ASSIGNMENT 5

Aim: You have a business with several offices; you want to lease phone lines to connect them up with each other and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting appropriate data structures.

Objective: To understand the concept of minimum spanning tree and finding the minimum cost of tree using Kruskals algorithm.

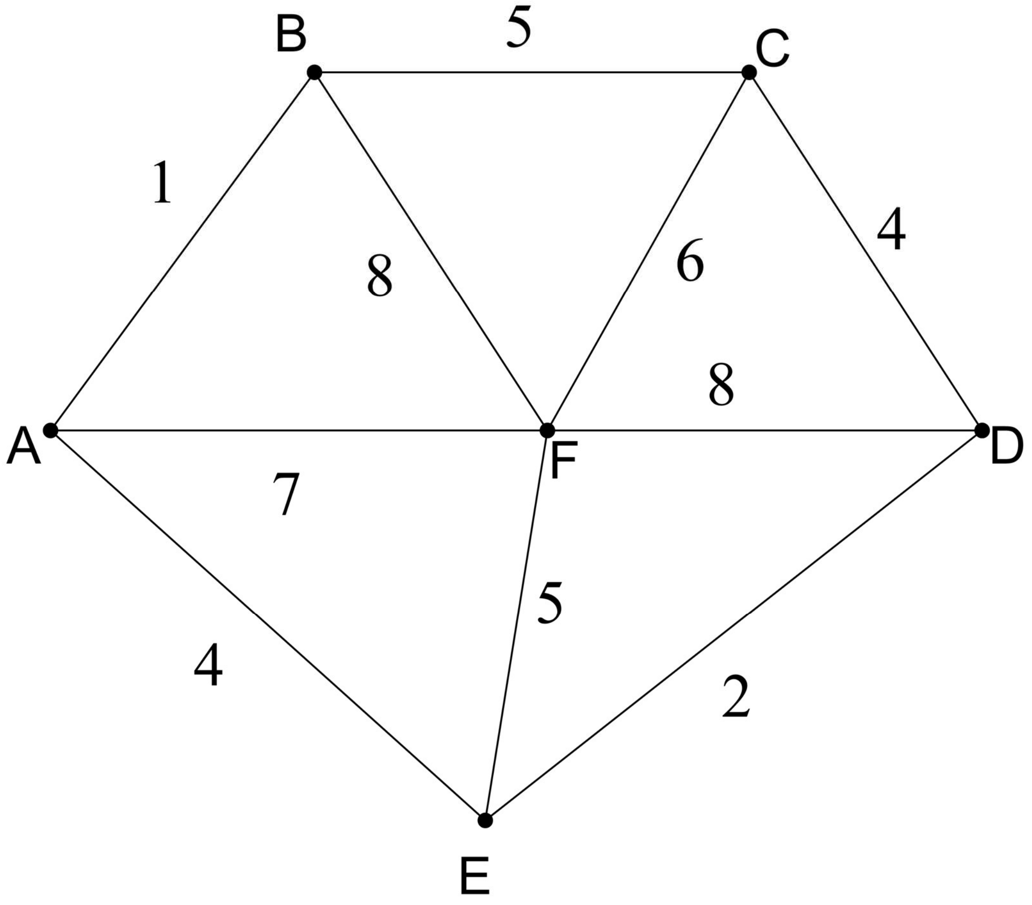
Theory: A spanning tree of the graph is a connected (if there is at least one path between every pair of vertices in a graph) subgraph in which there are no cycle. Suppose you have a connected undirected graph with a weight (or cost) associated with each edge. The cost of a spanning tree would be the sum of the costs of its edges. A minimum-cost spanning tree is a spanning tree that has the lowest cost. There are two basic algorithms for finding minimum-cost spanning trees: 1. Prim's Algorithm 2. Kruskal's Algorithm .

Kruskals's algorithm: It tarts with no nodes or edges in the spanning tree, and repeatedly add the cheapest edge that does not create a cycle.

Steps of Kruskal's Algorithm to find minimum spanning tree:

1. Select the shortest edge in a network
2. Select the next shortest edge which does not create a cycle
3. Repeat step 2 untill spanning tree has n-1 edges.

