DATA STRUCTURE LAB UNIVERSITY EXAM

2020-21

NAME : HELNA E M

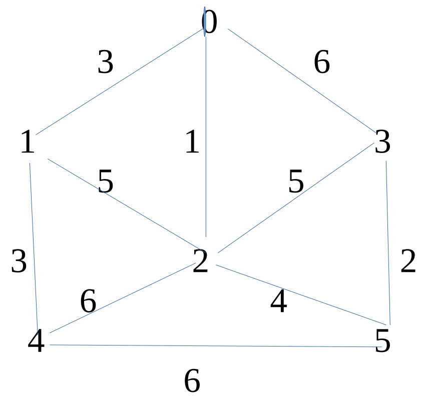
MCA S1

REGISTER NO : TKM20MCA-2020

20MCA220

QUESTION - 1

Develop a program to generate a minimum spanning tree using kruskal's algorithm for the given graph & compute the total cost.



ALGORITHM

Algorithm

Void kruskal()

Step 1 : Set elist.n=0

Step 2 : Repeat step 3 to step 6 until i<n

Step 3 : Repeat step 4 to step 5 until j<i

Step 4 : if Graph[i][j] != 0 go to ~~repeat~~ step 4.1 to step 4.4

Step 4.1 : elist.data[elist.n].u=i

Step 4.2 : elist.data[elist.n].v=j

Step 4.3 : elist.data[edist.n].w=Graph[i][j]

Step 4.4 : Increment elist.n ~~++~~ by one.

Step 5 : Increment j by 1

Step 6 : Increment i by 1

Step 7 : Call the function sort()

Step 8 : Set i=0 and repeat step 8.1 until i<n

Step 8.1 : Set belongs[i]=i and increment i by 1

Step 9 : Set spanlist.n=0

Step 8 : Set i=0 and repeat step 8.1 and 8.2 until i<elist.n

Step 8.1 : Call the function ~~function~~ find & return the val[ue]

2021/7/1 12:10

Step 8.2 : Call the function find() and return the value in Cno2. Increment i by 1

Step 9 : if Cno1 != Cno2. go to step 10

Step 10 : Set spanlist.data[spanlist.n] = elist.data[i];

Set spanlist.n = spanlist.n+1

Call the function Union()

find()

Step 1 : return the value belongs [vertexno]

Union()

Step 1 : Set i=0 and repeat step 2 & 3 until i<n

Step 2 : if belongs[i] == c2 go to step 2.1

step 2.1 : Set belongs [i] = c1

Step 3 : Increment i by 1.

Sort()

Step 1 : Set i=1 and repeat step 2 to step 5 until i<elist.n

Step 2 : Set j=0 and repeat step 3 to step 4 until j<elist.n-1

Step 3 : if elist.data[j].w > elist.data[j+1].w go to step 3.1

2021/7/1 12:10

Step 3.1 : temp = elist.data[j];
elist.data[j] = elist.data[j+1]
elist.data[j+1] = temp

Step 4 : Increment j by 1

Step 5 : Increment i by 1

display()

Step 1 : ~~Step~~ Set i = 0 and repeat step 2 & 3 until i < spanlist.n

Step 2 : cost = cost + spanlist.data[i].w

Step 3 : Increment i by 1

Main()

Step 1 : Set the edge weights

Step 2 : Call the function kruskal()

Step 3 : Call the function display()

