**Name: Amber Shukla**

**Batch: A1-B3-42**

**DAA Lab**

**Practical – 2 Fast Learners Problem**

**Minimum Spanning Forest:**

a. Consider a fixed input graph/matrix 20x20 size from a file. The matrix should be such that there are minimum of 3 disconnected components.

b. Find the connected components.

c. Run Prim’s separately on each connected component to get a Minimum Spanning Forest

**Code:**

#include <bits/stdc++.h>

using namespace std;

const int N = 20;

const int INF = 99;

void dfs(int u, int cost[N][N], bool visited[N], int component[], int &size, int n) {

    visited[u] = true;

    component[size++] = u;

    for (int v = 0; v < n; v++) {

        if (!visited[v] && cost[u][v] < INF) {

            dfs(v, cost, visited, component, size, n);

        }

    }

}

int main() {

    int n = N, cost[N][N];

    std::ifstream fin("matrix.txt");

    if (!fin) {

        std::cerr << "Error: cannot open 'matrix.txt'." << std::endl;

        return 1;

    }

    for (int i = 0; i < n; i++)

        for (int j = 0; j < n; j++)

            if (!(fin >> cost[i][j])) {

                std::cerr << "Error: expected 400 numbers in 'matrix.txt'." << std::endl;

                return 1;

            }

    fin.close();

    bool visited[N] = {false};

    int comp\_count = 0;

    int components[N][N], comp\_size[N] = {0};

    for (int i = 0; i < n; i++) {

        if (!visited[i]) {

            int size = 0;

            dfs(i, cost, visited, components[comp\_count], size, n);

            comp\_size[comp\_count] = size;

            comp\_count++;

        }

    }

    std::cout << "Found " << comp\_count << " connected component(s).\n";

    for (int cid = 0; cid < comp\_count; cid++) {

        std::cout << " Component " << cid + 1 << " size = " << comp\_size[cid] << " (vertices:";

        for (int k = 0; k < comp\_size[cid]; k++) {

            if (k) std::cout << ",";

            std::cout << components[cid][k];

        }

        std::cout << ")\n";

    }

    int total\_cost = 0;

    for (int cid = 0; cid < comp\_count; cid++) {

        int m = comp\_size[cid];

        if (m == 0) continue;

        std::cout << "\n--- Prim's on Component " << cid + 1 << " ---\n";

        if (m == 1) {

            std::cout << " Single vertex, no edges. Cost = 0\n";

            continue;

        }

        bool selected[N] = {false};

        int comp\_cost = 0;

        selected[components[cid][0]] = true;

        for (int edges = 0; edges < m - 1; edges++) {

            int min\_cost = INF, u = -1, v = -1;

            for (int ia = 0; ia < m; ia++) {

                int i = components[cid][ia];

                if (selected[i]) {

                    for (int jb = 0; jb < m; jb++) {

                        int j = components[cid][jb];

                        if (!selected[j] && cost[i][j] < min\_cost) {

                            min\_cost = cost[i][j];

                            u = i; v = j;

                        }

                    }

                }

            }

            if (u != -1 && v != -1 && min\_cost < INF) {

                selected[v] = true;

                std::cout << u << " - " << v << " : " << min\_cost << "\n";

                comp\_cost += min\_cost;

            }

        }

        std::cout << "Total Minimum Cost for Component " << cid + 1 << ": " << comp\_cost << "\n";

        total\_cost += comp\_cost;

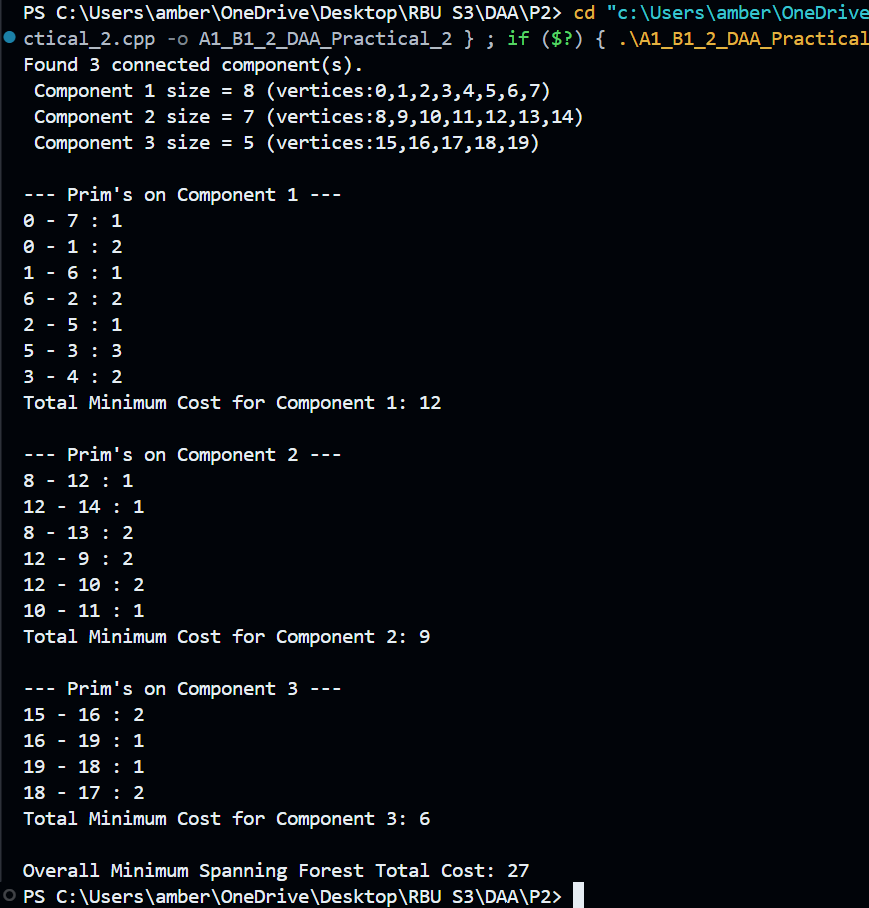
    }

    std::cout << "\nOverall Minimum Spanning Forest Total Cost: " << total\_cost << std::endl;

    return 0;

}

**Output:**

****