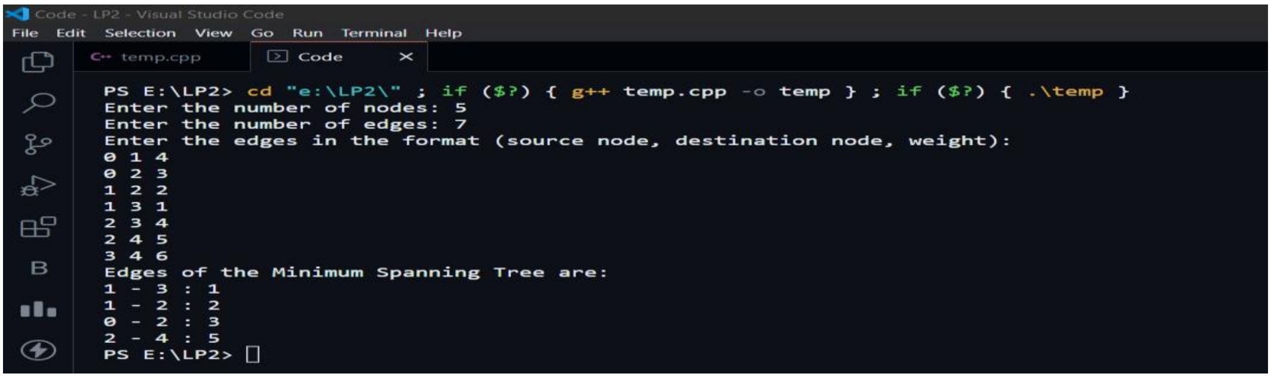
Assignment No 3

Minimum Spanning Tree

Kruskal's algorithm:

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

// Structure to represent an edge
struct Edge {
    int src, dest, weight;
};

// Structure to represent a graph
struct Graph {
    int V, E;
    vector<Edge> edges;
};

// Function to compare two edges according to their weight
bool compareEdges(const Edge& a, const Edge& b) {
    return a.weight < b.weight;
}

// Function to find the parent of a node in a disjoint set
int findParent(int node, vector<int>& parent) {
    if (parent[node] == node) {
        return node;
    }
    return findParent(parent[node], parent);
}

// Function to perform Union operation of two sets in a disjoint set
void unionSets(int x, int y, vector<int>& parent) {
    int xParent = findParent(x, parent);
    int yParent = findParent(y, parent);
    parent[xParent] = yParent;
}

// Function to find the Minimum Spanning Tree of a graph using Kruskal's algorithm
void kruskalMST(Graph graph) {
    int V = graph.V;
    vector<Edge> result; // Vector to store the edges of the MST

    // Sort the edges in non-decreasing order of their weight
    sort(graph.edges.begin(), graph.edges.end(), compareEdges);

    // Create a disjoint set for all the nodes
    vector<int> parent(V);
    for (int i = 0; i < V; i++) {
        parent[i] = i;
    }
    // Process each edge in the sorted order
    for (auto edge : graph.edges) {
        int x = edge.src;
        int y = edge.dest;

        // If including this edge doesn't form a cycle, include it in the MST
        if (findParent(x, parent) != findParent(y, parent)) {
            result.push_back(edge);
            unionSets(x, y, parent);
        }
    }

    // Print the edges of the MST
    cout << "Edges of the Minimum Spanning Tree are:\n";
    for (auto edge : result) {
        cout << edge.src << " - " << edge.dest << " : " << edge.weight << "\n";
    }
}

int main() {
    // Get user input for the number of nodes and edges
    int V, E;
    cout << "Enter the number of nodes: ";
    cin >> V;
    cout << "Enter the number of edges: ";
    cin >> E;

    // Create a graph
    Graph graph;
    graph.V = V;
    graph.E = E;

    // Get user input for the edges
    cout << "Enter the edges in the format (source node, destination node, weight):\n";
    for (int i = 0; i < E; i++) {
        Edge edge;
        cin >> edge.src >> edge.dest >> edge.weight;
        graph.edges.push_back(edge);
    }

    // Find the Minimum Spanning Tree of the graph
    kruskalMST(graph);

    return 0;
}
```



Code - LP2 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
temp.cpp
Code

```
PS E:\LP2> cd "e:\LP2\" ; if ($?) { g++ temp.cpp -o temp } ; if ($?) { .\temp }
Enter the number of nodes: 5
Enter the number of edges: 7
Enter the edges in the format (source node, destination node, weight):
0 1 4
0 2 3
1 2 2
1 3 1
2 3 4
2 4 5
3 4 6
Edges of the Minimum Spanning Tree are:
1 - 3 : 1
1 - 2 : 2
0 - 2 : 3
2 - 4 : 5
PS E:\LP2>
```