KRUSKAL ALGORITHM

#include<stdio.h>

#include<stdlib.h>

#define MAX 100

#define NIL -1

struct edge

{

int u;

int v;

int weight;

struct edge \*link;

}\*front = NULL;

void make\_tree(struct edge tree[]);

void insert\_pque(int i,int j,int wt);

struct edge \*del\_pque();

int isEmpty\_pque( );

void create\_graph();

int n;

int main()

{

int i;

struct edge tree[MAX];

int wt\_tree = 0;

create\_graph();

make\_tree(tree);

printf("\nEdges to be included in minimum spanning tree are :\n");

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

{

printf("\n%d->",tree[i].u);

printf("%d\n",tree[i].v);

wt\_tree += tree[i].weight;

}

printf("\nWeight of this minimum spanning tree is : %d\n", wt\_tree);

return 0;

}

void make\_tree(struct edge tree[])

{

struct edge \*tmp;

int v1,v2,root\_v1,root\_v2;

int father[MAX];

int i,count = 0;

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

father[i] = NIL;

while( !isEmpty\_pque( ) && count < n-1 )

{

tmp = del\_pque();

v1 = tmp->u;

v2 = tmp->v;

while( v1 !=NIL )

{

root\_v1 = v1;

v1 = father[v1];

}

while( v2 != NIL )

{

root\_v2 = v2;

v2 = father[v2];

}

if( root\_v1 != root\_v2 )

{

count++;

tree[count].u = tmp->u;

tree[count].v = tmp->v;

tree[count].weight = tmp->weight;

father[root\_v2]=root\_v1;

}

}

if(count < n-1)

{

printf("\nGraph is not connected, no spanning tree possible\n");

exit(1);

}

}

void insert\_pque(int i,int j,int wt)

{

struct edge \*tmp,\*q;

tmp = (struct edge \*)malloc(sizeof(struct edge));

tmp->u = i;

tmp->v = j;

tmp->weight = wt;

if( front == NULL || tmp->weight < front->weight )

{

tmp->link = front;

front = tmp;

}

else

{

q = front;

while( q->link != NULL && q->link->weight <= tmp->weight )

q = q->link;

tmp->link = q->link;

q->link = tmp;

if(q->link == NULL)

tmp->link = NULL;

}

}

struct edge \*del\_pque()

{

struct edge \*tmp;

tmp = front;

front = front->link;

return tmp;

}

int isEmpty\_pque( )

{

if ( front == NULL )

return 1;

else

return 0;

}

void create\_graph()

{

int i,wt,max\_edges,origin,destin;

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

scanf("%d",&n);

max\_edges = n\*(n-1)/2;

for(i=1; i<=max\_edges; i++)

{

printf("\nEnter edge %d(-1 -1 to quit): ",i);

scanf("%d %d",&origin,&destin);

if( (origin == -1) && (destin == -1) )

break;

printf("\nEnter weight for this edge : ");

scanf("%d",&wt);

if( origin >= n || destin >= n || origin<0 || destin<0)

{

printf("\nInvalid edge!\n");

i--;

}

else

insert\_pque(origin,destin,wt);

}

}

OUTPUT:

