# FIRST SEMESTER EXTERNAL LAB EXAMINATION

# DATA STRUCTURES LAB

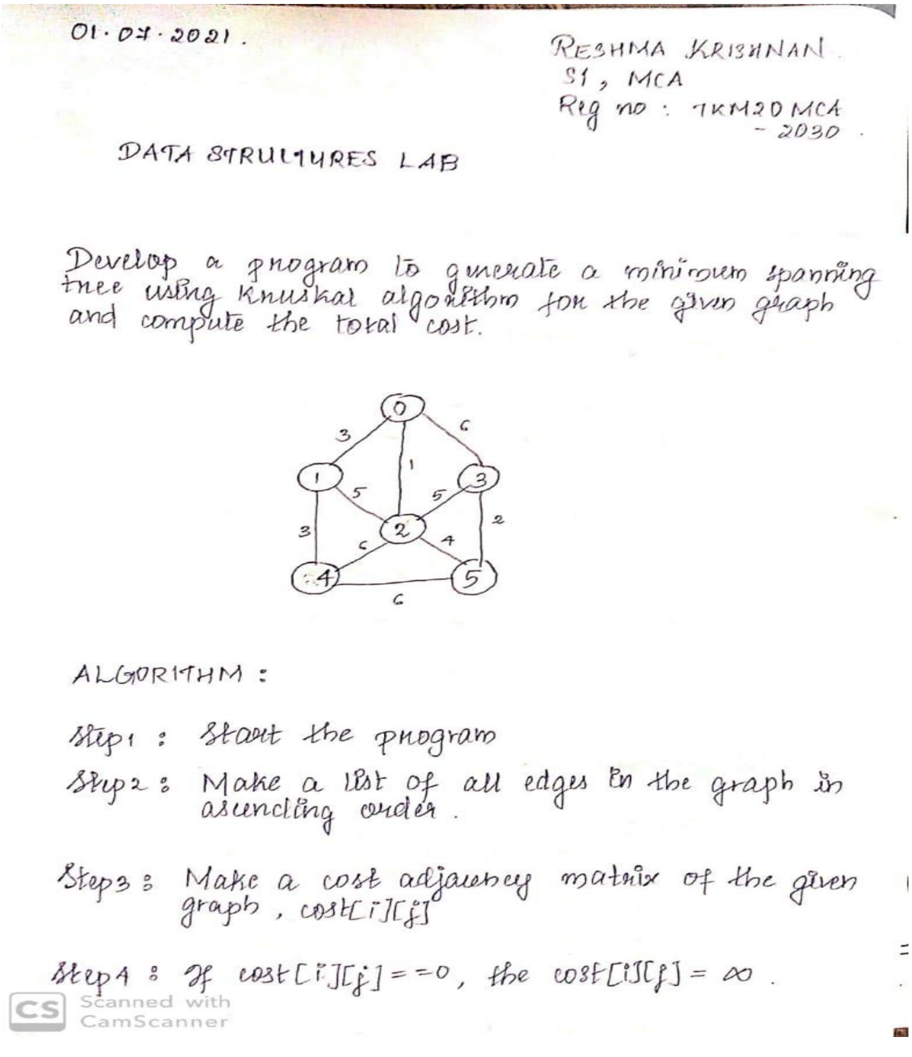
**RESHMA KRISHNAN**
**Semester-1 MCA**
**REG. No.: TKM20MCA-2030**

---

## QUESTION-1

**AIM:** To develop a program to generate a minimum spanning tree using kruskal's algorithm for the given graph and compute the total cost.

**ALGORITHM:**



01.07.2021.

RESHMA KRISHNAN.
S1, MCA
Reg no : TKM20 MCA
- 2030 .

