

1) #include<stdio.h>

#define MAX 30

typedef struct edge

{

int u,v,w;

}edge;

typedef struct edgelist

{

edge data[MAX];

int n;

}edgelist;

edgelist elist;

int G[MAX][MAX],n;

edgelist spanlist;

void kruskal();

int find(int belongs[],int vertexno);

void union1(int belongs[],int c1,int c2);

void sort();

void print();

int main()

{

int i,j,total\_cost;

n=6;

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

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

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

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

/\*G={ { 0,73,22,0,21,0 },

{ 73,0,0,18,0,0 },

{ 22,0,0,11,44,44 },

{ 0,18,11,0,0,32 },

{ 21,0,44,0,0,45 },

{0,0,44,32,45,0} }; \*/

kruskal();

print();

return 0;

}

void kruskal()

{

int belongs[MAX],i,j,cno1,cno2;

elist.n=0;

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

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

{

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

{

elist.data[elist.n].u=i;

elist.data[elist.n].v=j;

elist.data[elist.n].w=G[i][j];

elist.n++;

}

}

sort();

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

belongs[i]=i;

spanlist.n=0;

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

{

cno1=find(belongs,elist.data[i].u);

cno2=find(belongs,elist.data[i].v);

if(cno1!=cno2)

{

spanlist.data[spanlist.n]=elist.data[i];

spanlist.n=spanlist.n+1;

union1(belongs,cno1,cno2);

}

}

}

int find(int belongs[],int vertexno)

{

return(belongs[vertexno]);

}

void union1(int belongs[],int c1,int c2)

{

int i;

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

if(belongs[i]==c2)

belongs[i]=c1;

}

void sort()

{

int i,j;

edge temp;

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

for(j=0;j<elist.n-1;j++)

if(elist.data[j].w>elist.data[j+1].w)

{

temp=elist.data[j];

elist.data[j]=elist.data[j+1];

elist.data[j+1]=temp;

}

}

void print()

{

int i,cost=0;

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

{

printf("\n%d\t%d\t%d",spanlist.data[i].u,spanlist.data[i].v,spanlist.data[i].w);

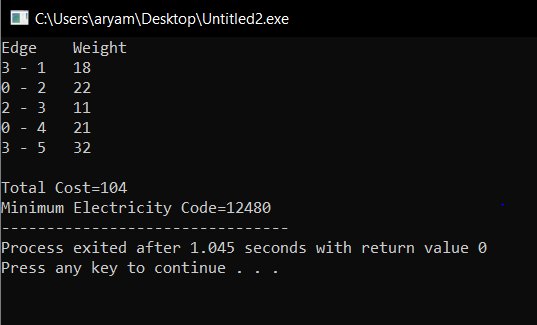
cost=cost+spanlist.data[i].w;

}

printf("\n\nCost of the spanning tree=%d",cost);

printf("\nMinimum Electricity Code=%d",cost\*120);

}

2) 

#include <limits.h>

#include <stdbool.h>

#include <stdio.h>

#define V 6

int minKey(int key[], bool mstSet[])

{

// Initialize min value

int min = INT\_MAX, min\_index;

for (int v = 0; v < V; v++)

if (mstSet[v] == false && key[v] < min)

min = key[v], min\_index = v;

return min\_index;

}

int printMST(int parent[], int graph[V][V])

{

int sum=0;

printf("Edge \tWeight\n");

for (int i = 1; i < V; i++){

printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);

sum=sum+graph[i][parent[i]];

}

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

printf("\nMinimum Electricity Code=%d",sum\*120);

}

void primMST(int graph[V][V])

{

int parent[V];

int key[V];

bool mstSet[V];

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

key[i] = INT\_MAX, mstSet[i] = false;

key[0] = 0;

parent[0] = -1;

for (int count = 0; count < V - 1; count++) {

int u = minKey(key, mstSet);

// Add the picked vertex to the MST Set

mstSet[u] = true;

for (int v = 0; v < V; v++)

if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v])

parent[v] = u, key[v] = graph[u][v];

}

printMST(parent, graph);

}

int main()

{

int graph[V][V] = { { 0,73,22,0,21,0 },

{ 73,0,0,18,0,0 },

{ 22,0,0,11,44,44 },

{ 0,18,11,0,0,32 },

{ 21,0,44,0,0,45 },

{0,0,44,32,45,0} };

// Print the solution

primMST(graph);

return 0;

}