PRAJITHA P

TKM20MCA-2028

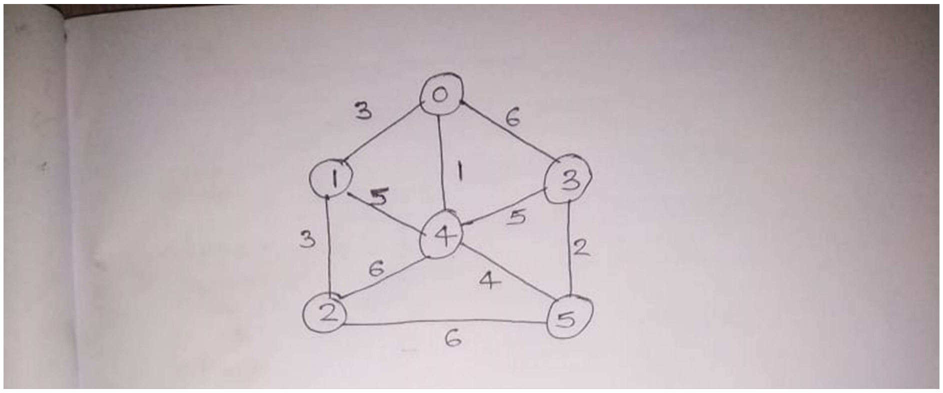
20MCA2028

https://github.com/PRAJITHA-P/DataStuctures

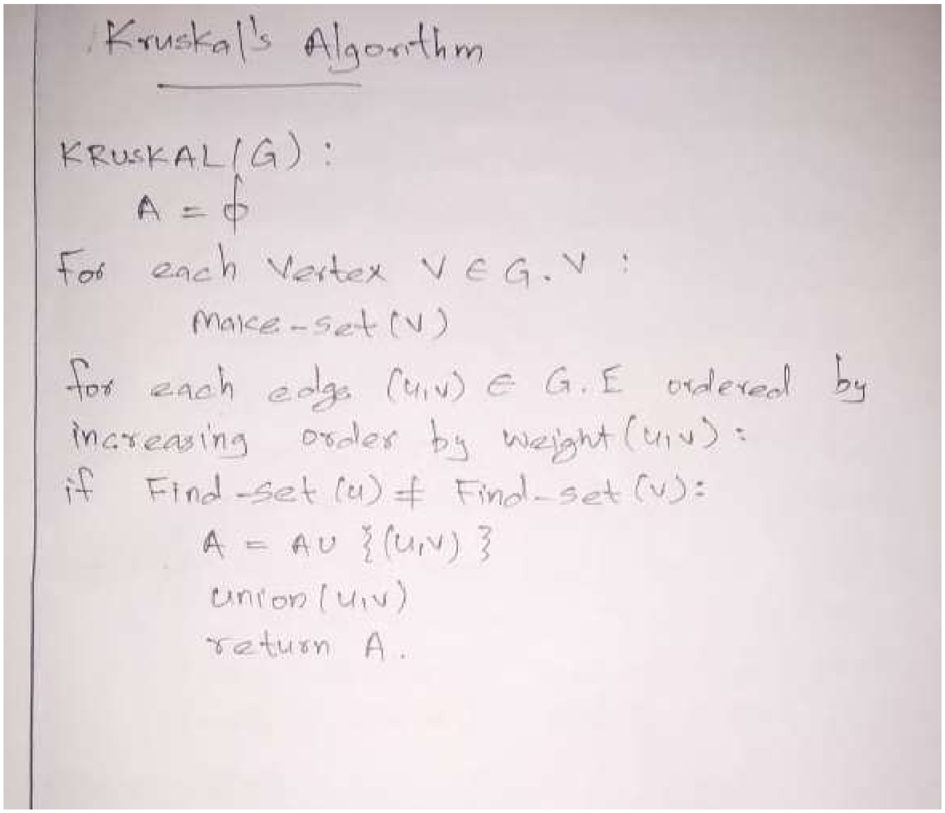
Part A

Question number: 2

Develop program to minimum spanning tree using Kruskal algorithm for the given graph and complete the total cost