DATA STRUCTURES LAB

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

0
1
2
3
4
5
3
6
1
5
5
2
3
6
4
6

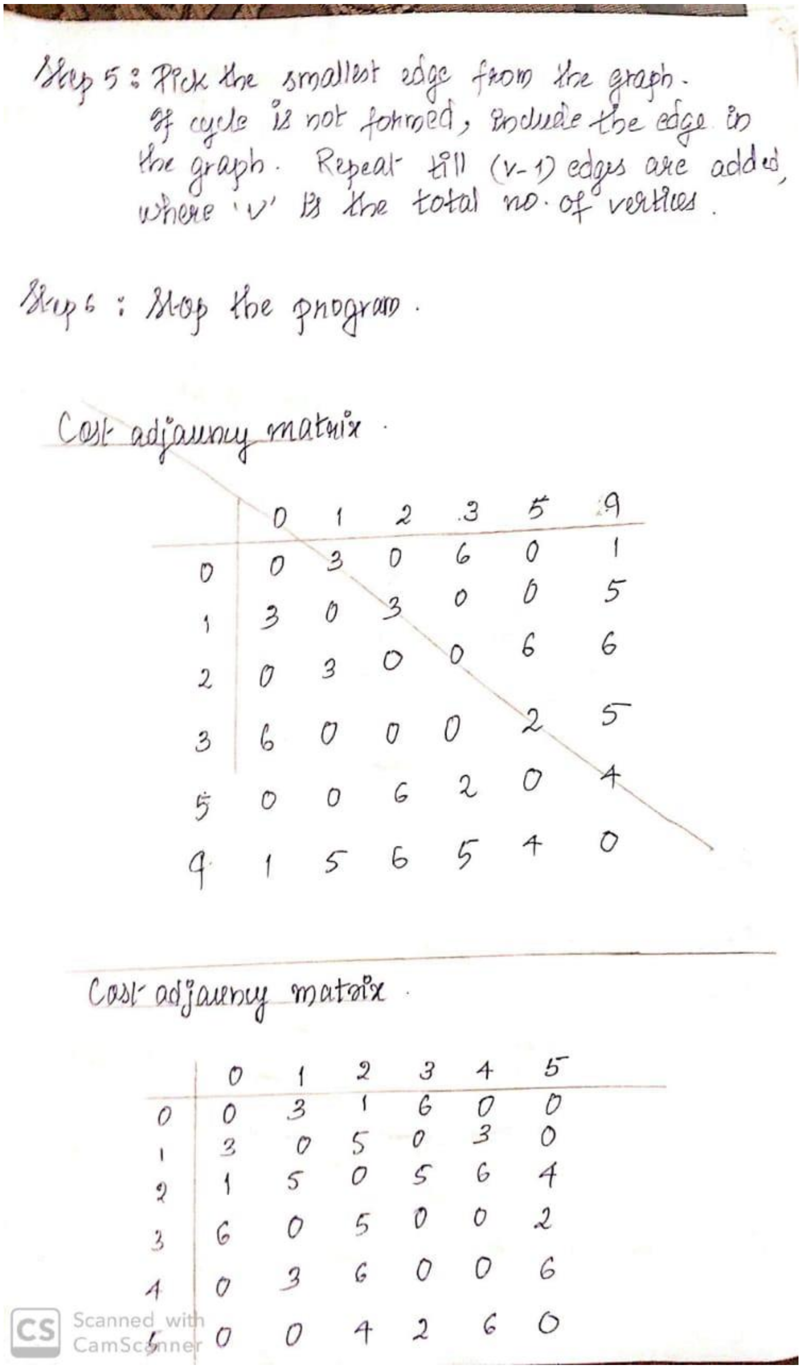
ALGORITHM :

Step1 : Start the program

Step2 : Make a list of all edges in the graph in ascending order.

Step3 : Make a cost adjacency matrix of the given graph, cost[i][j]

Step4 : If cost[i][j]==0, the cost[i][j] = ∞ .



Step 5 : Pick the smallest edge from the graph. If cycle is not formed, include the edge in the graph. Repeat till (v-1) edges are added, where 'v' is the total no. of vertices.

Step 6 : Stop the program.

Cost adjacency matrix.

$$\begin{array}{c|cccccc}
 & 0 & 1 & 2 & 3 & 5 & 9 \\
\hline
0 & 0 & 3 & 0 & 6 & 0 & 1 \\
1 & 3 & 0 & 3 & 0 & 0 & 5 \\
2 & 0 & 3 & 0 & 0 & 6 & 6 \\
3 & 6 & 0 & 0 & 0 & 2 & 5 \\
5 & 0 & 0 & 6 & 2 & 0 & 4 \\
9 & 1 & 5 & 6 & 5 & 4 & 0
\end{array}$$

Cost adjacency matrix.