2021/7/1 12:10

PROGRAM CODE

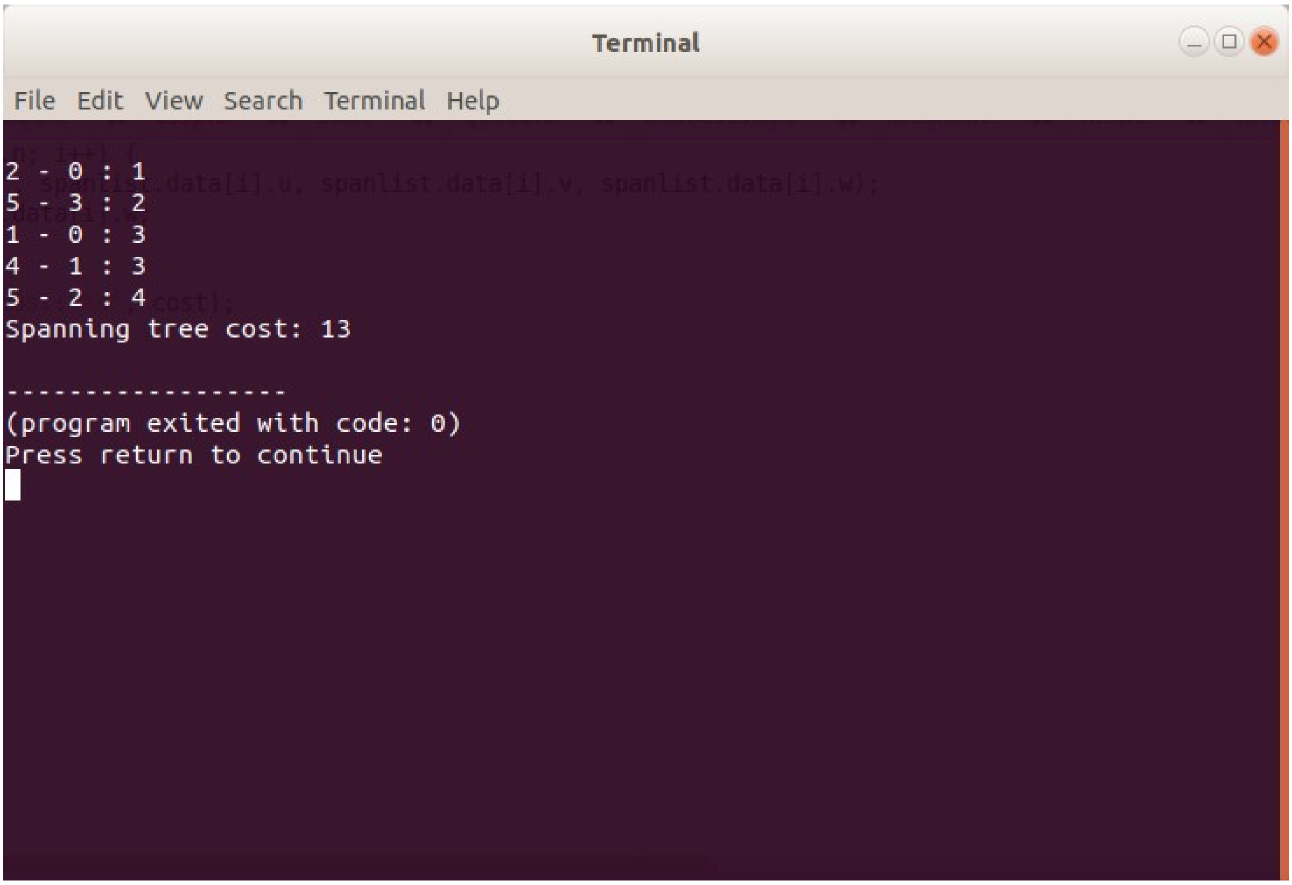
```
#include <stdio.h>

#define MAX 30

typedef struct edge {
  int u, v, w;
} edge;

typedef struct edge_list {
  edge data[MAX];
  int n;
} edge_list;

edge_list elist;

int Graph[MAX][MAX], n;
edge_list spanlist;

void kruskal();
int find(int belongs[], int vertexno);
void Union(int belongs[], int c1, int c2);
void sort();
void display();

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 (Graph[i][j] != 0) {
        elist.data[elist.n].u = i;
        elist.data[elist.n].v = j;
        elist.data[elist.n].w = Graph[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;
      Union(belongs, cno1, cno2);
    }
  }
}

int find(int belongs[], int vertexno) {
  return (belongs[vertexno]);
}

void Union(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 display() {
  int i, cost = 0;

  for (i = 0; i < spanlist.n; i++) {
