- **Assignment 6:** **D.0.A : 18/11/2021**

Statement : Write a C Program to Implement Breadth First Search Traversal in a Graph.

❖ **Algorithm→**

```
Algorithm-->
    create Graph G
    adj[n][n] is the adjacency matrix of the graph where n is
number of vertices
    initialize Queue q
    BFS(G,q)
    {
        declare a visited array of size n(number of vertixes)
        for i=0 to n-1 step 1 do
            visited[i] = 0
        input the starting vertex as node
        print the starting vertex
        visited[node]=1
        q.push(node)
        while(q.empty!=1) do{
            int s = q.pop()
            for j=0 to n-1 step 1 do{
                if (adj[s][j]=1 and visited[j]=0) do{
                    print(j)
                    q.push(j)
                    visited[j]=1
                }
            }
        }
    }
```

❖ **Source Code →**

```
//BFS Traversal Implementation Of Undirected Graph
#include <stdio.h>
#include <stdlib.h>
```

```c
#define max 100
int n;                      //number of vertices
int visisted[max];          //keeping track of visited
vertices
int adj[max][max];          //adjacency matrix of the graph
int front = -1, rear = -1; //used to implement queue
int queue[max];             //queue

void push(int x) //queue push operation
{
    if (front == -1 && rear == -1)
    {
        front = rear = 0;
        queue[rear] = x;
    }
    else
    {
        rear++;
        queue[rear] = x;
    }
}

int pop() //queue pop operation
{
    int temp;
    if (front == -1 && rear == -1)
    {
        exit(1);
    }
    temp = queue[front];
    front++;
    if (front > rear)
```

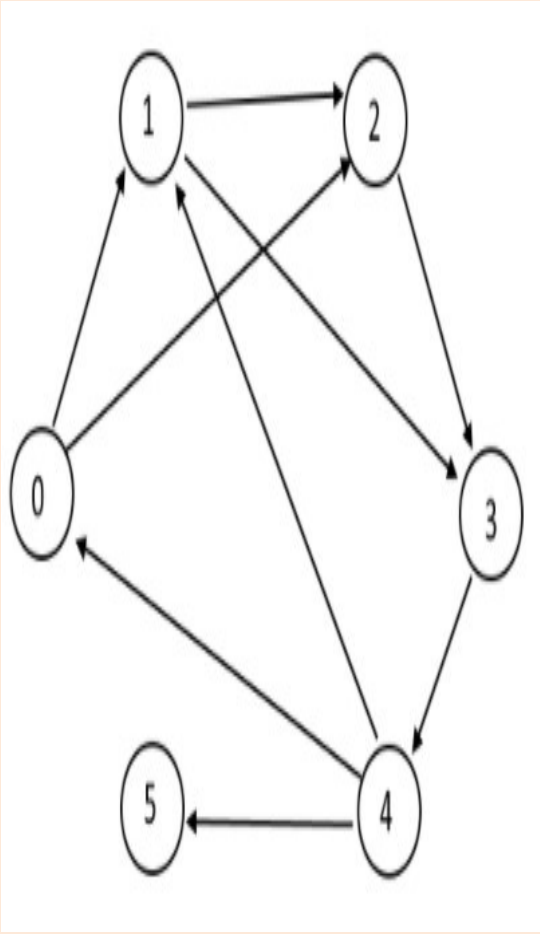
```
    {
        front = rear = -1;
    }
    return temp;
}

void creategraph() //Creation of the graph
{
    printf("Enter the number of Vertices : ");
    scanf("%d", &n);
    int e;
    printf("Enter the number of edges : ");
    scanf("%d", &e); //number of edges
    int i;
    int origin, destination;
    for (i = 0; i < e; i++)
    {
        printf("Enter the Origin : ");
        scanf("%d", &origin); //starting vertex
        printf("Enter the Destination : ");
        scanf("%d", &destination); //ending vertex
        if (origin >= n || destination >= n || origin < 0 ||
destination < 0)
        {
            printf("Enter Valid Vertex\n");
            i--;
        }
        else
        {
            adj[origin][destination] = 1;
            adj[destination][origin] = 1; //Undirected Graph
        }
    }
```

```
}

void traversalbfs()
{
    int i;
    for (i = 0; i < n; i++)
    {
        visisted[i] = 0; // no vertics is visited
    }
    int node;
    printf("Enter the vertices Where to Start : ");
    scanf("%d", &node);
    printf("BFS TRAVERSAL -->\n");
    printf("%d\t", node);
    push(node);                         //pushing the visited
nodes in the queue
    visisted[node] = 1;                 //visited vertices
    while (front != -1 && rear != -1) //checking the queue is
empty or not
    {
        node = pop();
        int j;
        for (j = 0; j < n; j++)
        {
            if (adj[node][j] == 1 && visisted[j] == 0) //if
the vertex is visited or not
            {
                printf("%d\t", j);
                push(j);          //pushing the visited nodes
in the queue
                visisted[j] = 1; //visited vertices
            }
        }
```

```
    }
}

int main()
{
    creategraph();  //creating the graph
    traversalbfs(); //BFS traversal
    return 0;
}
```

