ASSIGNMENT 5

AIM:-

You have a business with several offices,you want to lease a phone line to connect to them up with each other.and the phone company charges different amount of money to connect to different pair of cities . You want to set of lines that connects all your offices with minimum cost.Solve by using appropriate data structure.

OBJECTIVE:-

Implement Kruskal algorithm to determine the minimum cost require to connect offices.

THEORY:-

Given a connected and undirected graph, a *spanning tree* of that graph is a sub graph 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 un directed 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.

Kruskal's algorithm is a minimum-spanning-tree algorithm which finds an edge of the least possible weight that connects any two trees in the forest.[1] It is a greedy algorithm in graph theory as it finds a minimum spanning tree for a connected weighted graph adding increasing cost arcs at each step.[1] This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. If the graph is not connected, then it finds a *minimum spanning forest* (a minimum spanning tree for each connected component).

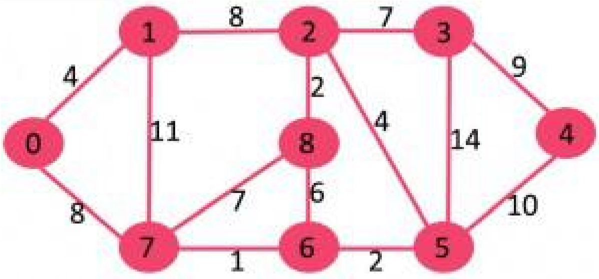
Steps:-

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.*

Example:-



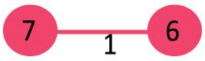
The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having (9 – 1) = 8 edges.

After sorting:

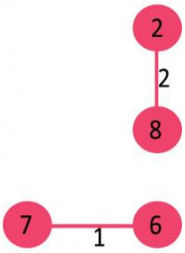
Weight	Src	Dest
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Now pick all edges one by one from sorted list of edges

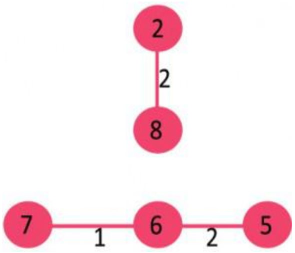
1. Pick edge 7-6: No cycle is formed, include it.



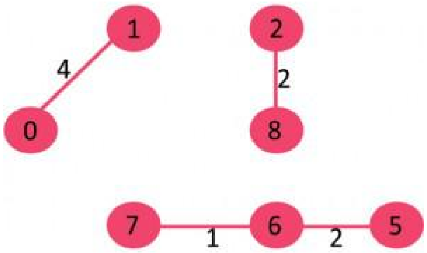
2. Pick edge 8-2: No cycle is formed, include it.



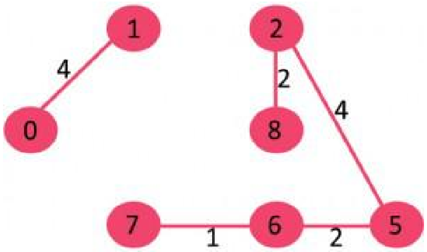
3. Pick edge 6-5: No cycle is formed, include it.



4. Pick edge 0-1: No cycle is formed, include it.

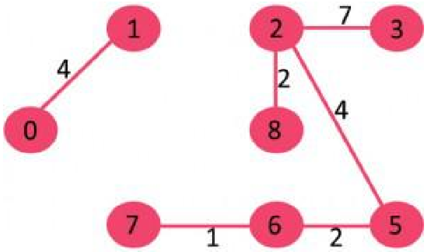


5. Pick edge 2-5: No cycle is formed, include it.



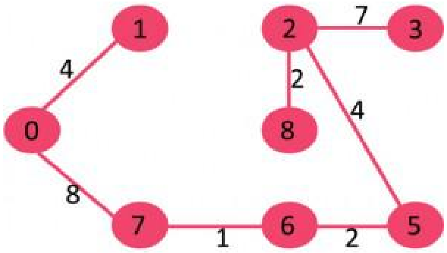
6. Pick edge 8-6: Since including this edge results in cycle, discard it.

7. Pick edge 2-3: No cycle is formed, include it.



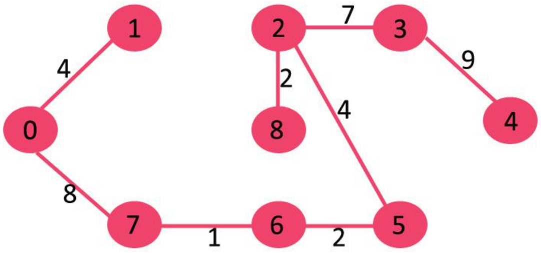
8. Pick edge 7-8: Since including this edge results in cycle, discard it.

9. Pick edge 0-7: No cycle is formed, include it.



10. Pick edge 1-2: Since including this edge results in cycle, discard it.

11. Pick edge 3-4: No cycle is formed, include it.



ALGORITHM:-

KRUSHAL ALGORITHM:

```
void krushal::mincost()
{
        int count,k,v1,v2,i,j,tree[10][10],pos,parent[10];
        int sum=0;
        count=0;
        k=0;
        for(i=0;i<vertices;i++)
                parent[i]=i;
        while(count!=vertices-1)
        {
        pos=minimum(edges);
        if(pos==-1)
                break;
        v1=G[pos].v1;
        v2=G[pos].v2;
        i=find(v1,parent);
        j=find(v2,parent);
        if(i!=j)
                {
                tree[k][0]=v1;
                tree[k][1]=v2;
```

```
			k++;
			count++;
			sum=sum+G[pos].cost;
			uni(i,j,parent);
			}
		G[pos].cost=MAX;
		}
		if(count==vertices-1)
		{
			cout<<"spanning tree is"<<endl;
			for(i=0;i<vertices-1;i++)
			{
				cout<<tree[i][0]<<"-"<<tree[i][1]<<endl;
			}
			cout<<"cost required to set cables"<<sum<<endl;
		}
		else
		{
			cout<<"connection can't be set up"<<endl;
		}
}
```

