DATA STRUCTURE EXTERNAL LAB EXAMINATION

SUBMITTED BY

ABHISHEK M

MCA S1

ROLL NO:20MCA202

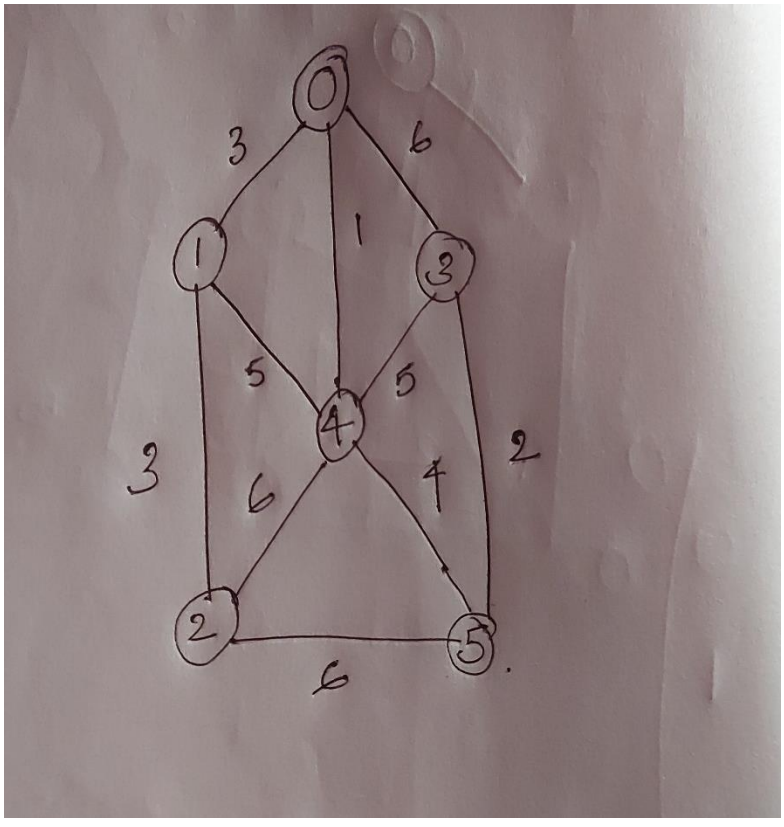
REG NO: TKM20MCA-2002

GITHUB LINK

https://github.com/ABHISHEKMABHI/ADVANCED-DS/tree/main/Lab%20Cycle

QUESTIONS

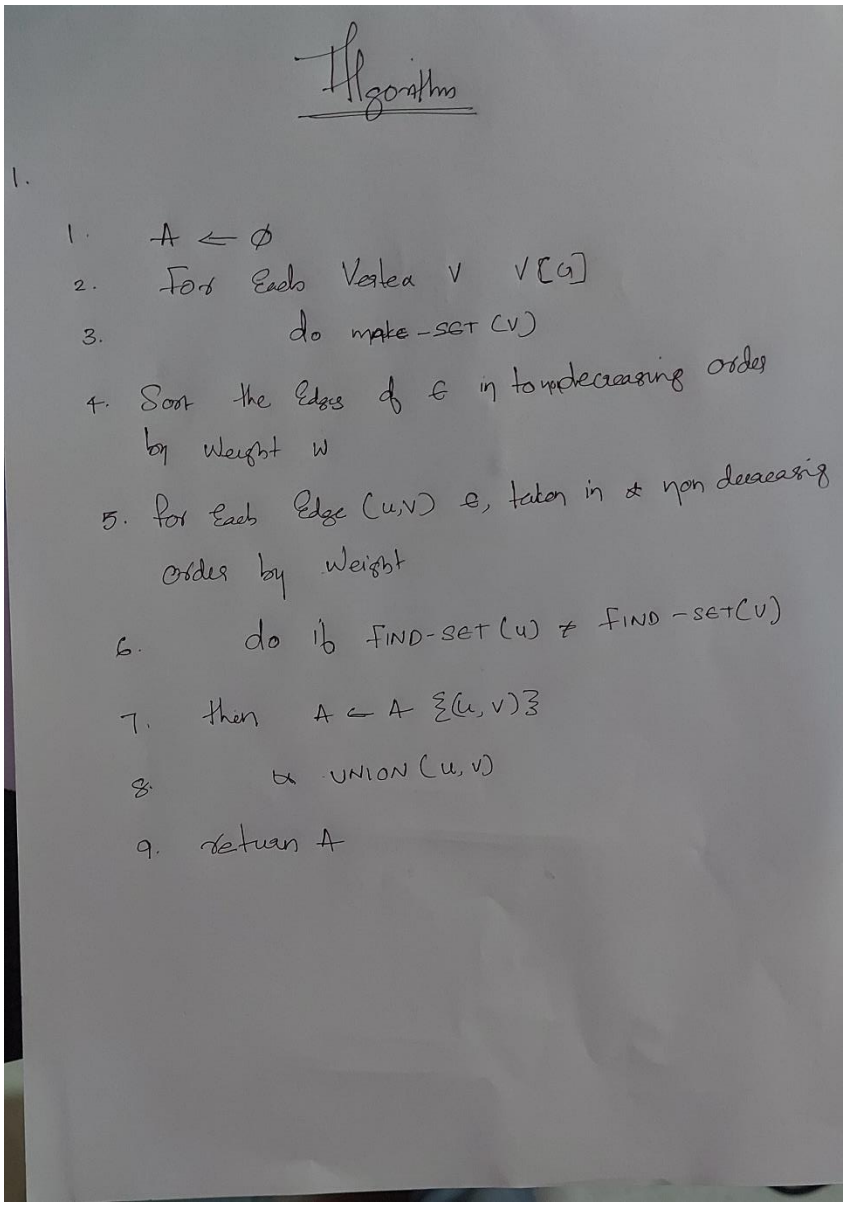
1.DEVELOP A PROGRAM TO GENERATE A MINIMUM SPANNING TREE USING KRUSKAL ALGORITHM FOR THE GIVEN GRAPH AND COMPUTE THE TOTAL COST



2.DEVELOP A PROGRAM TO IMPLEMENT BFS AND DFS

1.

ALGORITHM



Algorithm

1.

1. $A \leftarrow \emptyset$
2. For Each Vertex v $V[G]$
3. do make-SET (v)
4. Sort the Edges of E in to non decreasing order by weight w
5. for Each Edge (u,v) E, taken in a non decreasing order by weight
6. do if FIND-SET $(u) \neq$ FIND-SET (v)
7. then $A \leftarrow A \cup \{(u,v)\}$
8. & UNION (u,v)
9. return A