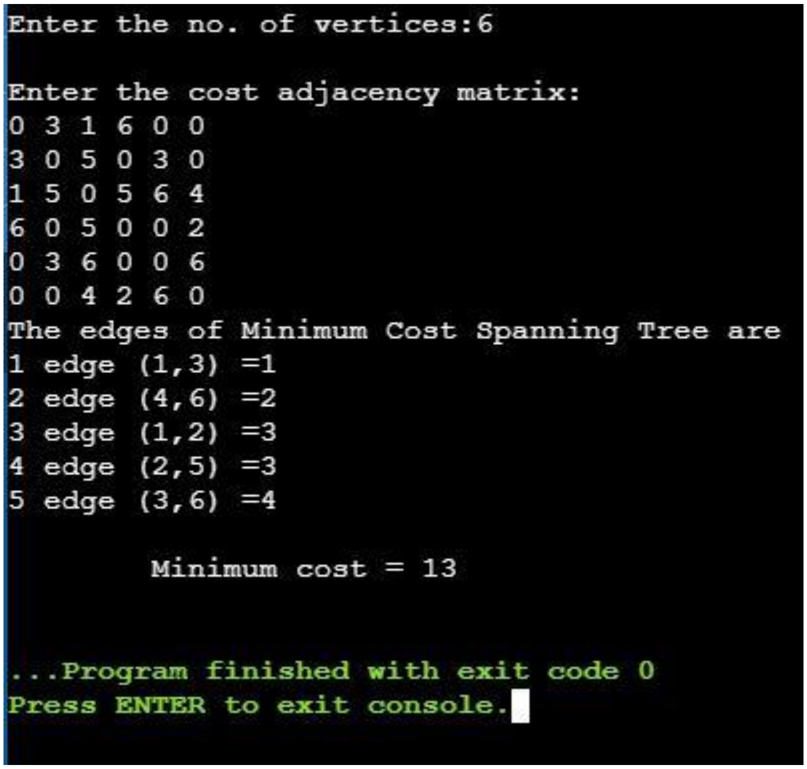
$$\begin{array}{c|cccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 & 0 & 3 & 1 & 6 & 0 & 0 \\
1 & 3 & 0 & 5 & 0 & 3 & 0 \\
2 & 1 & 5 & 0 & 5 & 6 & 4 \\
3 & 6 & 0 & 5 & 0 & 0 & 2 \\
4 & 0 & 3 & 6 & 0 & 0 & 6 \\
5 & 0 & 0 & 4 & 2 & 6 & 0
\end{array}$$

**PROGRAM CODE:**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 30
```

```
int n,i,j,cost[MAX][MAX], ne=1, min, a,b,u,v, mincost=0, parent[MAX];


int find( int );
int uni(int, int);

void main(){

    printf("Enter the no.of vertices in the graph given:");
    scanf("%d",&n);

    printf("\nEnter the cost adjacency matrix:");

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

    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            if(cost[i][j] == 0){
                cost[i][j]=9999;
            }
            else{
                 cost[i][j] = cost[i][j];}
            }

    }

    printf("\nThe edges of the minimum cost spanning tree are:\n");

    while(ne < n)
    {
        for(i=1,min=9999;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]=9999;
    }

