**LAB EXAMINATION-PRACTICAL**

**DATA STRUCTURES LAB**

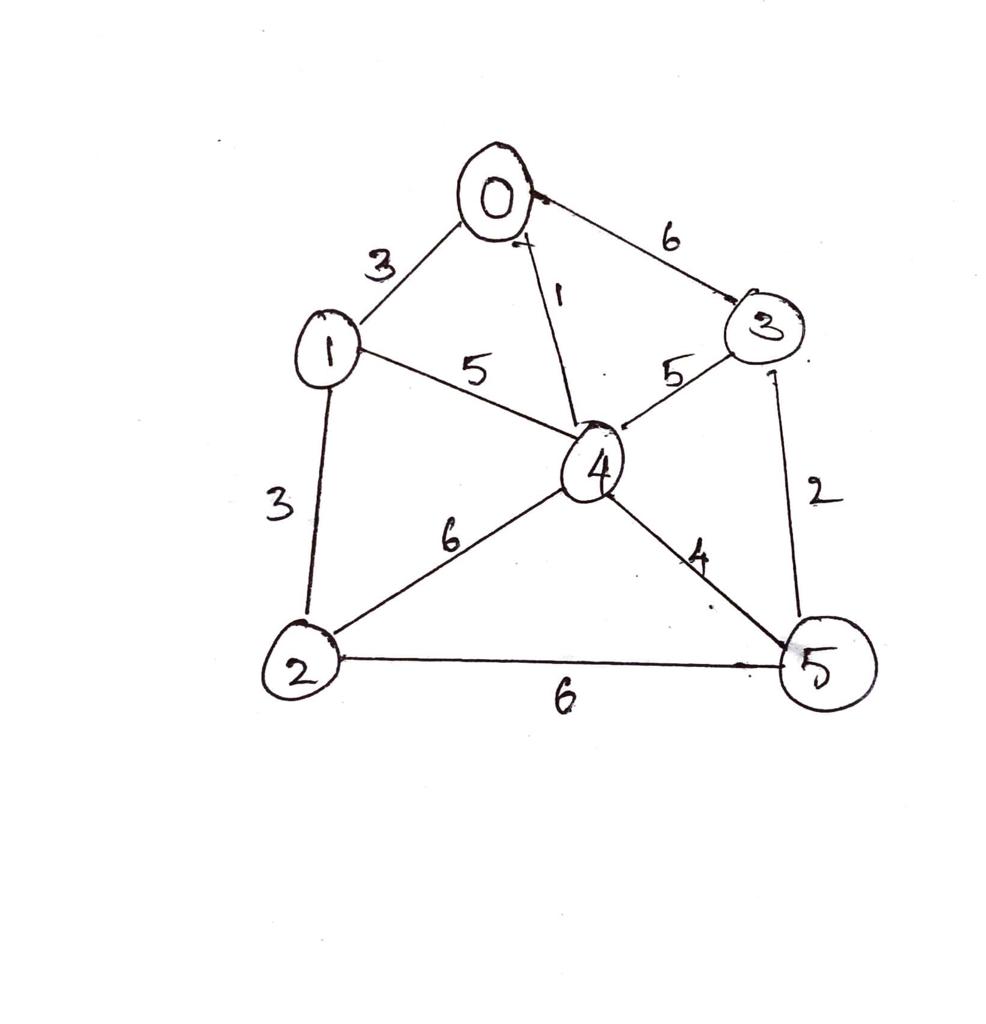
**SHAD IBRAHIM N P**

**MCA-2034**

**GIT REPO LINK:**

**QUESTION 1**

Develop a program to generate a minimum spanning tree using kruskals algorithm for the given graph and compute the total cost.