Algorithm



Kruskal's Algorithm

```
KRUSKAL(G) :
    $A = \phi$
For each Vertex $v \in G.V$ :
        Make-set(v)
for each edge $(u,v) \in G.E$ ordered by
increasing order by weight(u,v) :
if Find-set(u) $\neq$ Find-set(v):
        $A = A \cup \{(u,v)\}$
        union(u,v)
        return A.
```

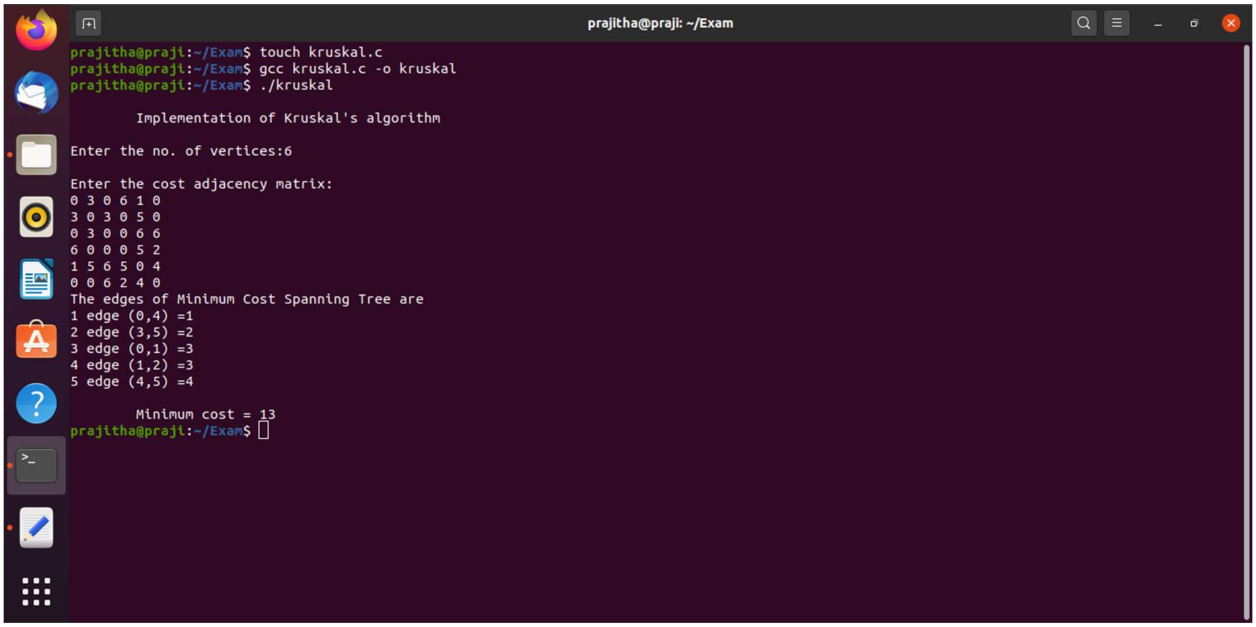
Program

```
#include<stdio.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=0;i<n;i++)
{
    for(j=0;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=0,min=999;i<n;i++)
    {
        for(j=0;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);


}
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

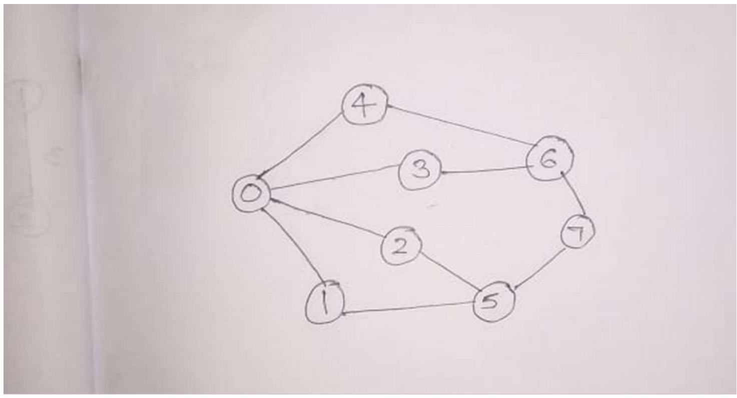