    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:**

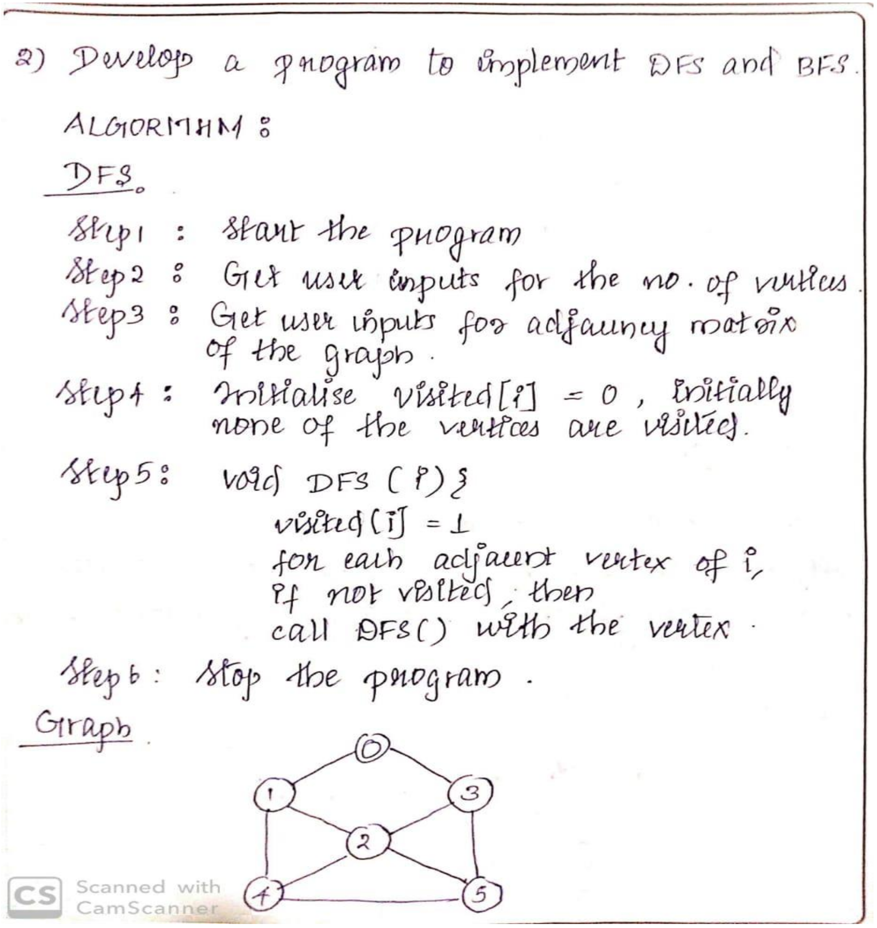


```
Enter the no. of vertices:6

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

        Minimum cost = 13


...Program finished with exit code 0
Press ENTER to exit console.
```

## QUESTION-2

**AIM:** To develop a program to implement DFS and BFS.

**ALGORITHM:**



2) Develop a program to implement DFS and BFS.

ALGORITHM :

DFS.

Step 1 : Start the program

Step 2 : Get user inputs for the no. of vertices.

Step 3 : Get user inputs for adjacency matrix of the graph.

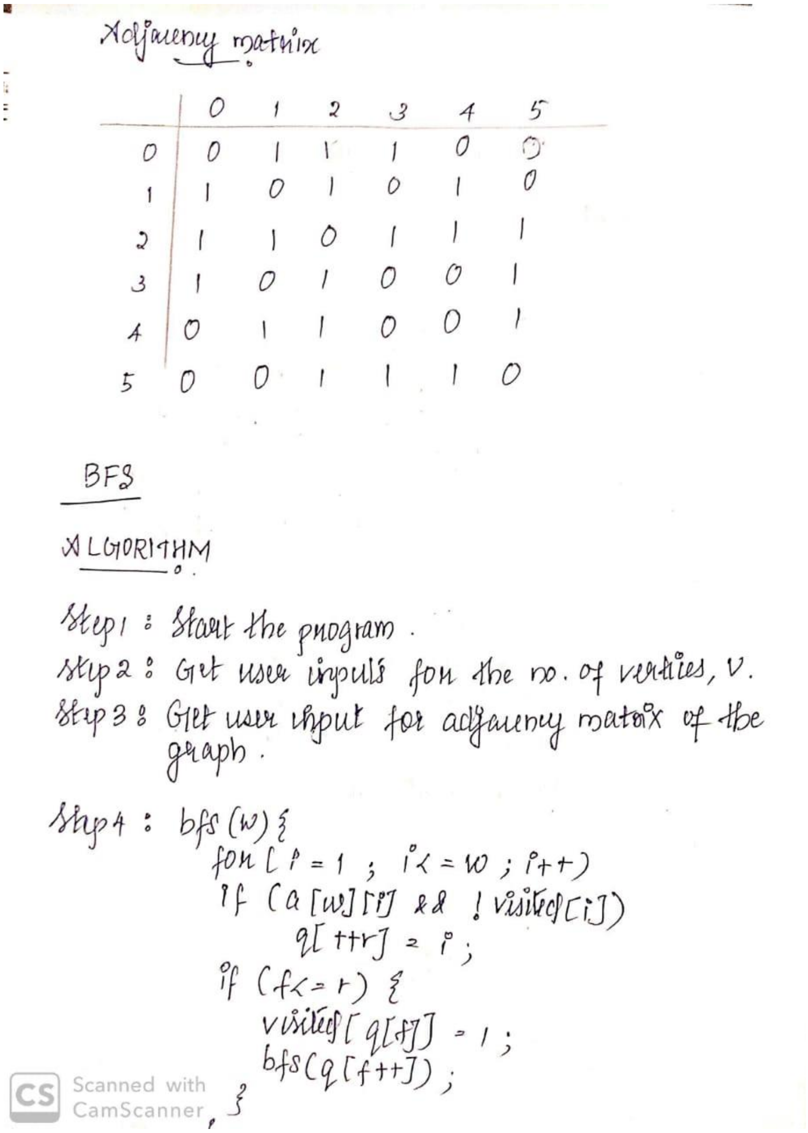
Step 4 : Initialise visited[i] = 0, initially none of the vertices are visited.