**CODE**

#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;

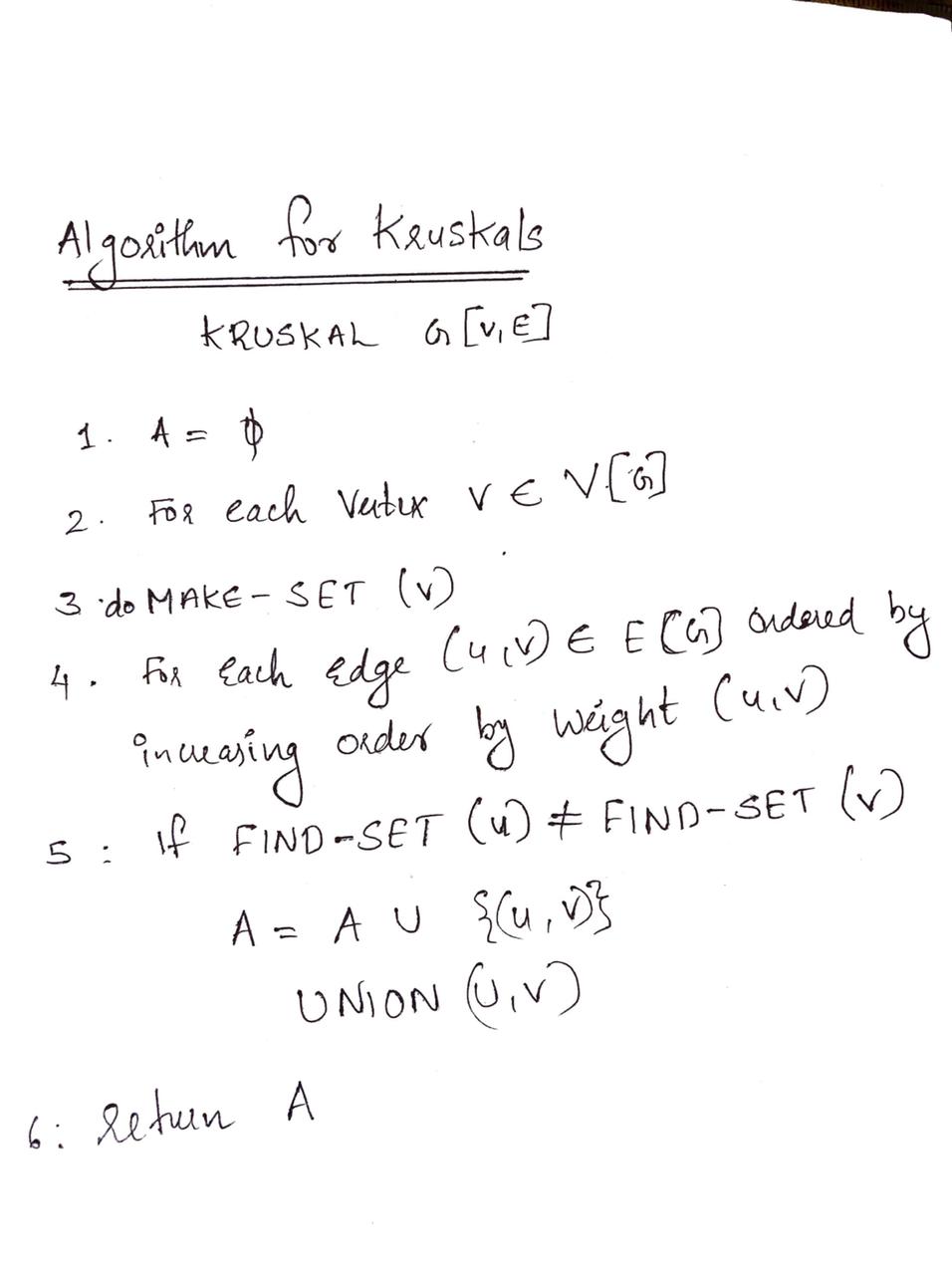
        return 1;

    }

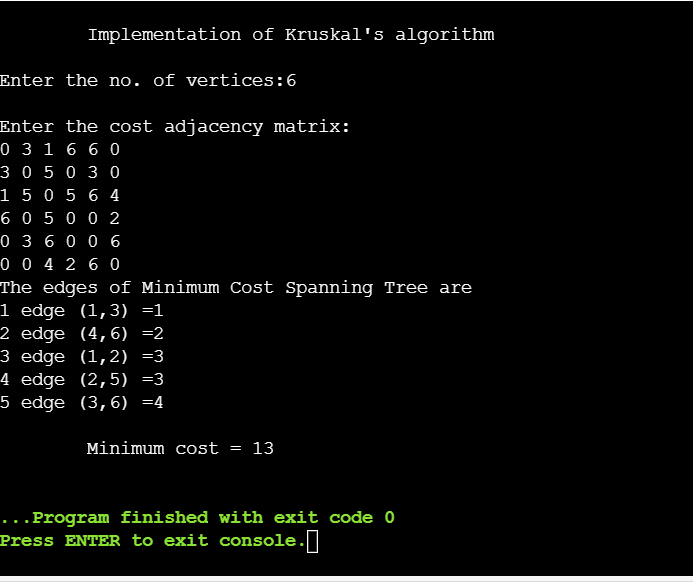
    return 0;

}

**ALGORITHM:**



**OUTPUT:**



**QUESTION 2**

Write a program to implement DFS and BFS.