    printf("\n%d - %d : %d", spanlist.data[i].u, spanlist.data[i].v, spanlist.data[i].w);
    cost = cost + spanlist.data[i].w;
  }

  printf("\nSpanning tree cost: %d", cost);
}

int main() {
  int i, j, total_cost;

  n = 6;

  Graph[0][0] = 0;
  Graph[0][1] = 3;
  Graph[0][2] = 1;
```

```
  Graph[0][3] = 6;
  Graph[0][4] = 0;
  Graph[0][5] = 0;

  Graph[1][0] = 3;
  Graph[1][1] = 0;
  Graph[1][2] = 5;
  Graph[1][3] = 0;
  Graph[1][4] = 3;
  Graph[1][5] = 0;

  Graph[2][0] = 1;
  Graph[2][1] = 5;
  Graph[2][2] = 0;
  Graph[2][3] = 5;
  Graph[2][4] = 6;    ;
  Graph[2][5] = 4;

  Graph[3][0] = 6;
  Graph[3][1] = 0;
  Graph[3][2] = 5;
  Graph[3][3] = 0;
  Graph[3][4] = 0;
  Graph[3][5] = 2;

  Graph[4][0] = 0;
  Graph[4][1] = 3;
  Graph[4][2] = 6;
  Graph[4][3] = 0;
  Graph[4][4] = 0;
  Graph[4][5] = 6;

  Graph[5][0] = 0;
  Graph[5][1] = 0;
  Graph[5][2] = 4;
  Graph[5][3] = 2;
  Graph[5][4] = 6;
  Graph[5][5] = 0;

  kruskal();
  display();
}
```