```
prajitha@praji:~/Exam$ touch kruskal.c
prajitha@praji:~/Exam$ gcc kruskal.c -o kruskal
prajitha@praji:~/Exam$ ./kruskal

        Implementation of Kruskal's algorithm

Enter the no. of vertices:6

Enter the cost adjacency matrix:
0 3 0 6 1 0
3 0 3 0 5 0
0 3 0 0 6 6
6 0 0 0 5 2
1 5 6 5 0 4
0 0 6 2 4 0
The edges of Minimum Cost Spanning Tree are
1 edge (0,4) =1
2 edge (3,5) =2
3 edge (0,1) =3
4 edge (1,2) =3
5 edge (4,5) =4

        Minimum cost = 13
prajitha@praji:~/Exam$ 
```

Part B

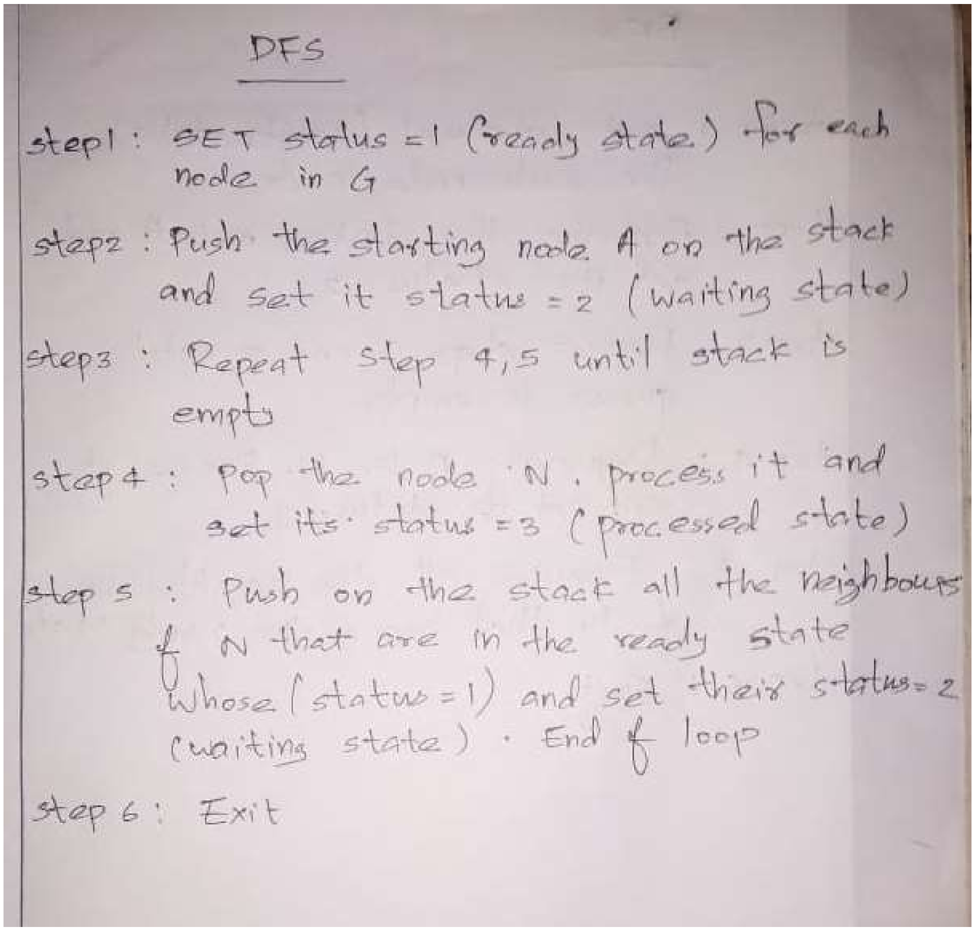
Question number: 2

Develop a program to implement DFS and BFS



Algorithm

DFS



DFS

step1 : SET status =1 (ready state) for each node in G

