]].push_back(edge[0]);
        degree[edge[0]]++;
        degree[edge[1]]++;
    }

    // Step 2: Initialize the queue with leaf nodes (degree == 1)
    queue<int> leaves;
    for (int i = 0; i < n; i++)
    {
        if (degree[i] == 1)
        {
            leaves.push(i);
        }
    }

    // Step 3: Trim the leaves iteratively
    while (n > 2)
    {
        int leavesCount = leaves.size();
        n -= leavesCount; // Remove current level of leaves

        for (int i = 0; i < leavesCount; i++)
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\adity\OneDrive\Desktop\WINTER DOMAIN CAMP\DAY 7> cd '
Roots of Minimum Height Trees: 3 4
```

Problem 2 Accounts Merge

C++ Q2.cpp > ...

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

vector<int> findMinHeightTrees(int n, vector<vector<int>> &edges)
{
    // Edge case: Single node tree
    if (n == 1)
        return {0};

    // Step 1: Build the graph (adjacency list) and degree array
    vector<vector<int>> graph(n);
    vector<int> degree(n, 0);
    for (const auto &edge : edges)
    {
        graph[edge[0]].push_back(edge[1]);
        graph[edge[1]].push_back(edge[0]);
        degree[edge[0]]++;
        degree[edge[1]]++;
    }

    // Step 2: Initialize the queue with leaf nodes (degree == 1)
    queue<int> leaves;
    for (int i = 0; i < n; i++)
    {
        if (degree[i] == 1)
        {
            leaves.push(i);
        }
    }

    // Step 3: Trim the leaves iteratively
    while (n > 2)
    {
        int leavesCount = leaves.size();
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Roots of Minimum Height Trees: 3 4
PS C:\Users\adity\OneDrive\Desktop\WINTER DOMAIN CAMP\DAY 7> c
Roots of Minimum Height Trees: 3 4
```

Problem 3 Rotting Oranges

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

class UnionFind
{
public:
    vector<int> parent, rank;

    UnionFind(int n)
    {
        parent.resize(n);
        rank.resize(n, 0);
        for (int i = 0; i < n; ++i)
            parent[i] = i;
    }

    int find(int x)
    {
        if (parent[x] != x)
        {
            parent[x] = find(parent[x]); // Path compression
        }
        return parent[x];
    }

    void unionSet(int x, int y)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

c:\mingw\lib\gcc\mingw32\6.3.0\include\c++\bits\stl_queue.h:247:7
td::pair<int, int>; _Sequence = std::deque<std::pair<int, int>, s
       push(value_type&& __x)
       ^~~~
c:\mingw\lib\gcc\mingw32\6.3.0\include\c++\bits\stl_queue.h:247:7
nt, int> >::value_type&& {aka std::pair<int, int>&&}'
PS C:\Users\adity\OneDrive\Desktop\WINTER DOMAIN CAMP\DAY 7> cd "
John johnnybravo@mail.com
John john00@mail.com john_newyork@mail.com johnsmith@mail.com
Mary mary@mail.com
```

Problem 4 Pacific Atlantic water Flow

C++ Q4.cpp > ...

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

void dfs(vector<vector<int>> &heights, vector<vector<bool>> &visited, int x, int y, int prevHeight)
{
    int m = heights.size(), n = heights[0].size();

    // Base conditions
    if (x < 0 || x >= m || y < 0 || y >= n || visited[x][y] || heights[x][y] < prevHeight)
    {
        return;
    }

    visited[x][y] = true;

    // Explore in all 4 directions
    vector<pair<int, int>> directions = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
    for (auto &dir : directions)
    {
        dfs(heights, visited, x + dir.first, y + dir.second, heights[x][y]);
    }
}

vector<vector<int>> pacificAtlantic(vector<vector<int>> &heights)
{
    int m = heights.size(), n = heights[0].size();

    // Create visited matrices for both oceans
    vector<vector<bool>> pacific(m, vector<bool>(n, false));
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
cd "c:\Users\adity\OneDrive\De
Cells that can flow to both oceans:
[0, 4]
[1, 3]
[1, 4]
[2, 2]
[3, 0]
[3, 1]
[4, 0]
```

Problem 5 Network Delay Time

Q5.cpp > ...

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

int networkDelayTime(vector<vector<int>> &times, int n, int k)
{
    // Step 1: Build the adjacency list
    vector<vector<pair<int, int>>> graph(n + 1);
    for (const auto &time : times)
    {
        int u = time[0], v = time[1], w = time[2];
        graph[u].emplace_back(v, w);
    }

    // Step 2: Min-heap to perform Dijkstra's algorithm
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
    vector<int> dist(n + 1, INT_MAX);
    dist[k] = 0;
    pq.emplace(0, k); // {distance, node}

    while (!pq.empty())
    {
        int curDist = pq.top().first;
        int node = pq.top().second;
        pq.pop();

        // If the current distance is greater than the stored distance, skip
        if (curDist > dist[node])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\adity\OneDrive\Desktop\WINTER DOMAIN CAMP\DAY 7> cd "c:\Users\adity\
Minimum time for all nodes to receive the signal: 2
```