1)

#include<stdio.h>

#include<stdlib.h>

#define MAX 20

#define CNT count++

int count=0;

int p[MAX],n,v,c=0;

struct EDGE

{

int src;

int dest;

int w;

}E[MAX],FINAL[MAX];

int comparator(const void \*a,const void \*b)

{

struct EDGE \*a1=(struct EDGE \*)a;

struct EDGE \*b1=(struct EDGE \*)b;

if(a1->w<b1->w)return -1;

if(a1->w>b1->w)return 1;

}

void edge\_input()

{

int i;

printf("\n\nENTER NUMBER OF VERTICES: ");

scanf("%d",&v);

printf("ENTER YOUR EDGES (FORMAT: Source,Destination,Weight)\n");

printf("ENTER -1,-1,-1 TO EXIT\n");

for(i=0;i<v\*v-1;i++)

{

printf("[%d]: ",i);

scanf("%d,%d,%d",&E[i].src,&E[i].dest,&E[i].w);

if(E[i].src==-1&&E[i].dest==-1&&E[i].w==-1)

break;

}

n=i;

}

void Union(int i,int j)

{ p[i]=j;CNT; }

int Find(int i)

{

while(p[i]>=0)

{CNT; i=p[i];CNT;}

return i;

}

int KruskalMST()

{

int i,j,k;

qsort(E,n,sizeof(struct EDGE),comparator);

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

{CNT; p[i]=-1;CNT; }

i=0;CNT;

int minCost=0;CNT;

while(i<=v-1)

{

CNT;

j=Find(E[i].src);CNT;

k=Find(E[i].dest);CNT;

if(j!=k)

{

CNT;

FINAL[c++]=E[i];CNT; CNT;

minCost+=E[i].w;CNT;

Union(j,k);

}

i++;CNT;

}

return(minCost);

}

void printEdge(struct EDGE \*x,int n1)

{

printf("\n\n^^^^^^^^^KRUSKAL'S MST^^^^^^^^^^^\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("SOURCE DESTINATION WEIGHT\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

int i;

for(i=0;i<n1;i++)

{

printf("%-15d",x[i].src);

printf("%-14d",x[i].dest);

printf("%-10d\n",x[i].w);

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

int main()

{

edge\_input();

KruskalMST();

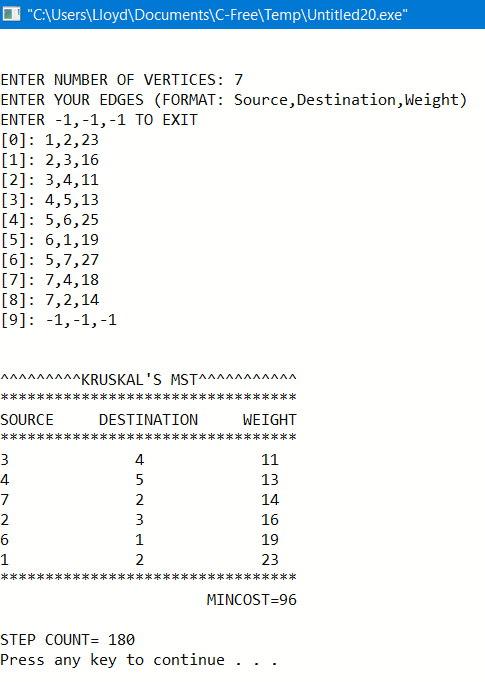
printEdge(FINAL, v-1);

printf(" MINCOST=%d\n",KruskalMST());

printf("\nSTEP COUNT= %d\n",count);

}

OUTPUT



2)

#include<stdio.h>

#include<limits.h>

#define MAX 10

#define G(m) m<0||m>=V

#define CNT c++

int c=0;

int GRAPH[MAX][MAX];

int V,E, minCost=0,count=0;

enum STATUS{VISITED,UNVISITED};

struct MSTEDGE

{

int src;

int dest;

int w;

}MST[MAX];

void printMST(struct MSTEDGE \*x,int n1)

{

printf("\n\n^^^^^^^^^^PRIM'S'S MST^^^^^^^^^^^\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("SOURCE DESTINATION WEIGHT\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

int i;

for(i=0;i<n1;i++)

{

printf("%-15d",x[i].src);

printf("%-14d",x[i].dest);

printf("%-10d\n",x[i].w);

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" MINCOST=%d",minCost);

}

void primMST()

{

int i,j;

int cost[MAX][MAX];

int dist[MAX],from[MAX];

enum STATUS node[MAX];

//\*\*\*\*\*\*\*\*\*\*\*INITIAL CONDITIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

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

{

CNT;CNT;

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

{ CNT;cost[i][j]=INT\_MAX;CNT;}

else

{ CNT;cost[i][j]=GRAPH[i][j];CNT;}

}

dist[0]=0;CNT;

node[0]=VISITED;CNT;

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

{

CNT;

dist[i]=cost[0][i];CNT;

from[i]=0;CNT;

node[i]=UNVISITED;CNT;

}

int u; //minimum edge to destination edge

int v; //from which source vertex

int x;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for(x=0;x<V-1;x++)

{

CNT;int min\_distance\_edge=INT\_MAX;CNT;

for(i=1;i<V;i++) //Searches the smallest edge of a particular vertex

{

CNT;

if(node[i]==UNVISITED && dist[i]<min\_distance\_edge )

{

CNT;min\_distance\_edge=dist[i];CNT;

v=i;CNT;

}

}

u=from[v];CNT; //It stores the source vertex info

MST[count].src=v;CNT;

MST[count].dest=u;CNT;

MST[count].w=cost[v][u];CNT;

count++;CNT;

node[v]=VISITED;CNT;

for(i=1;i<V;i++) //Distance array is updated

{

CNT;

if(node[i]==UNVISITED && cost[i][v]<dist[i])

{

CNT;

dist[i]=cost[i][v];CNT;

from[i]=v;CNT;

}

}

minCost+=cost[u][v];CNT;

}

}

void reset\_G()

{

int i,j;

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

{

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

GRAPH[i][j]=0;

}

}

void G\_create()

{

int i,x,y,w;

printf("ENTER NUMBER OF VERTICES:\n");

scanf("%d",&V);

reset\_G();

E=V\*(V-1);

printf("\n\nENTER EDGES(X,Y) AND IT WEIGHT (W) FOR YOUR GRAPH (FORMAT X,Y,W) (TO EXIT -1,-1,-1)\n");

do{

printf("-->");

scanf("%d,%d,%d",&x,&y,&w);

if(x==-1&&y==-1&&w==-1)

{break;}

else if(G(x)||G(y))

{printf("INVALID EDGE\n");}

else

{GRAPH[x][y]=w;GRAPH[y][x]=w;}

}while(x!=-1&&y!=-1);

}

int main()

{

G\_create();

primMST();

printMST(MST,count);

printf("\nSTEP COUNT= %d\n",c);

}

OUTPUT

