

**19BIT0292**

**Bhaumik Tandan**

**DIGITAL ASSIGNMENT-5**

**DATA STRUCTURES**

**AND**

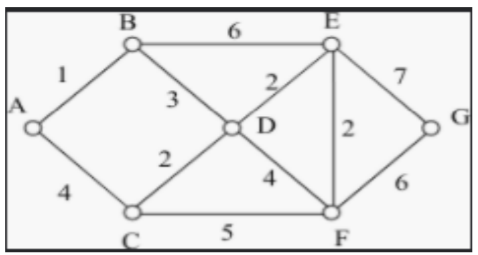
**ALGORITHMS**

**LABORATORY**

CSE2011

L57+L58

**Q1)** Illustrate minimum spanning tree using Kruskal’s algorithm for the following graph.



**CODE**

#include<stdio.h>

#include<stdlib.h>

int \*\*edges,n,e;

int \*parent,\*rank;

int find(int i)

{

if(parent[i]==i)

return i;

parent[i]=find(parent[i]);

return parent[i];

}

void Union(int x,int y)

{

int xroot=find(x);

int yroot=find(y);

if(xroot==yroot)

return;

if(rank[xroot]<rank[yroot])

parent[xroot]=yroot;

else if(rank[xroot]>rank[yroot])

parent[yroot]=xroot;

else

{

parent[yroot]=xroot;

rank[xroot]++;

}

}

int cmp(int \*\*a,int \*\*b)

{

return (\*a)[2]-(\*b)[2];

}

void kruskal()

{

printf("\nEdges in the minimum spanning tree are:-");

qsort(edges,e,sizeof(int\*),cmp);

int i,j,k=0;

parent=malloc(n\*sizeof(int));

rank=malloc(n\*sizeof(int));

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

{

parent[i]=i;

rank[i]=0;

}

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

{

int x=find(edges[i][0]);

int y=find(edges[i][1]);

if(x!=y)

{

printf("\n%c-%c %d",edges[i][0]+'A',edges[i][1]+'A',edges[i][2]);

Union(x,y);

k++;

}

if(k==n-1)

break;

}

if(k!=n-1)

printf("\nThe graph is not connected");

}

void take\_input()

{

int i,j;

printf("Enter number of vertices: ");

scanf("%d",&n);

printf("Enter number of edges: ");

scanf("%d",&e);

edges=(int\*\*)malloc(e\*sizeof(int\*));

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

{

edges[i]=(int\*)malloc(3\*sizeof(int));

char s,d,l;

int w;

printf("Enter weight, source and destination of edge %d: ",i+1);

scanf("%d",&w);

scanf("%c",&l);//to skip space

scanf("%c",&s);

scanf("%c",&l);//to skip space

scanf("%c",&d);

scanf("%c",&l);//to skip the newline character

edges[i][0]=s-'A';

edges[i][1]=d-'A';

edges[i][2]=w;

}

}

void default\_input\_test()

{

// 7

// 11

// 1 A B

// 4 A C

// 2 C D

// 3 B D

// 6 B E

// 5 C F

// 2 D E

// 4 D F

// 2 F E

// 7 G E

// 6 G F

n=7,e=11;

edges=(int\*\*)malloc(e\*sizeof(int\*));

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

edges[i]=(int\*)malloc(3\*sizeof(int));

edges[0][0]='A'-'A',edges[0][1]='B'-'A',edges[0][2]=1;

edges[1][0]='A'-'A',edges[1][1]='C'-'A',edges[1][2]=4;

edges[2][0]='C'-'A',edges[2][1]='D'-'A',edges[2][2]=2;

edges[3][0]='B'-'A',edges[3][1]='D'-'A',edges[3][2]=3;

edges[4][0]='B'-'A',edges[4][1]='E'-'A',edges[4][2]=6;

edges[5][0]='C'-'A',edges[5][1]='F'-'A',edges[5][2]=5;

edges[6][0]='D'-'A',edges[6][1]='E'-'A',edges[6][2]=2;

edges[7][0]='D'-'A',edges[7][1]='F'-'A',edges[7][2]=4;

edges[8][0]='F'-'A',edges[8][1]='E'-'A',edges[8][2]=2;

edges[9][0]='G'-'A',edges[9][1]='E'-'A',edges[9][2]=7;

edges[10][0]='G'-'A',edges[10][1]='F'-'A',edges[10][2]=6;

}

main()

{

// default\_input\_test();

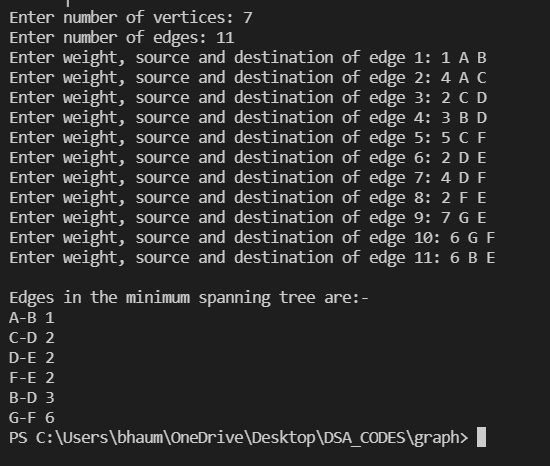
take\_input();

kruskal();

}

[**CLICK HERE FOR GITHUB LINK**](https://github.com/Bhaumik-Tandan/DSA_CODES/blob/master/graph/kruskal.c)

**OUTPUT**

****

**Q2)** Apply division hashing technique using the hash function H(k)=3k+2 mod h , h is the height of the hash table, and create one to one mapping between elements given {18, 17, 16, 91, 15, 18, 23, 27, 29, 21} with the index values of hash table having size of 10. Also remove the collision, using linear probing, if any.

**CODE**

#include<stdio.h>

#include<stdlib.h>

int \*arr,h;

int \*hash\_table;

int a,b;

void take\_input()

{

printf("(19BIT0292)Enter size of the hash: ");

scanf("%d",&h);

arr = (int \*)malloc(h\*sizeof(int));

hash\_table = (int \*)calloc(h,sizeof(int));

printf("Enter the elements of the array: ");

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

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

printf("Assuming hash function is a\*k+b Enter the value of a and b: ");

scanf("%d%d",&a,&b);

}

void fill\_hash\_table\_linear\_probing()

{

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

{

int index = (a\*arr[i]+b)%h;

while(hash\_table[index]!=0)

index = (index+1)%h;

hash\_table[index] = arr[i];

}

}

void print\_hash\_table()

{

for(int i=0;i<h;i++){

printf("%d->%d ",i,hash\_table[i]);

printf("\n");

}

}

main()

{

take\_input();

fill\_hash\_table\_linear\_probing();

print\_hash\_table();

}

[**CLICK HERE FOR GITHUB LINK**](https://github.com/Bhaumik-Tandan/DSA_CODES/blob/master/hashing/hashing_function.c)

**OUTPUT**