Example:



The solution is

AB	1
ED	2
CD	4
AE	4
EF	5

Total weight of tree: 16

Algorithm:

- Algorithm kruskal(G,V,E,T)

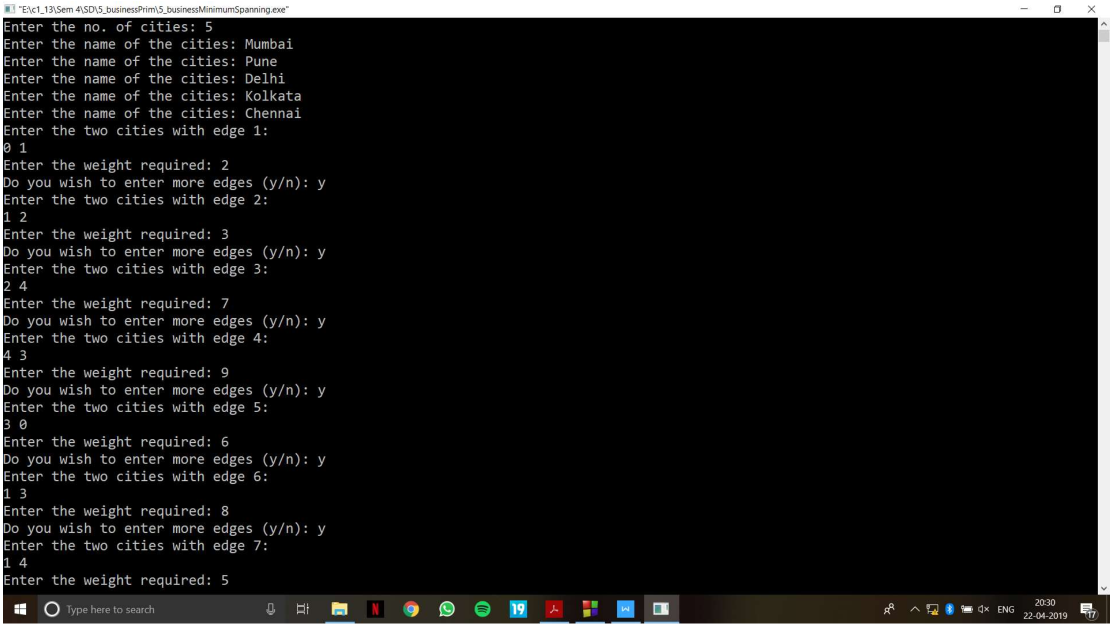
```
{
 1.Sort E in increasing order of weight
 2.let G=(V,E) and T=(A,B),A=V ,B is null
   set and let n =count(V)
3.Initialize n set ,each containing a different element of v.
4.while(|B|<n-1) do
    begin
     e=<u,v>the shortest edge not yet considered
     U=Member(u)
```

```
       V=Member(v)
     if( Union(U,V))
         update in B and add the cost
      } }
   end
5.T is the minimum spanning tree
}
```

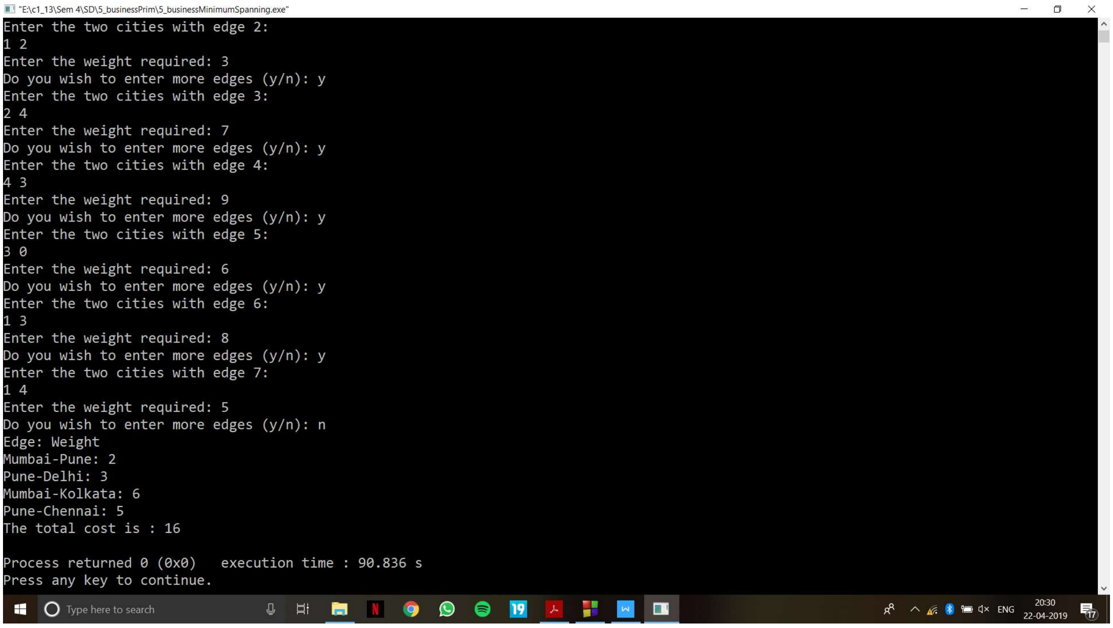
Program code:

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

using namespace std;

int minKey(int key[], bool mstSet[], int cities)
{
    int min = INT_MAX, min_index;

    for (int v = 0; v < cities; v++)
        if (mstSet[v] == false && key[v] < min)
            min = key[v], min_index = v;
    return min_index;
}

int main()
{
    int edges = 0,c1,c2,weight,cities,total = 0;
    char ch = 'y';
    cout<<"Enter the no. of cities: ";
    cin>>cities;
    string name[cities];
    for(int i=0;i<cities;i++)
    {
        cout<<"Enter the name of the cities: ";
        cin>>name[i];
    }
    int g[cities][cities];
    for(int i=0;i<cities;i++)
    {
        for(int j=0;j<cities;j++)
        {
            g[i][j] = 0;
        }
    }
    while(ch == 'y')
    {
        cout<<"Enter the two cities with edge "<<edges+1<<": "<<endl;
        cin>>c1>>c2;
        cout<<"Enter the weight required: ";
        cin>>weight;
        g[c1][c2] = weight;
        g[c2][c1] = weight;
        edges++;
        cout<<"Do you wish to enter more edges (y/n): ";
        cin>>ch;
    }
    int parent[cities],key[cities];
    bool mstSet[cities];
    for (int i = 0; i < cities; i++)
    {
        key[i] = INT_MAX;
        mstSet[i] = false;
    }
    key[0] = 0;
    parent[0] = -1;
    for (int count = 0; count < cities-1; count++)
    {
        int u = minKey(key, mstSet,cities);
        mstSet[u] = true;
        for (int c = 0; c < cities; c++)
        if (g[u][c] && mstSet[c] == false && g[u][c] < key[c])
            parent[c] = u, key[c] = g[u][c];
    }
    cout<<"Edge: Weight"<<endl;
    for(int i=1;i<cities;i++)
    {
        cout<<name[parent[i]]<<"-"<<name[i]<<": "<<g[i][parent[i]]<<endl;
        total = total + g[i][parent[i]];
    }
    cout<<"The total cost is : "<<total<<endl;
    return 0;
}
```

Output:


"E:\c1_13\Sem 4\SD\5_businessPrim\5_businessMinimumSpanning.exe"
Type here to search
ENG
20:30
22-04-2019

```
Enter the no. of cities: 5
Enter the name of the cities: Mumbai
Enter the name of the cities: Pune
Enter the name of the cities: Delhi
Enter the name of the cities: Kolkata
Enter the name of the cities: Chennai
Enter the two cities with edge 1:
0 1
Enter the weight required: 2
Do you wish to enter more edges (y/n): y
Enter the two cities with edge 2:
1 2
Enter the weight required: 3
Do you wish to enter more edges (y/n): y
Enter the two cities with edge 3:
2 4
Enter the weight required: 7
Do you wish to enter more edges (y/n): y
Enter the two cities with edge 4:
4 3
Enter the weight required: 9
Do you wish to enter more edges (y/n): y
Enter the two cities with edge 5:
3 0
Enter the weight required: 6
Do you wish to enter more edges (y/n): y
Enter the two cities with edge 6:
1 3
Enter the weight required: 8
Do you wish to enter more edges (y/n): y
Enter the two cities with edge 7:
1 4
Enter the weight required: 5
```


"E:\c1_13\Sem 4\SD\5_businessPrim\5_businessMinimumSpanning.exe"
Type here to search
ENG
20:30
22-04-2019

```
Enter the two cities with edge 2:
1 2
Enter the weight required: 3
Do you wish to enter more edges (y/n): y
Enter the two cities with edge 3:
2 4
Enter the weight required: 7
Do you wish to enter more edges (y/n): y
Enter the two cities with edge 4:
4 3
Enter the weight required: 9
Do you wish to enter more edges (y/n): y
Enter the two cities with edge 5:
3 0
Enter the weight required: 6
Do you wish to enter more edges (y/n): y
Enter the two cities with edge 6:
1 3
Enter the weight required: 8
Do you wish to enter more edges (y/n): y
Enter the two cities with edge 7:
1 4
Enter the weight required: 5
Do you wish to enter more edges (y/n): n
Edge: Weight
Mumbai-Pune: 2
Pune-Delhi: 3
Mumbai-Kolkata: 6
Pune-Chennai: 5
The total cost is : 16

Process returned 0 (0x0)   execution time : 90.836 s
Press any key to continue.
```

Conclusion: Kruskal's algorithm can be shown to run in O(**E log E**) time, where E is the number of edges in the graph. Thus we have connected all the offices with a total minimum cost using kruskal’s algorithm.