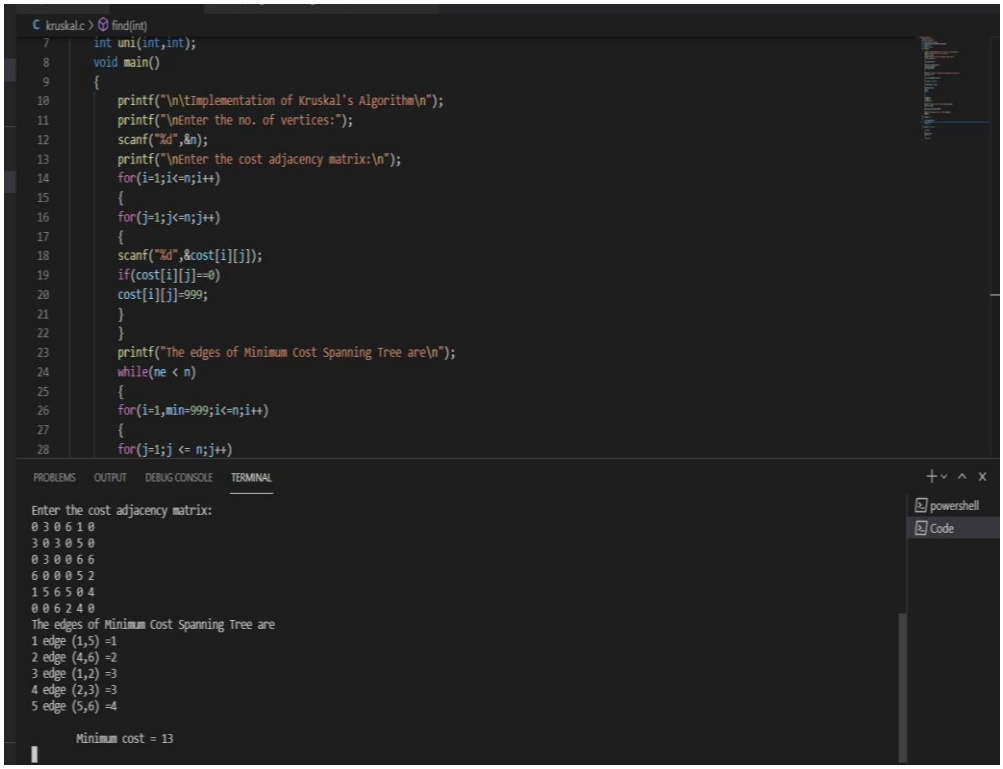
PROGRAM CODE

1.
```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int i,j,k,a,b,u,v,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
int find(int);
int uni(int,int);
void main()
{
        printf("\n\tImplementation of Kruskal's Algorithm\n");
        printf("\nEnter the no. of vertices:");
        scanf("%d",&n);
        printf("\nEnter the cost adjacency matrix:\n");
        for(i=1;i<=n;i++)
        {
        for(j=1;j<=n;j++)
        {
        scanf("%d",&cost[i][j]);
        if(cost[i][j]==0)
        cost[i][j]=999;
        }
        }
        printf("The edges of Minimum Cost Spanning Tree are\n");
        while(ne < n)
        {
```

```
        for(i=1,min=999;i<=n;i++)
        {
        for(j=1;j <= n;j++)
        {
        if(cost[i][j] < min)
        {
        min=cost[i][j];
        a=u=i;
        b=v=j;
        }
        }
        }
        u=find(u);
        v=find(v);
        if(uni(u,v))
        {
        printf("%d edge (%d,%d) =%d\n",ne++,a,b,min);
        mincost +=min;
        }
        cost[a][b]=cost[b][a]=999;
        }
        printf("\n\tMinimum cost = %d\n",mincost);
        getch();
}
int find(int i)
{
        while(parent[i])
        i=parent[i];
        return i;
}
int uni(int i,int j)
```