Step 5 :

```
void DFS (i) {
    visited[i] = 1
    for each adjacent vertex of i,
    if not visited, then
    call DFS() with the vertex.
```

Step 6 : Stop the program.

Graph.



Adjacency matrix

|   | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 2 | 1 | 1 | 0 | 1 | 1 | 1 |
| 3 | 1 | 0 | 1 | 0 | 0 | 1 |
| 4 | 0 | 1 | 1 | 0 | 0 | 1 |
| 5 | 0 | 0 | 1 | 1 | 1 | 0 |

BFS

ALGORITHM

Step1 : Start the program.

step2 : Get user inputs for the no. of vertices, v.

Step3 : Get user input for adjacency matrix of the graph.

```
Step4 : bfs (w) {
          for [ i = 1 ; i<= w ; i++ )
          if (a [w] [i] && ! visited[i])
               q[ ++r] = i ;
          if (f<=r) {
             visited [ q[f]] = 1 ;
             bfs(q[f++]) ;
        }
```

Scanned with CamScanner

**PROGRAM CODE:**

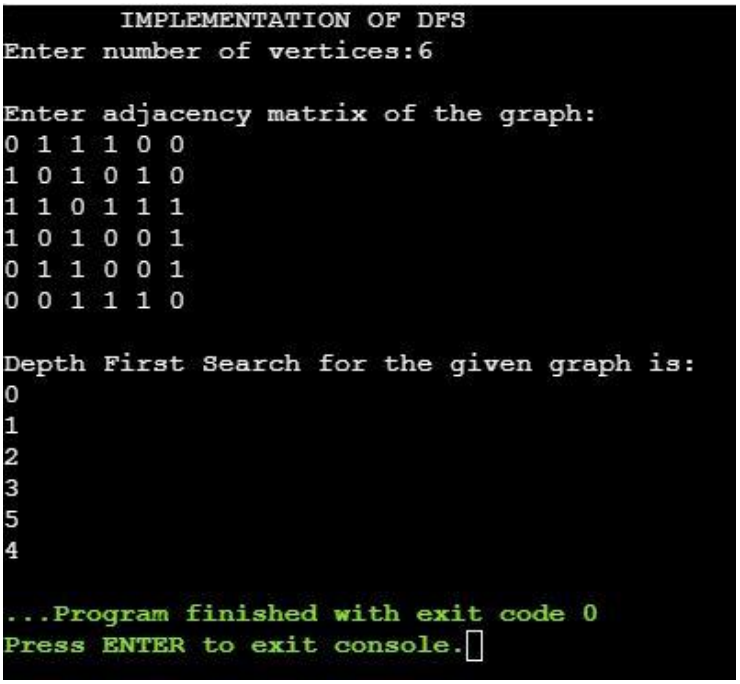
DFS.c

```
#include<stdio.h>
int v, i, j, G[30][30],visited[20];
void DFS(int);

void main()
{

    printf("        IMPLEMENTATION OF DFS        ");
    printf("\nEnter number of vertices:");
    scanf("%d", &v);

    printf("\nEnter adjacency matrix of the graph:");
    for(i=0;i<v;i++){
       for(j=0;j<v;j++){
            scanf("%d",&G[i][j]);
       }
    }

   for(i=0;i<v;i++){
        visited[i]=0;
   }
    printf("\nDepth First Search for the given graph is:");
    DFS(0);
}

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

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