**CODE FOR DFS**

#include<stdio.h>

int n,g[100][100],v[100],s;

void dfs(int s)

{

    int i;

    v[s]=1;

    printf("Visited Node = %d\n",s);

    for(i=0;i<n;i++){

        if(v[i]==0&&g[s][i]==1)

        {

            dfs(i);

        }

    }

}

int main()

{

    int i,j;

    printf("Enter the no of vertices\n");

    scanf("%d",&n);

    printf("Enter the adjacency Matrix : \n");

    for(i=0;i<n;i++){

        for(j=0;j<n;j++){

            printf("%d=",j);

            scanf("%d",&g[i][j]);

        }

    }

    for(i=0;i<n;i++)

    {

        v[i]=0;

    }

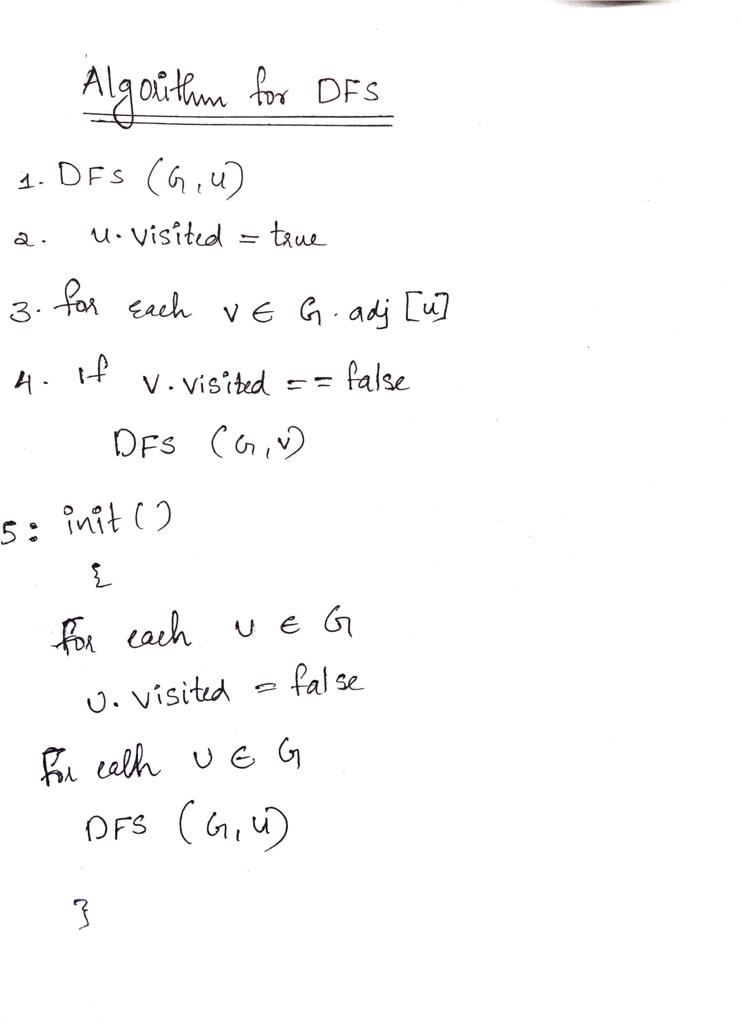
    printf("Enter the starting vertex");

    scanf("%d",&s);