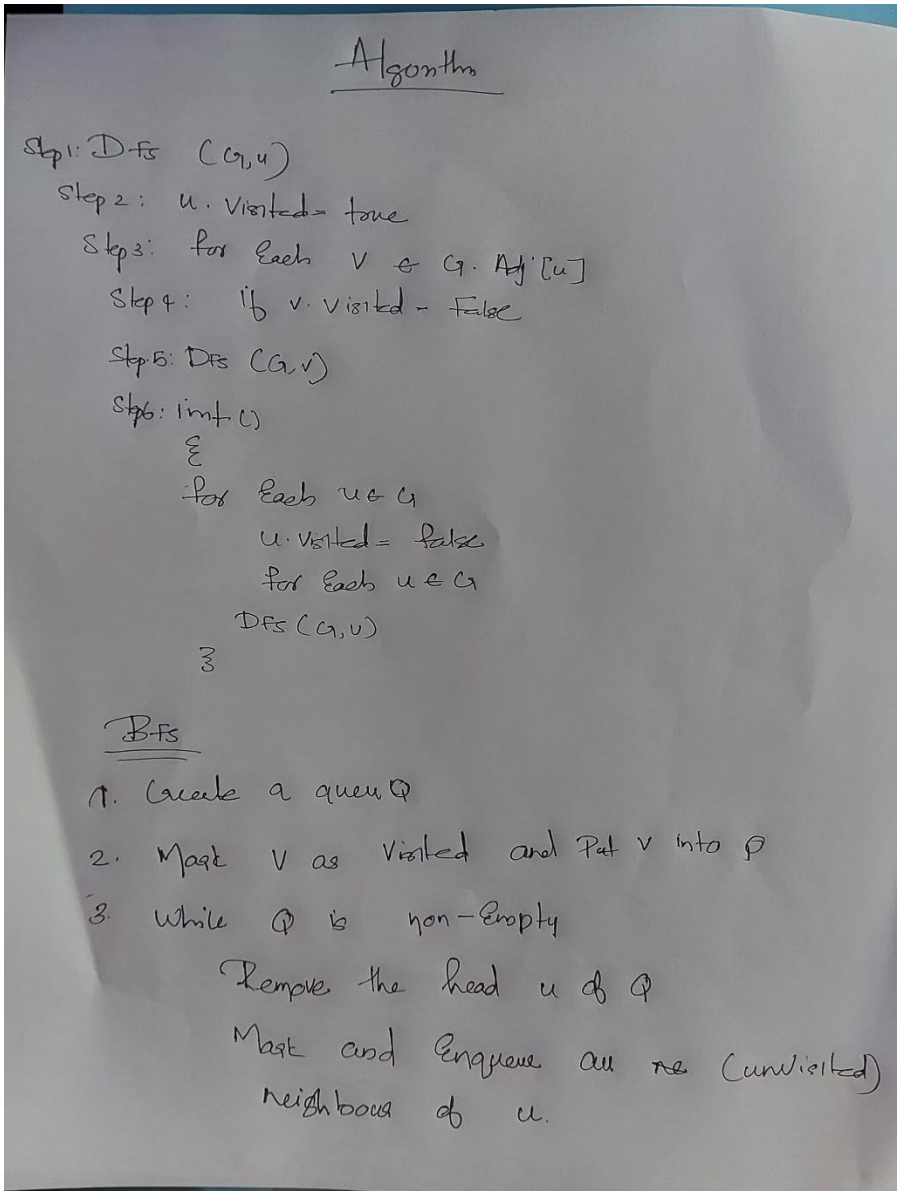
```
{
        if(i!=j)
        {
        parent[j]=i;
        return 1;
        }
        return 0;
}
```

OUTPUT



PROGRAM2

ALGORITM



Algorithm

Step 1: DFS (G, u)
Step 2: u.visited = true
Step 3: for each v ∈ G.Adj[u]
Step 4: if v.visited = false
Step 5: DFS (G, v)
Step 6: init()
{
for each u ∈ G
u.visited = false
for each u ∈ G
DFS (G, u)
}

BFS

1. Create a queue Q
2. Mark v as visited and put v into Q
3. While Q is non-empty
Remove the head u of Q
Mark and enqueue all (unvisited) neighbours of u.

