ASSIGNMENT NO.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 study the use of kruskal's and prims algorithm in given problem.

Theory:- *What is Minimum Spanning Tree?*

Given a connected and undirected graph, a *spanning tree* of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. A *minimum spanning tree (MST)* or minimum weight spanning tree for a weighted, connected and undirected graph is a spanning tree with weight less than or equal to the weight of every other spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.

How many edges does a minimum spanning tree has?

A minimum spanning tree has (V – 1) edges where V is the number of vertices in the given graph.

What are the applications of Minimum Spanning Tree?

See this for applications of MST.

Below are the steps for finding MST using Kruskal's algorithm

1. *Sort all the edges in non-decreasing order of their weight.*
2. *Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge. Else, discard it.*
3. *Repeat step#2 until there are (V-1) edges in the spanning tree.*

The step#2 uses Union-Find algorithm to detect cycle.

The algorithm is a Greedy Algorithm. The Greedy Choice is to pick the smallest weight edge that does not cause a cycle in the MST constructed so far.

Algorithm:-

- create a forest *F* (a set of trees), where each vertex in the graph is a separate tree
- create a set *S* containing all the edges in the graph
- while *S* is nonempty and *F* is not yet spanning
  - remove an edge with minimum weight from *S*
  - if the removed edge connects two different trees then add it to the forest *F*, combining two trees into a single tree

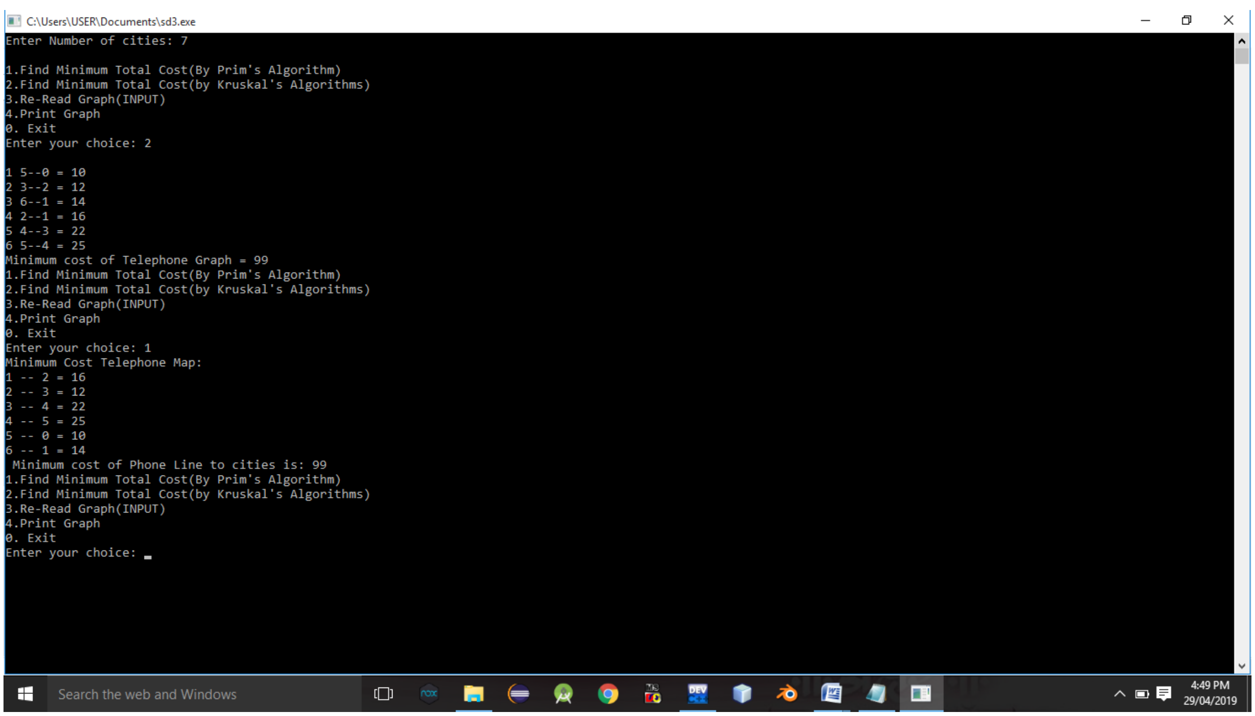
At the termination of the algorithm, the forest forms a minimum spanning forest of the graph. If the graph is connected, the forest has a single component and forms a minimum spanning tree.

Program Code:-

```
#include <iostream>
using namespace std;
#define V 5
int parent[V];
int find(int i)
{
    while (parent[i] != i)
        i = parent[i];
    return i;
}
void union1(int i, int j)
{
    int a = find(i);
    int b = find(j);
    parent[a] = b;
}
void kruskalMST(int cost[][V])
{
    int mincost = 0;
    for (int i = 0; i < V; i++)
        parent[i] = i;
    int edge_count = 0;
    while (edge_count < V - 1) {
        int min = INT_MAX, a = -1, b = -1;
        for (int i = 0; i < V; i++) {
            for (int j = 0; j < V; j++) {
                if (find(i) != find(j) && cost[i][j] < min) {
                    min = cost[i][j];
                    a = i;
                    b = j;
                }
            }
        }

        union1(a, b);
        cout<<"Edge "<< edge_count++<<":"<<a<<"-"<<b<<" cost:
"<<min<<endl;
        mincost += min;
    }
    cout<<"\n Minimum cost= "<< mincost;
}
int main()
{
   int i,j,n,noofedges,costmat[V][V];
```

```
    cout<<"Enter total number of offices:";
    cin>>n;

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
        {
            costmat[i][j]=999;
        }
    }
    cout<<"Enter number of phone lines";
    cin>>noofedges;
    for(i=0;i<noofedges;i++)
    {
        int a,b,cost;
        cout<<"Enter the two cities and corresponding cost";
        cin>>a>>b>>cost;
        costmat[a][b]=cost;
        costmat[b][a]=cost;
    }
    kruskalMST(costmat);

    return 0;
}
```

Output Screenshots:-



Conclusion:- Thus,we have studied implementation of kruskal’s algorithm.