RESULT : The program was successfully executed and get the desired output.

OUTPUT


Terminal
File Edit View Search Terminal Help
2 - 0 : 1
5 - 3 : 2
1 - 0 : 3
4 - 1 : 3
5 - 2 : 4
Spanning tree cost: 13

(program exited with code: 0)
Press return to continue

PROGRAM - 2

QUESTION

Develop a program to implement DFS & BFS

ALGORITHM

Algorithm

Step 1 : Read the number of vertices as n.

Step 2 : Read the adjacency matrix of the graph

Step 3 : Set visited [i] = 0

Step 4 : Call the function DFS & pass argument 0

void DFS (int)

Step 1 : Set visited ~~of~~ [0] = 1

Step 2 : Set j = 0. Repeat step 3 ~~&~~ to 4 until j < n

Step 3 : if ! visited [j] && G [i] [j] == 1

Step 3.1 : ~~Set DFS j~~ Call the function DFS by passing j as argument.

Step 4 : Increment j by 1.

2021/7/1 12:09

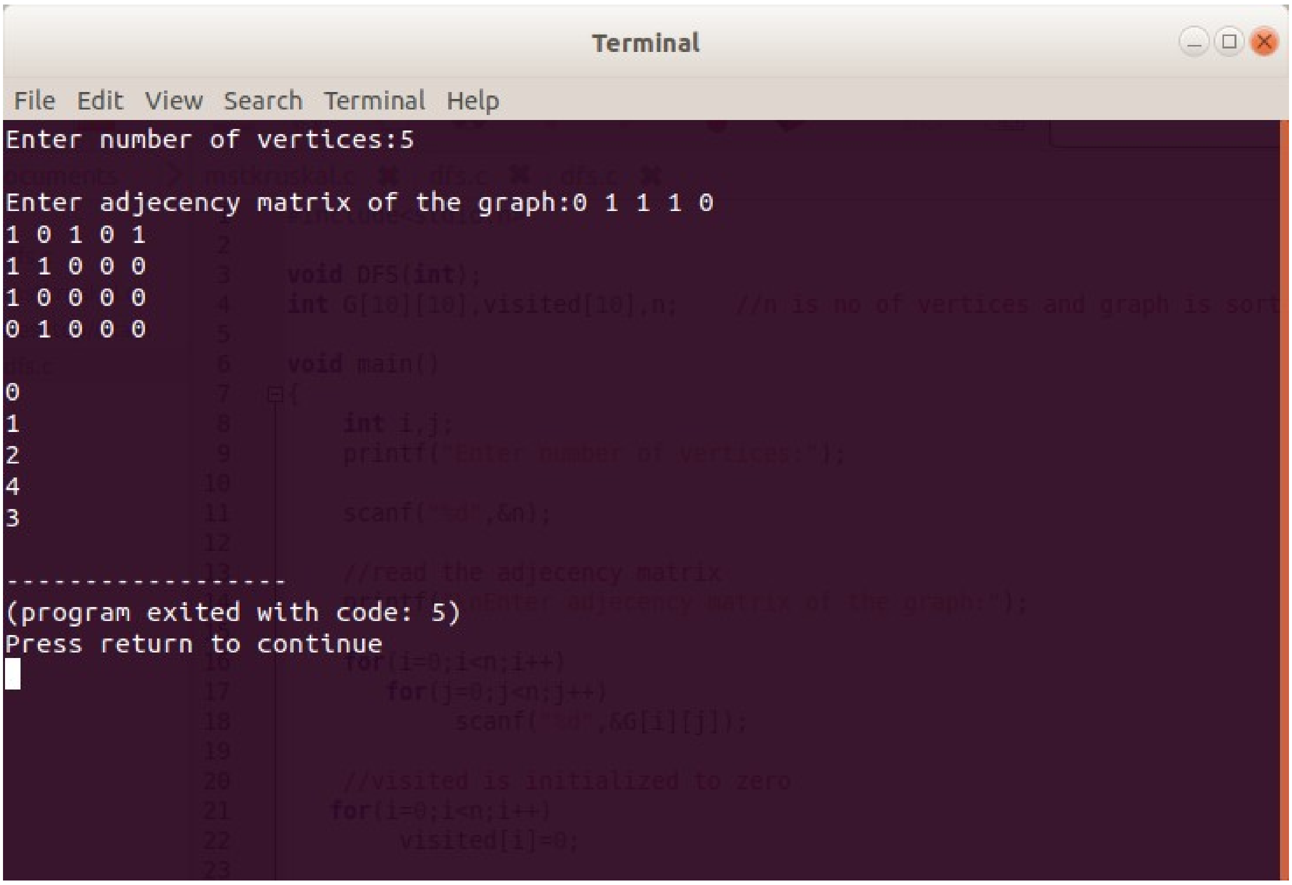
PROGRAM CODE

```
#include<stdio.h>

void DFS(int);
int G[10][10],visited[10],n;

void main()
{
    int i,j;
            printf("Enter number of vertices:");

            scanf("%d",&n);

            printf("\nEnter adjecency matrix of the graph:");

            for(i=0;i<n;i++)
       for(j=0;j<n;j++)
                                scanf("%d",&G[i][j]);

   for(i=0;i<n;i++)
        visited[i]=0;

    DFS(0);
}

void DFS(int i)
{
    int j;
            printf("\n%d",i);
    visited[i]=1;

            for(j=0;j<n;j++)
       if(!visited[j]&&G[i][j]==1)
            DFS(j);
}
```

RESULT

The program was successfully executed and get the desired output.

OUTPUT


Terminal
File Edit View Search Terminal Help
Enter number of vertices:5
Enter adjecency matrix of the graph:0 1 1 1 0
1 0 1 0 1
1 1 0 0 0
1 0 0 0 0
0 1 0 0 0
0
1
2
4
3

(program exited with code: 5)
Press return to continue