



**S. B. JAIN INSTITUTE OF TECHNOLOGY,
MANAGEMENT & RESEARCH, NAGPUR.**

Practical No. 7

Aim: Develop a program that finds and prints all the nodes reachable from a specified starting node in a directed graph using Breadth-First Search (BFS).

Name of Student: Shrutika Pradeep Bagdi

Roll No: CS22130

Semester/Year: V/III

Academic Session:2024-2025

Date of Performance:

Date of Submission:

AIM: Develop a program that finds and prints all the nodes reachable from a specified starting node in a directed graph using Breadth-First Search (BFS).

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- To understand and implement the Breadth First Search method of graph traversal.

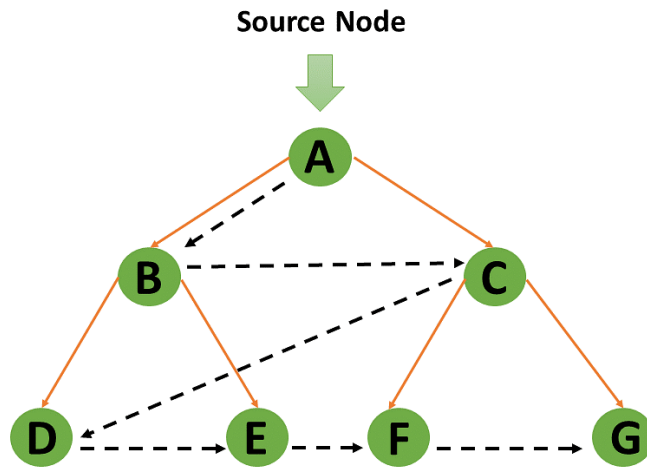
THEORY:

Breadth First Search (BFS) is a fundamental **graph traversal algorithm**. Graph traversal means visiting every vertex and edge exactly once in a well-defined order. While using certain graph algorithms, you must ensure that each vertex of the graph is visited exactly once.

It begins with a node, then first traverses all its adjacent. Once all adjacent are visited, then their adjacent are traversed level by level. BFS itself can be used to detect cycle in a directed and undirected graph, find shortest path in an unweighted graph and many more problems. Breadth-First Search uses a queue data structure technique to store the vertices.

Algorithm:

```
BFS (G, s)           //Where G is the graph and s is the source node
    let Q be queue.
    Q.enqueue( s )    //Inserting s in queue
    mark s as visited.
    while ( Q is not empty)
        v = Q.dequeue( )
        for all neighbours w of v in Graph G
            if w is not visited
                Q.enqueue( w )
                mark w as visited.
```



Applications of BFS Algorithm:

1. For GPS navigation
2. Path finding algorithms
3. In Ford-Fulkerson algorithm to find maximum flow in a network
4. Cycle detection in an undirected graph
5. In minimum spanning tree

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
typedef struct {
    int items[MAX];
    int front;
    int rear;
} Queue;
typedef struct {
    int adj[MAX][MAX];
    int numVertices;
```

```
} Graph;
void initializeGraph(Graph* g, int vertices);
void addEdge(Graph* g, int src, int dest);
void bfs(Graph* g, int start);
void enqueue(Queue* q, int value);
int dequeue(Queue* q);
int isEmpty(Queue* q);
void initializeQueue(Queue* q);
int main() {
    Graph g;
    int vertices, edges, src, dest;
    printf("Enter number of vertices: ");
    scanf("%d", &vertices);
    printf("Enter number of edges: ");
    scanf("%d", &edges);
    initializeGraph(&g, vertices);
    printf("Enter edges (src dest) one per line:\n");
    for (int i = 0; i < edges; i++) {
        scanf("%d %d", &src, &dest);
        addEdge(&g, src - 1, dest - 1);
    }
    int start;
    printf("Enter starting vertex for BFS: ");
    scanf("%d", &start);
    printf("Nodes reachable from vertex %d:\n", start);
    bfs(&g, start - 1); // Adjust for 1-based to 0-based indexing
    return 0;
}
void initializeGraph(Graph* g, int vertices) {
    g->numVertices = vertices;
    for (int i = 0; i < vertices; i++)
```

```
    for (int j = 0; j < vertices; j++)
        g->adj[i][j] = 0; // Initialize adjacency matrix
}

void addEdge(Graph* g, int src, int dest) {
    g->adj[src][dest] = 1; // Add edge from src to dest
    g->adj[dest][src] = 1; // If undirected graph
}

void bfs(Graph* g, int start) {
    Queue q;
    int visited[MAX] = {0}; // Visited array
    initializeQueue(&q);
    enqueue(&q, start);
    visited[start] = 1;