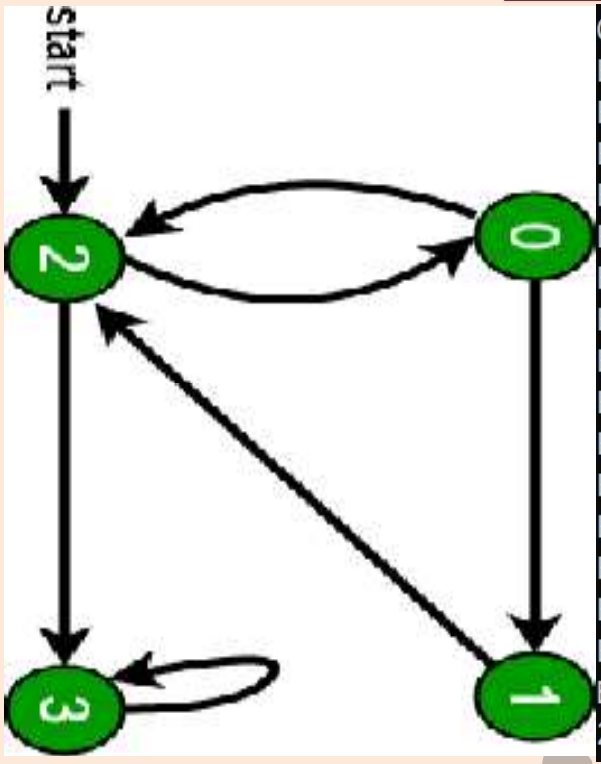
❖ **Output→**

SET (1)




```
C:\Users\MUKHERJEE\Desktop\C-C++\Graph Algorithms>a
Enter the number of Vertices : 6
Enter the number of edges : 9
Enter the Origin : 0
Enter the Destination : 1
Enter the Origin : 1
Enter the Destination : 2
Enter the Origin : 2
Enter the Destination : 3
Enter the Origin : 3
Enter the Destination : 4
Enter the Origin : 4
Enter the Destination : 5
Enter the Origin : 4
Enter the Destination : 0
Enter the Origin : 4
Enter the Destination : 1
Enter the Origin : 0
Enter the Destination : 2
Enter the Origin : 1
Enter the Destination : 3
Enter the vertices Where to Start : 0
BFS TRAVERSAL -->
0       1       2       3       4       5
```

SET (2)




```
C:\Users\MUKHERJEE\Desktop\C-C++\Graph Algorithms>a
Enter the number of Vertices : 4
Enter the number of edges : 6
Enter the Origin : 0
Enter the Destination : 2
Enter the Origin : 2
Enter the Destination : 0
Enter the Origin : 0
Enter the Destination : 1
Enter the Origin : 1
Enter the Destination : 2
Enter the Origin : 2
Enter the Destination : 3
Enter the Origin : 3
Enter the Destination : 3
Enter the vertices Where to Start : 2
BFS TRAVERSAL -->
2       0       3       1
```

❖ Discussion (Running time Complexity) →

Here We implement the BFS Traversal Algorithm Using the queue. So, In Queue the Enque and Deque operation takes O(1) time. For, V number of vertices it takes O(V) time. In, the inner loop also we are checking for every vertices. So that loop also take O(V) time. As we implement the graph using adjacency matrix representation, So our time complexity for BFS Traversal will be $O(V^2)$.