

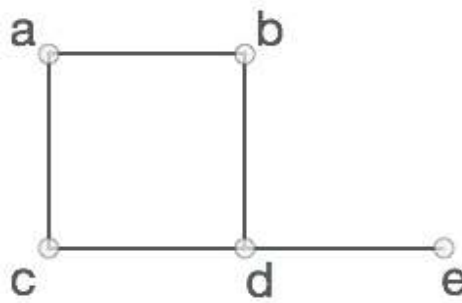
### Practical No 8

**Aim:** Write a Program to implement a graph using adjacency matrix and traverse by using DFS/BFS

**Theory:**

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as **vertices**, and the links that connect the vertices are called **edges**.

Formally, a graph is a pair of sets **(V, E)**, where **V** is the set of vertices and **E** is the set of edges, connecting the pairs of vertices. Take a look at the following graph –

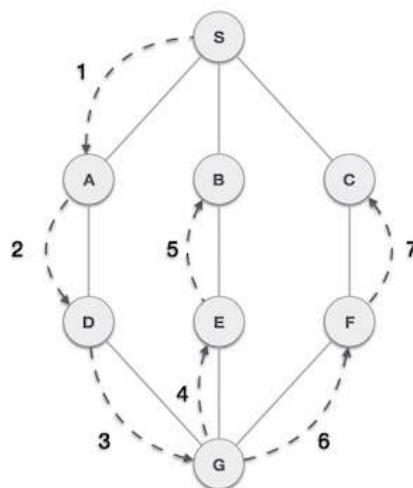


In the above graph,

$V = \{a, b, c, d, e\}$

$E = \{ab, ac, bd, cd, de\}$

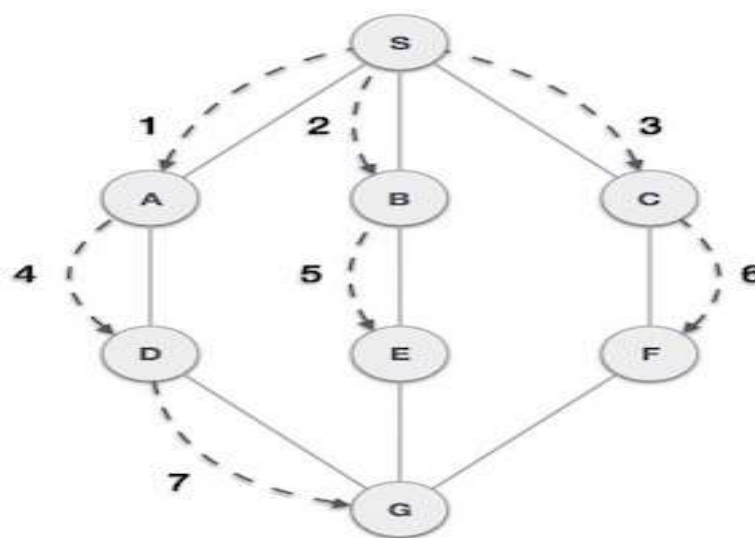
Depth First Search (DFS) algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.



As in the example given above, DFS algorithm traverses from A to B to C to D first then to E, then to F and lastly to G. It employs the following rules.

- **Rule 1** – Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack.
- **Rule 2** – If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.)
- **Rule 3** – Repeat Rule 1 and Rule 2 until the stack is empty.

Breadth First Search (BFS) algorithm traverses a graph in a breadthward motion and uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration.



As in the example given above, BFS algorithm traverses from A to B to E to F first then to C and G lastly to D. It employs the following rules.

- **Rule 1** – Visit the adjacent unvisited vertex. Mark it as visited. Display it. Insert it in a queue.
- **Rule 2** – If no adjacent vertex is found, remove the first vertex from the queue.
- **Rule 3** – Repeat Rule 1 and Rule 2 until the queue is empty.

### Program:

```
#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("Program to Illustrate concept of      a graph using adjacency
matrix and traverse by using DFS/BFS \n");
    printf("ENTER THE NUMBER VERTICES ");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;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++];
    }
}
```