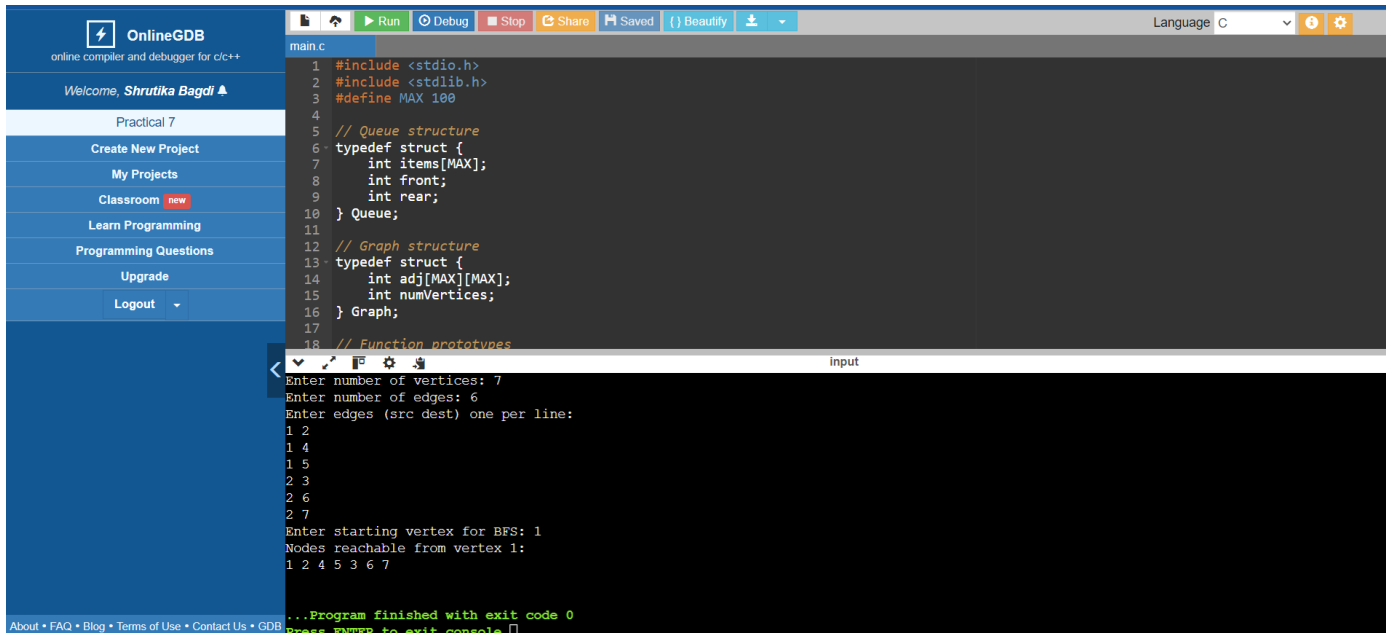
    while (!isEmpty(&q)) {
        int u = dequeue(&q);
        printf("%d ", u + 1); // Adjust output for 1-based indexing
        for (int v = 0; v < g->numVertices; v++) {
            if (g->adj[u][v] && !visited[v]) {
                enqueue(&q, v);
                visited[v] = 1; // Mark as visited
            }
        }
    }
    printf("\n");
}

void initializeQueue(Queue* q) {
    q->front = -1;
    q->rear = -1;
}

void enqueue(Queue* q, int value) {
```

```
if (q->rear == MAX - 1) {
    printf("Queue is full\n");
    return;
}
if (q->front == -1) {
    q->front = 0;
}
q->rear++;
q->items[q->rear] = value;
}
int dequeue(Queue* q) {
    if (isEmpty(q)) {
        printf("Queue is empty\n");
        return -1;
    }
    int item = q->items[q->front];
    q->front++;
    if (q->front > q->rear) {
        q->front = q->rear = -1; // Reset the queue
    }
    return item;
}
int isEmpty(Queue* q) {
    return q->front == -1;
}
```

INPUT & OUTPUT WITH DIFFERENT TEST CASES:



The screenshot displays the OnlineGDB interface. On the left, a sidebar shows the user 'Shrutika Bagdi' and a list of navigation options including 'Practical 7', 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area is split into two panes. The top pane shows the C source code for a Breadth First Search (BFS) algorithm. The code includes headers for `stdio.h` and `stdlib.h`, defines a constant `MAX` as 100, and defines two structures: `Queue` with `items`, `front`, and `rear` members, and `Graph` with an adjacency list `adj` and `numVertices`. The bottom pane shows the program's execution. It prompts for the number of vertices (7) and edges (6), then lists the edges (1 2, 1 4, 1 5, 2 3, 2 6, 2 7). It then prompts for the starting vertex (1) and displays the nodes reachable from vertex 1: 1 2 4 5 3 6 7. The program finishes with exit code 0.

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define MAX 100
4
5 // Queue structure
6 typedef struct {
7     int items[MAX];
8     int front;
9     int rear;
10 } Queue;
11
12 // Graph structure
13 typedef struct {
14     int adj[MAX][MAX];
15     int numVertices;
16 } Graph;
17
18 // Function prototypes
19
20 Enter number of vertices: 7
21 Enter number of edges: 6
22 Enter edges (src dest) one per line:
23 1 2
24 1 4
25 1 5
26 2 3
27 2 6
28 2 7
29 Enter starting vertex for BFS: 1
30 Nodes reachable from vertex 1:
31 1 2 4 5 3 6 7
32
33 ...Program finished with exit code 0
34 Press ENTER to exit console.[]
```

CONCLUSION:

DISCUSSION AND VIVA VOCE:

- Explain the Breadth First Search algorithm.
- Discuss the complexity of Breadth First Search algorithm.
- Explain the applications of Breadth First Search algorithm.

REFERENCES:

- <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>
- <https://www.javatpoint.com/breadth-first-search-algorithm>
- <https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/>
- <https://www.simplilearn.com/tutorials/data-structure-tutorial/bfs-algorithm>
- https://www.tutorialspoint.com/data_structures_algorithms/breadth_first_traversal.htm
- <https://www.programiz.com/dsa/graph-bfs>