SCHOOL OF COMPUTING

LOK RANJAN S

CH.SC.U4CSE24259

Week – 3

Design and Analysis of Algorithm(23CSE211)

1. BREADTH – FIRST SEARCH(BFS)

Code:

```
#include <stdio.h>
int queue[20], front = -1, rear = -1;
int visited[20], adj[20][20], n;
void bfs(int start) {
    int i;
    printf("BFS Traversal: ");
    queue[++rear] = start;
    visited[start] = 1;
    while (front != rear) {
        start = queue[++front];
        printf("%d ", start);
        for (i = 0; i < n; i++) {
            if (adj[start][i] == 1 && visited[i] == 0) {
                queue[++rear] = i;
                visited[i] = 1;
            }
        }
    }
}
int main() {
    int i, j, start;
    printf("Name: LOK RANJAN S\n");
    printf("Roll No: CH.SC.U4CSE24259\n\n");
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &adj[i][j]);
    for (i = 0; i < n; i++)
```

```
        visited[i] = 0;
    printf("Enter starting vertex: ");
    scanf("%d", &start);
    bfs(start);
    return 0;
}
```

Output:

```
lok-ranjan@lok-ranjan-ASUS-TUF-Gaming-A17-FA706IHRB-FA706IHRB:~/Desktop$ ./two
Name: LOK RANJAN S
Roll No: CH.SC.U4CSE24259

Enter number of vertices: 3
Enter adjacency matrix:
2
5
4
3
23
6
6
4
3
Enter starting vertex: 5
BFS Traversal: 5 lok-ranjan@lok-ranjan-ASUS-TUF-Gaming-A17-FA706IHRB-FA706IHRB:
```

⍰ Time Complexity: O(N^2)

Since the graph is represented using an Adjacency Matrix (adj[n][n]), for every vertex we visit (dequeued), we must iterate through all N potential neighbors in the inner for loop to check for connections. Doing this for all N vertices results in N * N operations.

⍰ Space Complexity: O(N)

The queue array stores the vertices to be visited. In the worst case, the queue might store all vertices of the graph, making the space complexity proportional to the number of vertices N.