Lab Experiment No.7 (kruskal\_algo)

-----------------------------------------------------------------------------------------------------------------------------

Name: Prathamesh Sadashiv Gadekar

Roll No.:14

Class : SE-IT

-----------------------------------------------------------------------------------------------------------------------------

Represent a graph of your college campus using adjacency list /adjacency matrix. Nodes should represent the various departments/institutes and links should represent the distance between them. Find minimum spanning tree

a) Using Kruskal’s algorithm.

-----------------------------------------------------------------------------------------------------------------------------

#include<iostream>

#include<string.h>

using namespace std;

class Graph

{

char vertices[10][10];

int cost[10][10],no;

public:

Graph();

void creat\_graph();

void display();

int Position(char[]);

void kruskal\_algo();

};

/\* Initialzing adj matrix with 999 \*/

/\* 999 denotes infinite distance \*/

Graph::Graph()

{

no=0;

for(int i=0;i<10;i++)

for(int j=0;j<10;j++)

{

cost[i][j]=999;

}

}

/\* Taking inputs for creating graph \*/

void Graph::creat\_graph()

{

char ans,Start[10],End[10];

int wt,i,j;

cout<<"Enter the number of deparments in the college: ";

cin>>no;

cout<<"\nEnter the department name: ";

for(i=0;i<no;i++)

cin>>vertices[i];

do

{

cout<<"\nEnter two department names to add edge betwwen them: ";

cin>>Start>>End;

cout<<"Enter distance between them: ";

cin>>wt;

i=Position(Start);

j=Position(End);

cost[i][j]=cost[j][i]=wt;

cout<<"\nDo you want to add more edges (Y=YES/N=NO)? : "; /\* Type 'Y' or 'y' for YES and 'N' or 'n' for NO \*/

cin>>ans;

}while(ans=='y' || ans=='Y');

}

/\* Displaying Cost matrix \*/

void Graph::display()

{

int i,j;

cout<<"\n\nCost matrix: ";

for(i=0;i<no;i++)

{

cout<<"\n";

for(j=0;j<no;j++)

cout<<"\t"<<cost[i][j];

}

}

/\* Retrieving position of vertices in 'vertices' array \*/

int Graph::Position(char key[10])

{

int i;

for(i=0;i<10;i++)

if(strcmp(vertices[i],key)==0)

return i;

return -1;

}

void Graph::kruskal\_algo()

{

int i,j,v[10]={0},x,y,Total\_cost=0,min,gr=1,flag=0,temp,d;

while(flag==0)

{

min=999;

for(i=0;i<no;i++)

{

for(j=0;j<no;j++)

{

if(cost[i][j]<min)

{

min=cost[i][j];

x=i;

y=j;

}

}

}

if(v[x]==0 && v[y]==0)

{

v[x]=v[y]=gr;

gr++;

}

else if(v[x]!=0 && v[y]==0)

v[y]=v[x];

else if(v[x]==0 && v[y]!=0)

v[x]=v[y];

else

{

if(v[x]!=v[y])

{

d=v[x];

for(i=0;i<no;i++)

{

if(v[i]==d)

v[i]=v[y];

}//end for

}

}

cost[x][y]=cost[y][x]=999;

Total\_cost=Total\_cost+min; /\* calculating cost of minimum spanning tree \*/

cout<<"\n"<<vertices[x]<<"\t"<<vertices[y]<<"\t"<<min;

temp=v[0]; flag=1;

for(i=0;i<no;i++)

{

if(temp!=v[i])

{

flag=0;

break;

}

}

}

cout<<"\nTotal cost of the tree= "<<Total\_cost;

}

int main()

{

Graph g;

g.creat\_graph();

g.display();

cout<<"\n\n\nMinimum Spanning tree using Kruskal algo=>";

cout<<"\nSource Destination Distance\n";

g.kruskal\_algo();

return 0;

}

**OUTPUT :**