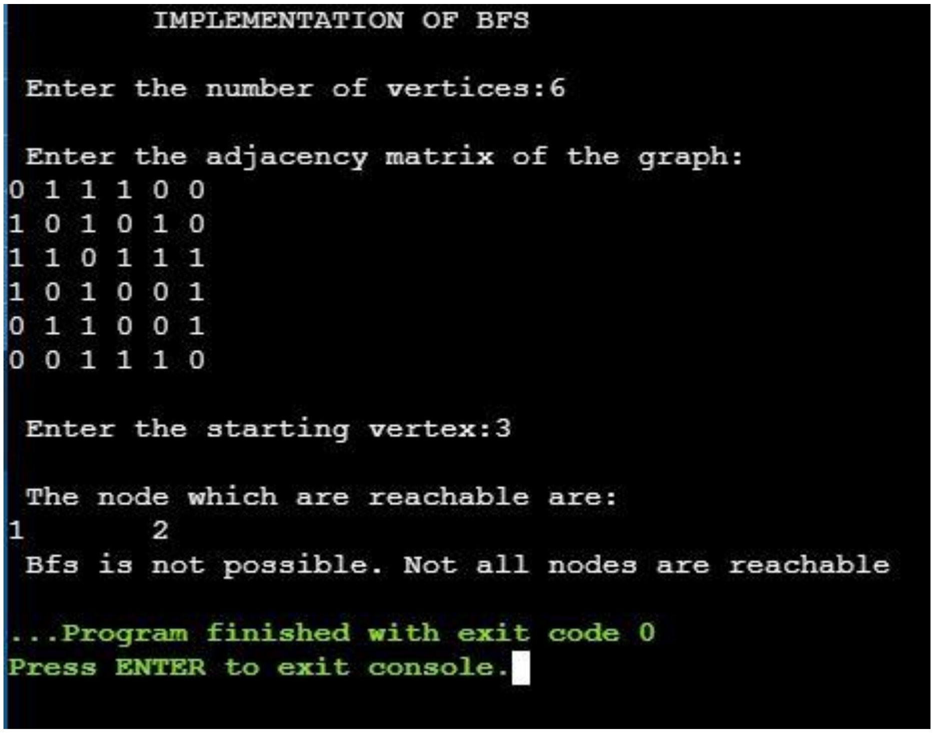
BFS.c

```
#include<stdio.h>
int v, i, j, visited[20], a[20][20], q[20], r = -1, f = 0;

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

void main() {
    int start;
    printf("\n Enter the number of vertices:");
    scanf("%d", &v);

    for(i=1; i <= v; i++) {
        q[i] = 0;
        visited[i] = 0;
    }

    printf("\n Enter graph data in matrix form:\n");
    for(i=1; i<=v; i++) {
    for(j=1;j<=v;j++) {
    scanf("%d", &a[i][j]);
   }
 }

    printf("\n Enter the starting vertex:");
    scanf("%d", &start);
    bfs(start);
    printf("\n The node which are reachable are:\n");

    for(i=1; i <= v; i++) {
        if(visited[i])
        printf("%d\t", i);
        else {
            printf("\n Bfs is not possible. Not all nodes are reachable");
            break;
     }
  }
}
```

**OUTPUT:**



```
       IMPLEMENTATION OF DFS
Enter number of vertices:6

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

Depth First Search for the given graph is:
0
1
2
3
5
4

...Program finished with exit code 0
Press ENTER to exit console.
```



```
        IMPLEMENTATION OF BFS

 Enter the number of vertices:6

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

 Enter the starting vertex:3

 The node which are reachable are:
1       2
 Bfs is not possible. Not all nodes are reachable

...Program finished with exit code 0
Press ENTER to exit console.
```