Prims

#include<stdio.h>

#include<conio.h>

int vis[10],vt[10],et[10][2],e=0,n;

float cost[10][10],sum=0;

void prims()

{

int x=1,min,i,j,m,k,u,v;

vt[x]=1;

vis[x]=1;

for(i=1;i<n;i++)

{

j=x;

min=999;

while(j>0)

{

k=vt[j];

for(m=2;m<=n;m++)

{

if(cost[k][m]<min && vis[m]==0)

{

min=cost[k][m];

u=k;

v=m;

}

}

j--;

}

vt[++x]=v;

et[i][1]=u;

et[i][2]=v;

e++;

vis[v]=1;

sum=sum+cost[u][v];

}

}

void main()

{

int i,j;

printf("Enter the number of vertices:");

scanf("%d",&n);

printf("\nEnter the cost matrix:\n");

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

scanf("%f",&cost[i][j]);

}

}

prims();

for(i=1;i<=e;i++)

{

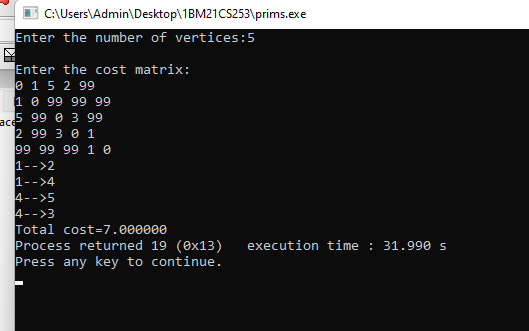
printf("%d-->%d\n",et[i][1],et[i][2]);

}

printf("Total cost=%f",sum);

}

OUTPUT



Kruskals

#include<stdio.h>

int cost[10][10],t[10][10],parent[10],n;

void kruskal()

{

int i,j,u,v;

int count=0;

int k=0;

int sum=0;

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

{

parent[i]=i;

}

while(count!=n-1)

{

int min=999;

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

{

for(j=0;j<n;j++)

{

if(cost[i][j]<min&&cost[i][j]!=0)

{

min=cost[i][j];

u=i;

v=j;

}

}

}

i=find(u);

j=find(v);

if(i!=j)

{

t[k][0]=u;

t[k][1]=v;

k++;

count++;

sum=sum+cost[u][v];

union\_ij(i,j);

}

cost[u][v]=cost[v][u]=999;

}

printf("Spanning Tree:\n");

for(i=0;i<n-1;i++)

printf("%d->%d\t",t[i][0],t[i][1]);

printf("\nTotal Cost=%d",sum);

getch();

}

void union\_ij(int i,int j)

{

if(i<j)

{

parent[j]=i;

}

else

{

parent[i]=j;

}

}

int find(int v)

{

while(parent[v]!=v)

{

v=parent[v];

}

return v;

}

int main()

{

int i,j;

printf("\nEnter the number of vertices:");

scanf("%d",&n);

printf("\Enter the cost matrix:");

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

{

for(j=0;j<n;j++)

{

scanf("%d",&cost[i][j]);

}

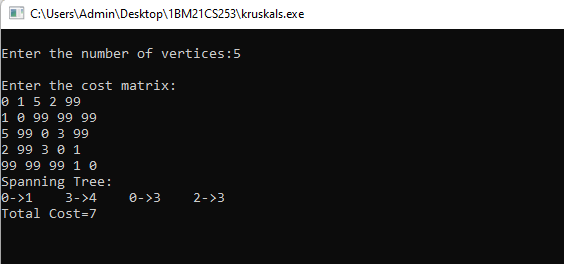
}

kruskal();

return 0;

}

OUTPUT



DIJKTRA’S

#include<stdio.h>

#include<conio.h>

#define INFINITY 9999

#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()

{

int G[MAX][MAX],i,j,n,u;

printf("Enter no. of vertices:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

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

for(j=0;j<n;j++)

scanf("%d",&G[i][j]);

printf("\nEnter the starting node:");

scanf("%d",&u);

dijkstra(G,n,u);

return 0;

}

void dijkstra(int G[MAX][MAX],int n,int startnode)

{

int cost[MAX][MAX],distance[MAX],pred[MAX];

int visited[MAX],count,mindistance,nextnode,i,j;

//pred[] stores the predecessor of each node

//count gives the number of nodes seen so far

//create the cost matrix

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

for(j=0;j<n;j++)

if(G[i][j]==0)

cost[i][j]=INFINITY;

else

cost[i][j]=G[i][j];

//initialize pred[],distance[] and visited[]

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

{

distance[i]=cost[startnode][i];

pred[i]=startnode;

visited[i]=0;

}

distance[startnode]=0;

visited[startnode]=1;

count=1;

while(count<n-1)

{

mindistance=INFINITY;

//nextnode gives the node at minimum distance

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

if(distance[i]<mindistance&&!visited[i])

{

mindistance=distance[i];

nextnode=i;

}

//check if a better path exists through nextnode

visited[nextnode]=1;

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

if(!visited[i])

if(mindistance+cost[nextnode][i]<distance[i])

{

distance[i]=mindistance+cost[nextnode][i];

pred[i]=nextnode;

}

count++;

}

//print the path and distance of each node

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

if(i!=startnode)

{

printf("\nDistance of node%d=%d",i,distance[i]);

printf("\nPath=%d",i);

j=i;

do

{

j=pred[j];

printf("<-%d",j);

}while(j!=startnode);

}

}

OUTPUT