PROGRAM CODE

```
#include<stdio.h>

int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];
int delete();
void add(int item);
void bfs(int s,int n);
void dfs(int s,int n);
void push(int item);
int pop();

void main()
{
int n,i,s,ch,j;
char c,dummy;
printf("ENTER THE NUMBER VERTICES ");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("ENTER 1 IF %d HAS A NODE WITH %d ELSE 0 ",i,j);
scanf("%d",&a[i][j]);
}
}
printf("THE ADJACENCY MATRIX IS\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
```

```
printf(" %d",a[i][j]);
}
printf("\n");
}

do
{
for(i=1;i<=n;i++)
vis[i]=0;
printf("\nMENU");
printf("\n1.B.F.S");
printf("\n2.D.F.S");
printf("\nENTER YOUR CHOICE");
scanf("%d",&ch);
printf("ENTER THE SOURCE VERTEX :");
scanf("%d",&s);

switch(ch)
{
case 1:bfs(s,n);
break;
case 2:
dfs(s,n);
break;
}
printf("DO U WANT TO CONTINUE(Y/N) ? ");
scanf("%c",&dummy);
scanf("%c",&c);
}while((c=='y')||(c=='Y'));
}


//**************BFS(breadth-first search) code**************//
```

```
void bfs(int s,int n)
{
int p,i;
add(s);
vis[s]=1;
p=delete();
if(p!=0)
printf(" %d",p);
while(p!=0)
{
for(i=1;i<=n;i++)
if((a[p][i]!=0)&&(vis[i]==0))
{
add(i);
vis[i]=1;
}
p=delete();
if(p!=0)
printf(" %d ",p);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
bfs(i,n);
}


void add(int item)
{
if(rear==19)
printf("QUEUE FULL");
else
{
if(rear==-1)
```

```
{
q[++rear]=item;
front++;
}
else
q[++rear]=item;
}
}
int delete()
{
int k;
if((front>rear)||(front==-1))
return(0);
else
{
k=q[front++];
return(k);
}
}


//***************DFS(depth-first search) code*****************//
void dfs(int s,int n)
{
int i,k;
push(s);
vis[s]=1;
k=pop();
if(k!=0)
printf(" %d ",k);
while(k!=0)
{
for(i=1;i<=n;i++)
```

```
if((a[k][i]!=0)&&(vis[i]==0))
{
push(i);
vis[i]=1;
}
k=pop();
if(k!=0)
printf(" %d ",k);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
dfs(i,n);
}
void push(int item)
{
if(top==19)
printf("Stack overflow ");
else
stack[++top]=item;
}
int pop()
{
int k;
if(top==-1)
return(0);
else
{
k=stack[top--];
return(k);
}
}
```

OUTPUT

```
ENTER THE NUMBER VERTICES 4
ENTER 1 IF 1 HAS A NODE WITH 1 ELSE 0 0
ENTER 1 IF 1 HAS A NODE WITH 2 ELSE 0 1
ENTER 1 IF 1 HAS A NODE WITH 3 ELSE 0 1
ENTER 1 IF 1 HAS A NODE WITH 4 ELSE 0 0
ENTER 1 IF 2 HAS A NODE WITH 1 ELSE 0 1
ENTER 1 IF 2 HAS A NODE WITH 2 ELSE 0 0
ENTER 1 IF 2 HAS A NODE WITH 3 ELSE 0 0
ENTER 1 IF 2 HAS A NODE WITH 4 ELSE 0 0
ENTER 1 IF 3 HAS A NODE WITH 1 ELSE 0 1
ENTER 1 IF 3 HAS A NODE WITH 2 ELSE 0 0
ENTER 1 IF 3 HAS A NODE WITH 3 ELSE 0 0
ENTER 1 IF 3 HAS A NODE WITH 4 ELSE 0 1
ENTER 1 IF 4 HAS A NODE WITH 1 ELSE 0 0
ENTER 1 IF 4 HAS A NODE WITH 2 ELSE 0 0
ENTER 1 IF 4 HAS A NODE WITH 3 ELSE 0 1
ENTER 1 IF 4 HAS A NODE WITH 4 ELSE 0 0
THE ADJACENCY MATRIX IS
 0 1 1 0
 1 0 0 0
 1 0 0 1
 0 0 1 0

MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE1
ENTER THE SOURCE VERTEX :1
 1 2  3  4 DO U WANT TO CONTINUE(Y/N) ? y

MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE2
ENTER THE SOURCE VERTEX :1
 1  3  4  2 DO U WANT TO CONTINUE(Y/N) ? n
```