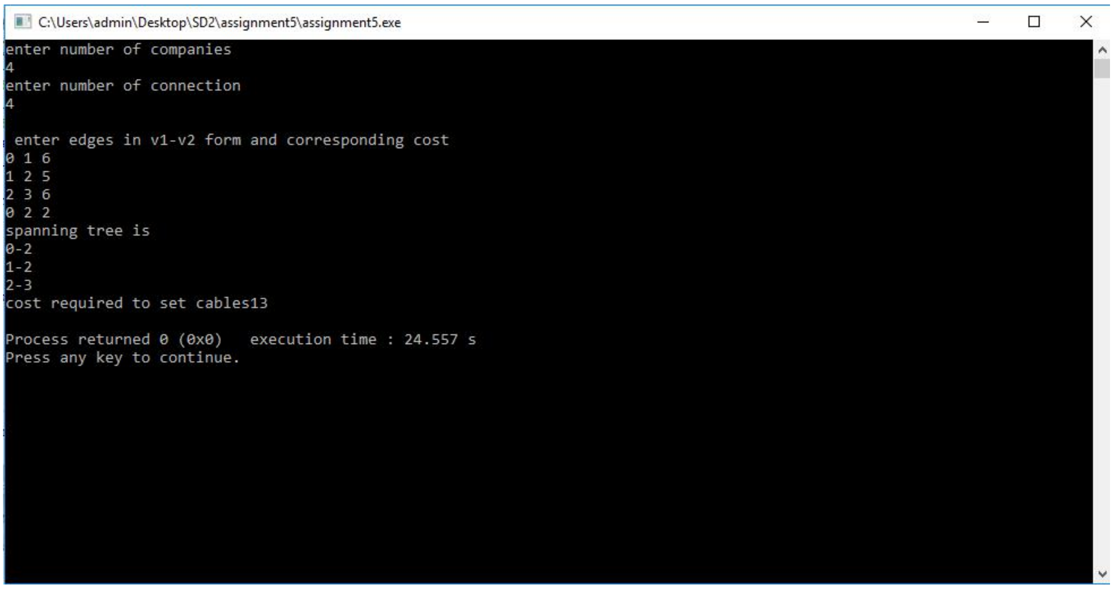
CODE:-

```
#include<iostream>
#define MAX 999
using namespace std;
class krushal
{
private:
	struct node
	{
		int v1,v2,cost;
	}G[20];
public:
	int edges,vertices;
	void create();
	void mincost();
	void input();
	int minimum(int);
};
```

```
int find (int v2,int parent[])
{
        while(parent[v2]!=v2)
        {
                v2=parent[v2];
        }
}
void uni(int i,int j,int parent[])
{
        if(i<j)
                parent[j]=i;
        else
                parent[i]=j;
}
void krushal::input()
{
        cout<<"enter number of companies"<<endl;
        cin>>vertices;
        cout<<"enter number of connection"<<endl;
        cin>>edges;
}
void krushal::create()
{
        cout<<"\n enter edges in v1-v2 form and corresponding
cost"<<endl;
        for(int k=0;k<edges;k++)
        {
            cin>>G[k].v1>>G[k].v2>>G[k].cost;
        }
}
int krushal::minimum(int n)
{
        int i,small,pos;
        small=MAX;
        pos=-1;
        for(i=0;i<n;i++)
        {
                if(G[i].cost<small)
                {
                        small=G[i].cost;
                pos=i;
            }
        }
        return pos;
}
void krushal::mincost()
{
        int count,k,v1,v2,i,j,tree[10][10],pos,parent[10];
        int sum=0;
        count=0;
        k=0;
        for(i=0;i<vertices;i++)
            parent[i]=i;
        while(count!=vertices-1)
        {
        pos=minimum(edges);
        if(pos==-1)
            break;
        v1=G[pos].v1;
        v2=G[pos].v2;
        i=find(v1,parent);
        j=find(v2,parent);
        if(i!=j)
            {
            tree[k][0]=v1;
            tree[k][1]=v2;
            k++;
            count++;
            sum=sum+G[pos].cost;
            uni(i,j,parent);
            }
        G[pos].cost=MAX;
        }
        if(count==vertices-1)
        {
            cout<<"spanning tree is"<<endl;
            for(i=0;i<vertices-1;i++)
            {
                cout<<tree[i][0]<<"-"<<tree[i][1]<<endl;
            }
```

```
                cout<<"cost required to set cables"<<sum<<endl;
        }
        else
        {
                cout<<"connection can't be set up"<<endl;
        }
}
int main()
{
        krushal tr;
        tr.input();
        tr.create();
        tr.mincost();
}
```

OUTPUT:-



```
enter number of companies
4
enter number of connection
4

 enter edges in v1-v2 form and corresponding cost
0 1 6
1 2 5
2 3 6
0 2 2
spanning tree is
0-2
1-2
2-3
cost required to set cables13

Process returned 0 (0x0)   execution time : 24.557 s
Press any key to continue.
```

CONCLUSION:-

We have successfully implemented Kruskal algorithm to determine the Minimum cost.