step2 : Push the starting node A on the stack and set it status = 2 (waiting state)

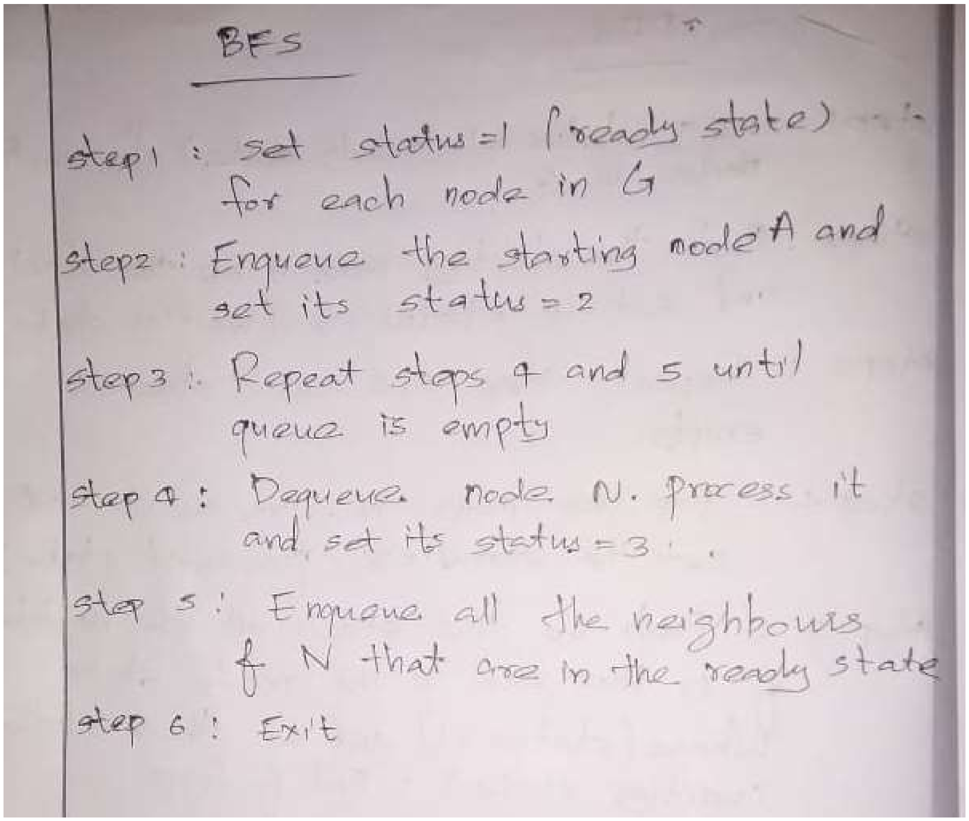
step3 : Repeat step 4,5 until stack is empty

step4 : Pop the node N . process it and set its status =3 (processed state)

step 5 : Push on the stack all the neighbours of N that are in the ready state whose (status =1) and set their status= 2 (waiting state) . End of loop

step 6 : Exit

BSF



BFS

step1 : set status =1 (ready state)
for each node in G

step2 : Enqueue the starting node A and
set its status = 2

step3 : Repeat steps 4 and 5 until
queue is empty

step 4 : Dequeue node N. process it
and set its status = 3 .

step 5 : Enqueue all the neighbours
f N that are in the ready state

step 6 : Exit

Program

DFS

