**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>

#define MAX 999;

using namespace std;

class kruskal

{

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 kruskal::input()

{

cout<<"enter number of companies"<<endl;

cin>>vertices;

cout<<"enter number of connection"<<endl;

cin>>edges;

}

void kruskal::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 kruskal::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 kruskal::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()

{

kruskal k;

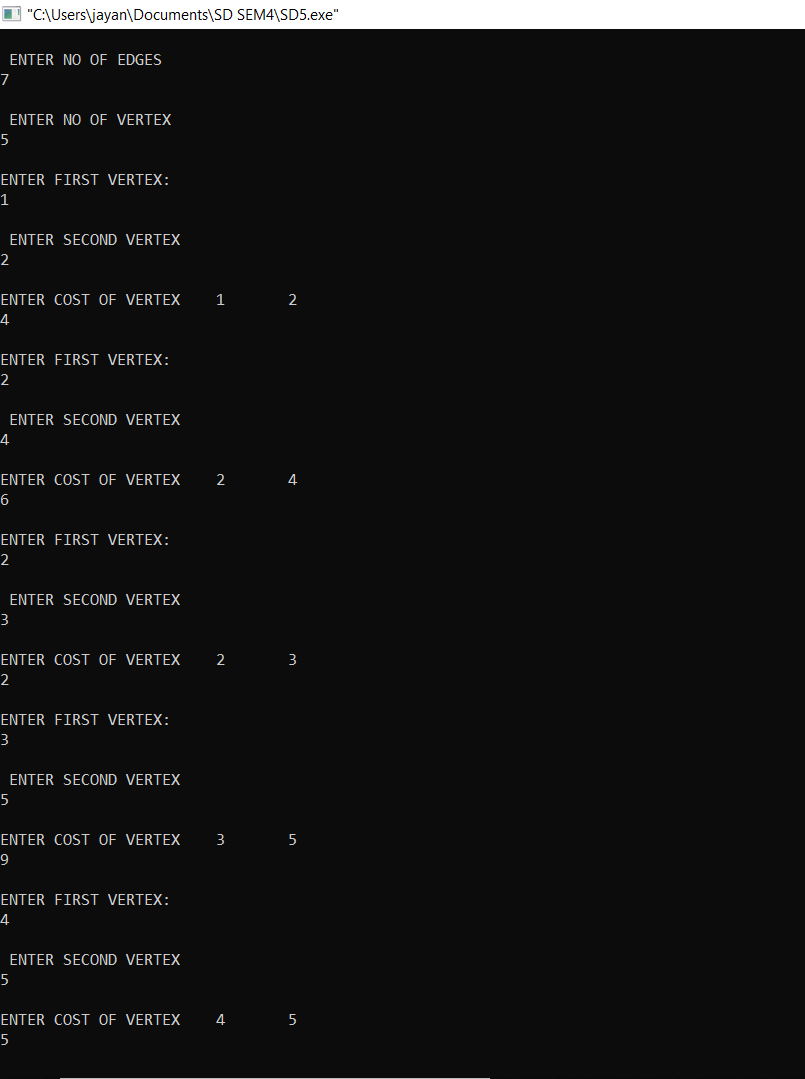
k.input();

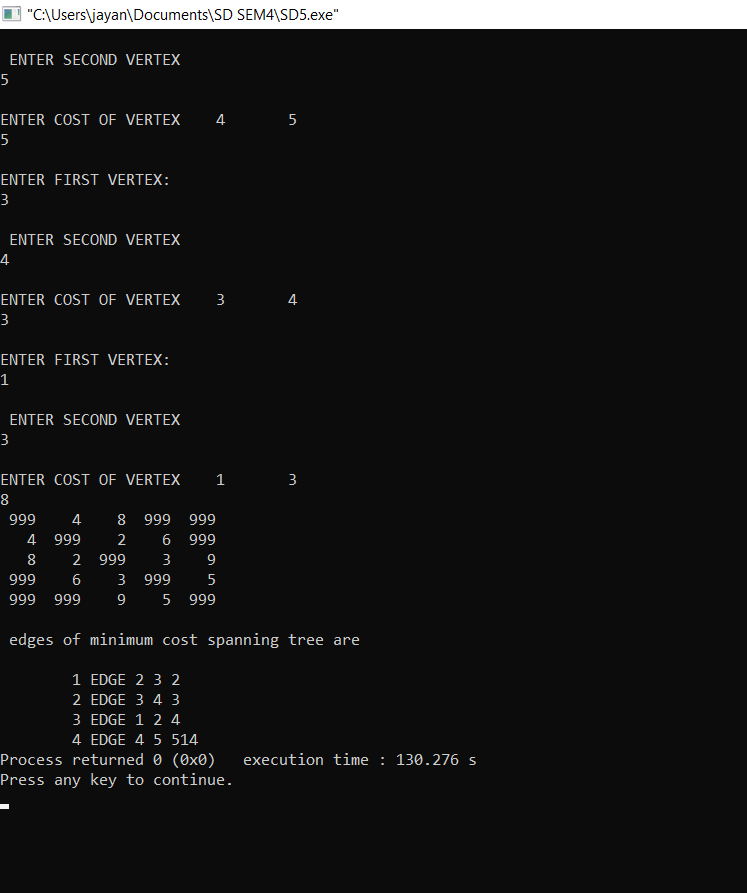
k.create();

k.mincost();

}

**Output:**

****

****

**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.