1. Minimum Spanning Tree **(22/Apr/2020)**
   1. Prims Algorithm
   2. Kruskal’s Algorithm
2. **Prim’s Algorithm**

Source Code :

#include<conio.h>

#include<iostream>

using namespace std;

int mat[1000][1000],tree[1000],n;

class spanning

{

public:

void readgraph()

{

cout<<"Enter no of vertices : ";

cin>>n;

cout<<"Enter Adjacency Matrix : ";

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

{

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

{

cin>>mat[i][j];

}

}

}

void printgraph()

{

cout<<"Adjacency Matrix : ";

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

{

cout<<"\n";

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

{

cout<<mat[i][j]<<" ";

}

}

}

void prims()

{

int total=0,v1,v2;

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

{

tree[1]=0;

tree[1]=1;

}

cout<<"\n v1 v2 min\_dist";

for(int k=2;k<=n;k++)

{

int min\_dist=3000;

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

{

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

{

if(mat[i][j]&&((tree[i]&&!tree[j])||(!tree[i]&&tree[j])))

{

if(mat[i][j]<min\_dist)

{

min\_dist=mat[i][j];

v1=i;

v2=j;

}

}

}

}

cout<<"\n"<<v1<<"-"<<v2<<"="<<min\_dist;

tree[v1]=1;

tree[v2]=1;

total=total+min\_dist;

}

cout<<"\n Cost of Minimum Spanning Tree is : "<<total;

}

};

int main()

{

spanning s;

s.readgraph();

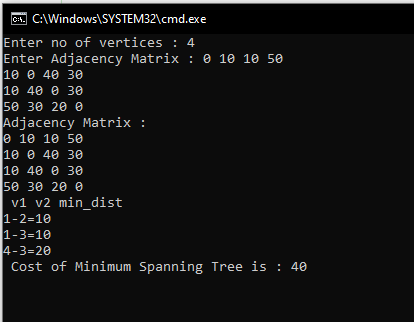
s.printgraph();

s.prims();

getch();

}

**OUTPUT**



1. **Kruskal’s Algorithm**

Source Code :

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

int mat[10][10],i,j,k,n,m,c,visit,visited[10],l,v,count,count1,vst,p;

int main()

{

int v1,v2;

cout<<"enter no of vertices : ";

cin >> n;

cout <<"enter no of edges : ";

cin >>m;

cout <<"Adjacency Matrix : ";

for(k=1;k<=m;k++)

{

cin >>i >>j >>c;

mat[i][j]=c;

mat[j][i]=c;

}

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

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

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

mat[i][j]=31999;

visit=1;

while(visit<n)

{

v=31999;

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

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

if(mat[i][j]!=31999 && mat[i][j]<v && mat[i][j]!=-1 )

{

int count =0;

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

{

if(visited[p]==i || visited[p]==j)

count++;

}

if(count >= 2)

{

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

if(mat[i][p]!=31999 && p!=j)

v1=p;

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

if(mat[j][p]!=31999 && p!=i)

v2=p;

if(mat[v1][v2]==-1)

continue;

}

l=i;

k=j;

v=mat[i][j];

}

cout <<"edge from " <<l <<"-->"<<k;

mat[l][k]=-1;

mat[k][l]=-1;

visit++;

int count=0;

count1=0;

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

{

if(visited[i]==l)

count++;

if(visited[i]==k)

count1++;

}

if(count==0)

visited[++vst]=l;

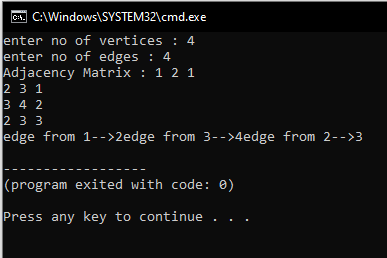
if(count1==0)

visited[++vst]=k;

}

}

**OUTPUT**



[**https://github.com/harinarayanank/Competitive-Lab**](https://github.com/harinarayanank/Competitive-Lab/tree/master/Sorting%20Algorithms%20Set%201)