```
#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);
}
```

BFS

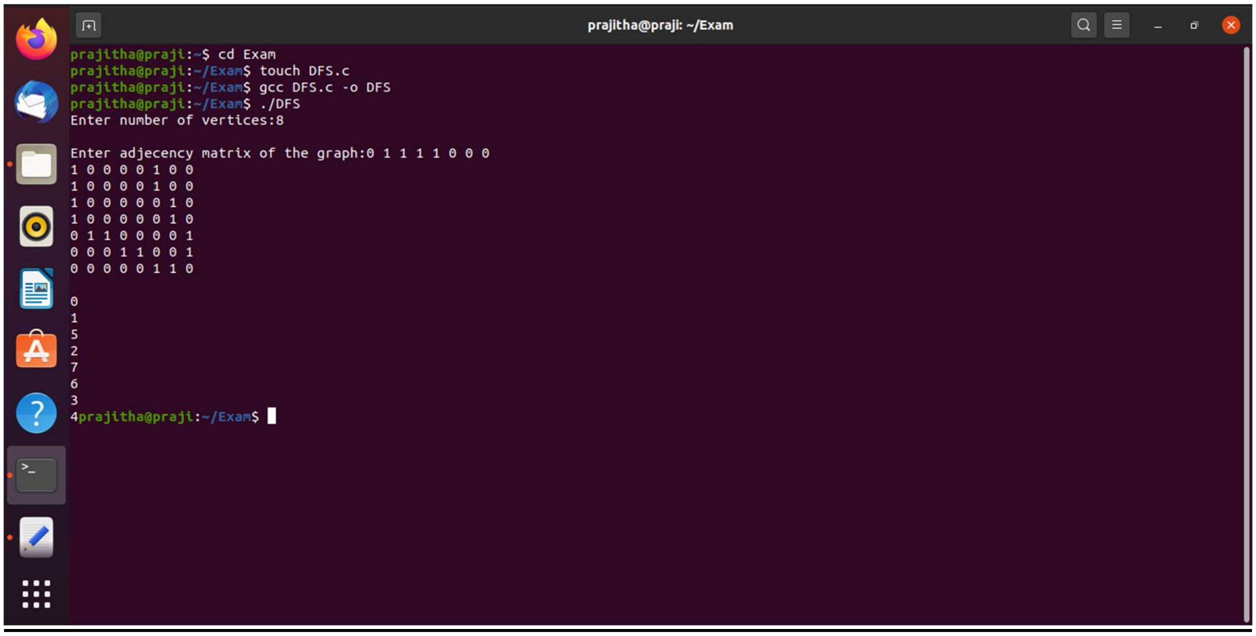
```
#include<stdio.h>
int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;
void bfs(int v);
int main(){
         int v;
     printf("Enter the number of vertices:");
     scanf("%d",&n);
         printf("Enter the adjacency matrix:");
         for(i=0;i<n;i++){
            for(j=0;j<n;j++)
```

```
                scanf("%d",&a[i][j]);


    }
        printf("Enter the starting vertex:");
        scanf("%d",&v);
        for(i=0;i<n;i++){
             q[i]=0;
            visited[i]=0;
    }
        bfs(v);
        printf("The reachable nodes are:");
        for(i=0;i<n;i++){
          if(visited[i])
                  printf("%d\t",i);


    }
return 0;
}
void bfs(int v){
        for(int i=0;i<n;i++){
            if(a[v][i] && !visited[i])
                        q[++r]=i;
        }
        if(f<=r){
                visited[q[f]]=1;
                bfs(q[++f]);
    }
```

