Data Structures and Algorithms
(CS09203)

Lab Report

Name:	Farhan Naseer
Registration #:	SEU-F16-125
Lab Report #:	11
Dated:	28-06-2018
Submitted To:	Sir. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 11
Implementation of Kruskal Algorithms

Objective
The objective of this session is to implement and understand kruskal algorithms.

Software Tool
1. I use Code Blocks with GCC compiler.

1 Theory

This section discusses how to create the graph and tell the number of edges and vertices . trees are used to model electrical circuits, chemical compounds, highway maps, and so on. They are also used in the analysis of electrical circuits, finding the shortest route, project planning, linguistics, genetics, social science, and so forth Undirected Edge - An undirected egde is a bidirectional edge. If there is a undirected edge between vertices A and B then edge (A , B) is equal to edge (B , A). Directed Edge - A directed egde is a unidirectional edge. If there is a directed edge between vertices A and B then edge (A , B) is not equal to edge (B , A). Weighted Edge - A weighted egde is an edge with cost on it.

2 Task

2.1 Procedure: Task 5

Write a C++ code using functions for the following operations. 1.Krsukal Algorithms

2.2

```
#include<bits/stdc++.h>
```

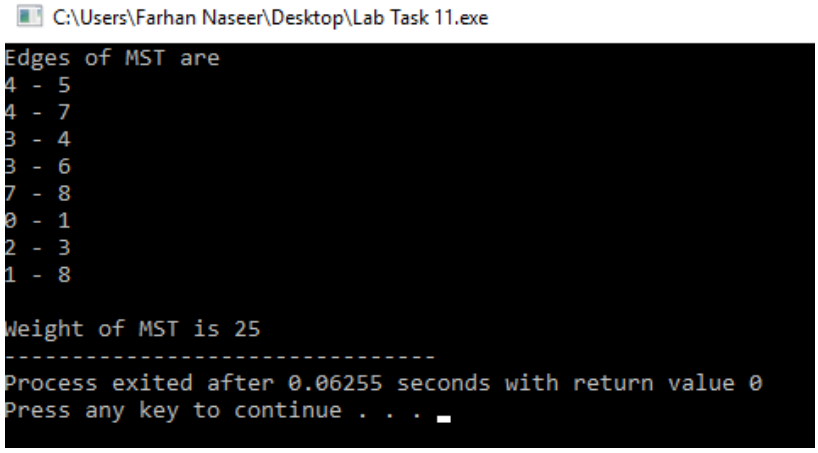

Figure 1: output

```
using namespace std;

typedef  pair<int, int> iPair;


struct Graph
{
    int V, E;
    vector< pair<int, iPair> > edges;


    Graph(int V, int E)
    {
        this->V = V;
        this->E = E;
    }


    void addEdge(int u, int v, int w)
    {
        edges.push_back({w, {u, v}});
    }
```

```
    int kruskalMST();
};


struct DisjointSets
{
    int *parent, *rnk;
    int n;


    DisjointSets(int n)
    {

        this->n = n;
        parent = new int[n+1];
        rnk = new int[n+1];


        for (int i = 0; i <= n; i++)
        {
            rnk[i] = 0;

            parent[i] = i;
        }
    }

    int find(int u)
    {

        if (u != parent[u])
            parent[u] = find(parent[u]);
        return parent[u];
    }

    void merge(int x, int y)
    {
        x = find(x), y = find(y);

        if (rnk[x] > rnk[y])
            parent[y] = x;
```

```
        else
            parent[x] = y;

        if (rnk[x] == rnk[y])
            rnk[y]++;
    }
};

int Graph::kruskalMST()
{
    int mst_wt = 0;
    sort(edges.begin(), edges.end());

    DisjointSets ds(V);

    vector< pair<int, iPair> >::iterator it;
    for (it=edges.begin(); it!=edges.end(); it++)
    {
        int u = it->second.first;
        int v = it->second.second;

        int set_u = ds.find(u);
        int set_v = ds.find(v);

        if (set_u != set_v)
        {
            cout << u << " - " << v << endl;

            mst_wt += it->first;

            ds.merge(set_u, set_v);
        }
    }

    return mst_wt;
}

int main()
{
```

```
int V = 9, E = 14;
Graph g(V, E);
g.addEdge(0, 1, 4);
g.addEdge(0, 7, 8);
g.addEdge(1, 2, 8);
g.addEdge(1, 8,5 );
g.addEdge(1, 6, 10);
g.addEdge(2, 6, 4);
g.addEdge(2, 3, 4);
g.addEdge(2, 8, 4);
g.addEdge(2, 5, 4);
g.addEdge(2, 1, 8);
g.addEdge(3, 6, 3);
g.addEdge(3, 2, 4);
g.addEdge(3, 4, 3);
g.addEdge(4, 3, 3);

g.addEdge(4, 6, 6);
g.addEdge(4, 5, 1);
g.addEdge(4, 7, 2);
g.addEdge(5, 2, 4);
g.addEdge(5, 7, 3);
g.addEdge(5, 4, 1);

g.addEdge(6, 1, 10);
g.addEdge(6, 2, 4);
g.addEdge(6, 3, 3);
g.addEdge(6, 4, 6);

g.addEdge(7, 4, 2);
g.addEdge(7, 5, 3);
g.addEdge(7, 8, 3);
g.addEdge(8, 1, 5);
g.addEdge(8, 2, 4);
g.addEdge(8, 5, 3);
cout << "Edges of MST are \n";
int mst_wt = g.kruskalMST();

cout << "\nWeight of MST is " << mst_wt;
```

```
    return 0;
}
```

3 Conclusion

In today lab we have discussed how we can create a tree for krsukal algorithms and how to display it on a screen by a code.