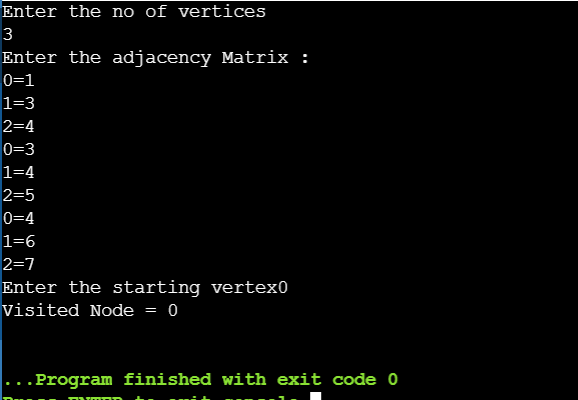
    dfs(s);

}

**ALGORITHM FOR DFS**



**OUTPUT FOR DFS**



**CODE FOR BFS**

#include<stdio.h>

int n,g[20][20],v[20],a[20],f=-1,r=-1,s;

void bfs(int s)

    {

     int i=0;

     for(i=0;i<n;i++)

     {

         if(v[i]==0 && g[s][i]==1){

             r=r+1;

             a[r]=i;

             v[i]=1;

             printf("%d",i);

     }

    }

    f=f+1;

    if(f<=r){

        bfs(a[f]);

    }

}

int main(){

    int i,j;

    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",&g[i][j]);

    for(i=0;i<n;i++)

    v[i]=0;

    printf("Starting Vertex");

    scanf("%d",&s);

    f=r=0;

    a[r]=s;

    v[s]=1;

    printf("BFS traversal is %d",s);

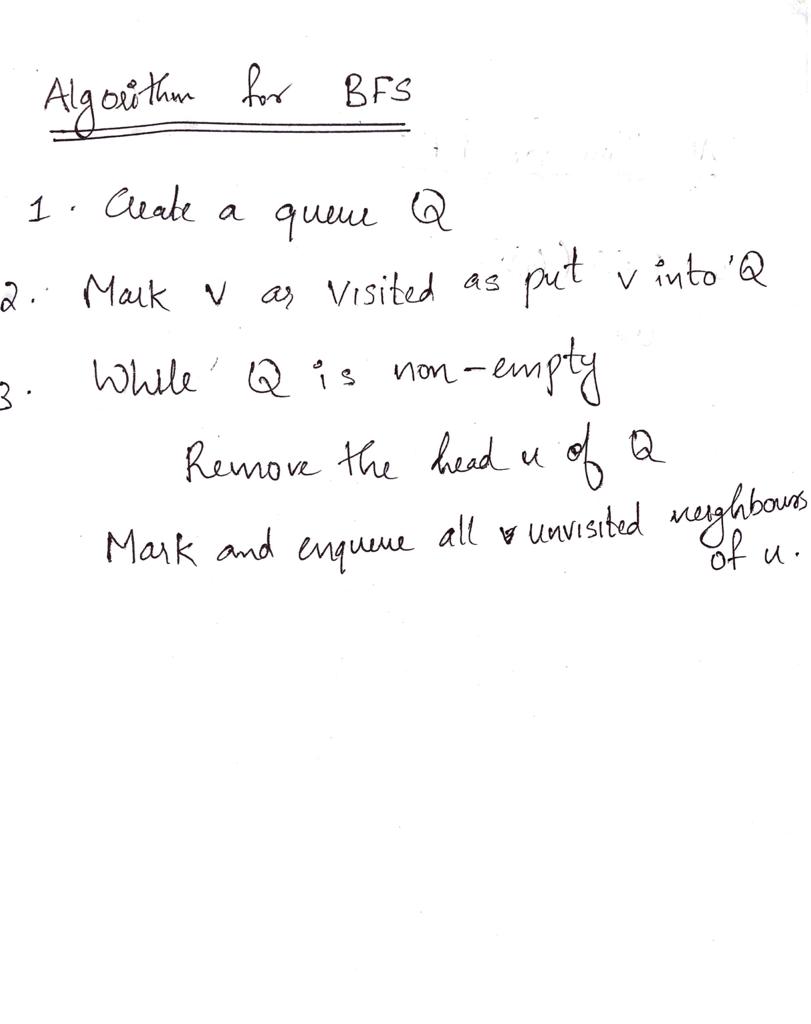
    bfs(s);

    if(r!=n-1)

    printf("BFS is not possible");

}

**ALGORITHM FOR BFS**



**OUTPUT FOR BFS**