```
}
```

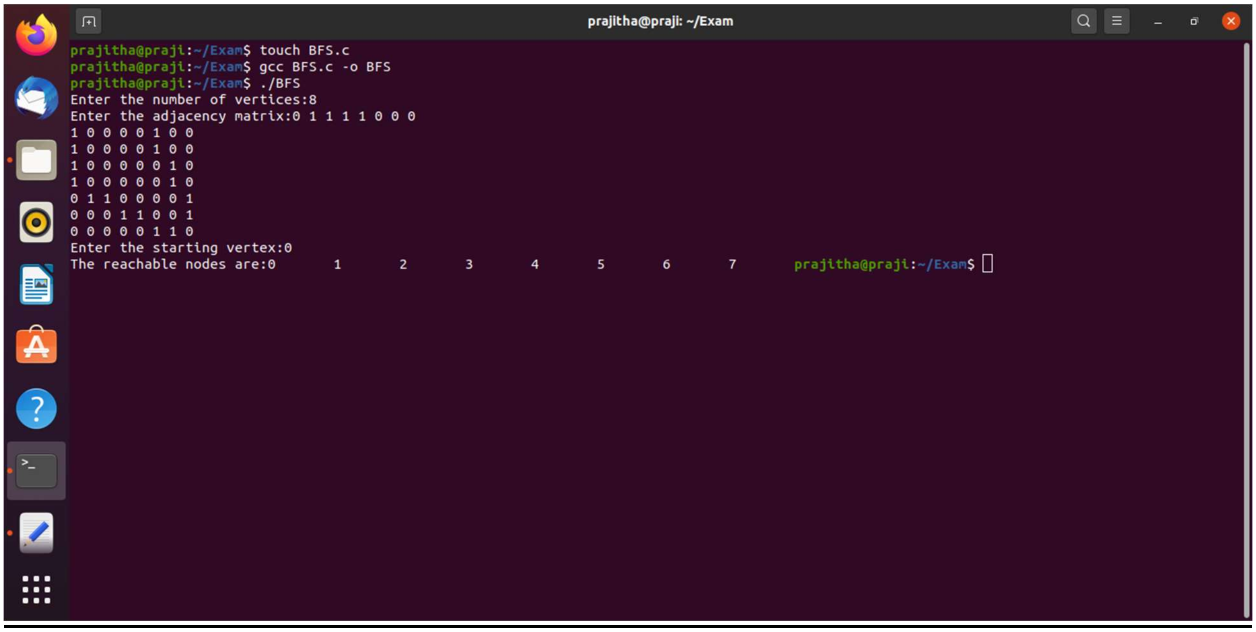
Output

DFS



```
prajitha@praji:~$ cd Exam
prajitha@praji:~/Exam$ touch DFS.c
prajitha@praji:~/Exam$ gcc DFS.c -o DFS
prajitha@praji:~/Exam$ ./DFS
Enter number of vertices:8

Enter adjecency matrix of the graph:0 1 1 1 1 0 0 0
1 0 0 0 0 1 0 0
1 0 0 0 0 1 0 0
1 0 0 0 0 0 1 0
1 0 0 0 0 0 1 0
0 1 1 0 0 0 0 1
0 0 0 1 1 0 0 1
0 0 0 0 0 1 1 0

0
1
5
2
7
6
3
4prajitha@praji:~/Exam$ 
```

BFS



```
prajitha@praji:~/Exam$ touch BFS.c
prajitha@praji:~/Exam$ gcc BFS.c -o BFS
prajitha@praji:~/Exam$ ./BFS
Enter the number of vertices:8
Enter the adjacency matrix:0 1 1 1 1 0 0 0
1 0 0 0 0 1 0 0
1 0 0 0 0 1 0 0
1 0 0 0 0 0 1 0
1 0 0 0 0 0 1 0
0 1 1 0 0 0 0 1
0 0 0 1 1 0 0 1
0 0 0 0 0 1 1 0
Enter the starting vertex:0
The reachable nodes are:0       1       2       3       4       5       6       7       prajitha@praji:~/Exam$ 
```