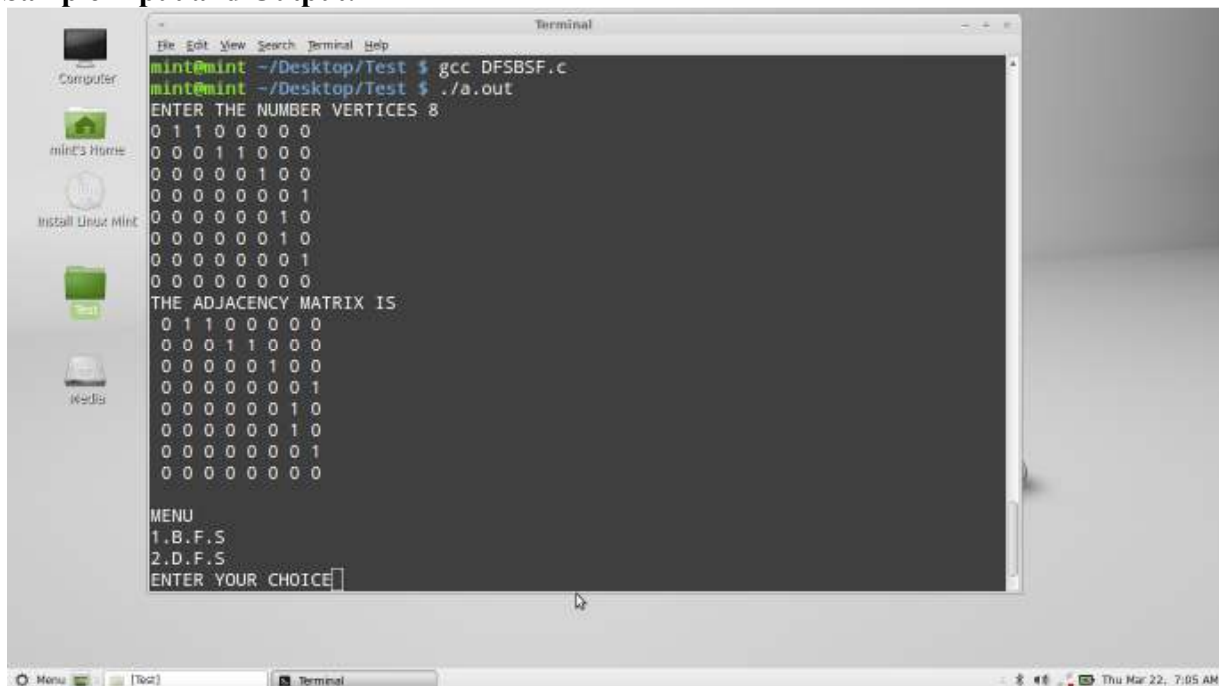
```

return(k);
}
}

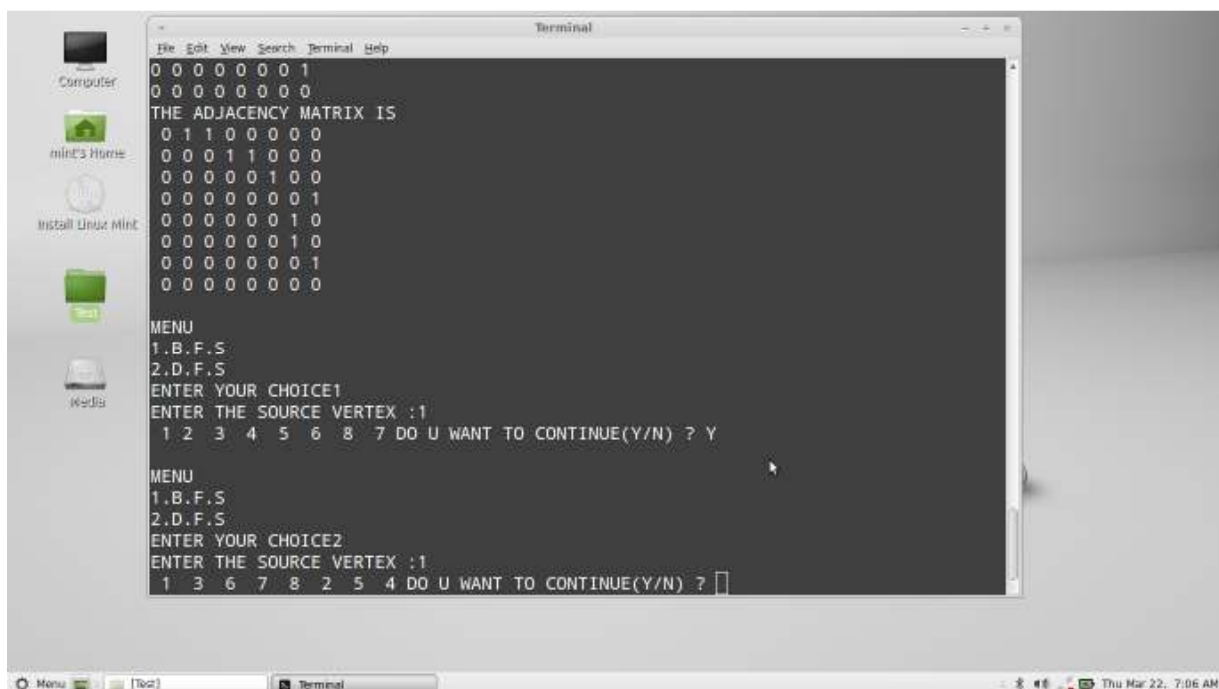
//*****DFS(depth-first search) code*****//
void dfs(int s,int n)
{
int i,k;
push(s);
vis[s]=1;
k=pop();
if(k!=0)
printf(" %d ",k);
while(k!=0)
{
for(i=1;i<=n;i++)
if((a[k][i]!=0)&&(vis[i]==0))
{
push(i);
vis[i]=1;
}
k=pop();
if(k!=0)
printf(" %d ",k);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
dfs(i,n);
}
void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

```

### Sample Input and Output:

A screenshot of a Linux desktop environment with a terminal window open. The terminal shows the compilation and execution of a program. The user enters '8' for the number of vertices, followed by an 8x8 adjacency matrix. The program then displays a menu with two options: '1.B.F.S' and '2.D.F.S'.

```
mint@mint ~/Desktop/Test $ gcc DFSBSF.c
mint@mint ~/Desktop/Test $ ./a.out
ENTER THE NUMBER VERTICES 8
0 1 1 0 0 0 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
THE ADJACENCY MATRIX IS
0 1 1 0 0 0 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE
```

A second screenshot of the terminal window showing the program's execution after user input. The user has chosen '1' for Breadth-First Search (BFS). The program prompts for a source vertex (1) and asks if they want to continue (Y/N). The user enters 'Y'. The program then prompts for a second choice (2) and a source vertex (1), followed by another 'Y/N' prompt.

```
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
THE ADJACENCY MATRIX IS
0 1 1 0 0 0 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE1
ENTER THE SOURCE VERTEX :1
1 2 3 4 5 6 8 7 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 6 7 8 2 5 4 DO U WANT TO CONTINUE(Y/N) ?
```

**Conclusion:** The Graph traversals were successfully implemented.