DESIGN AND ANALYSIS OF ALGORITHM

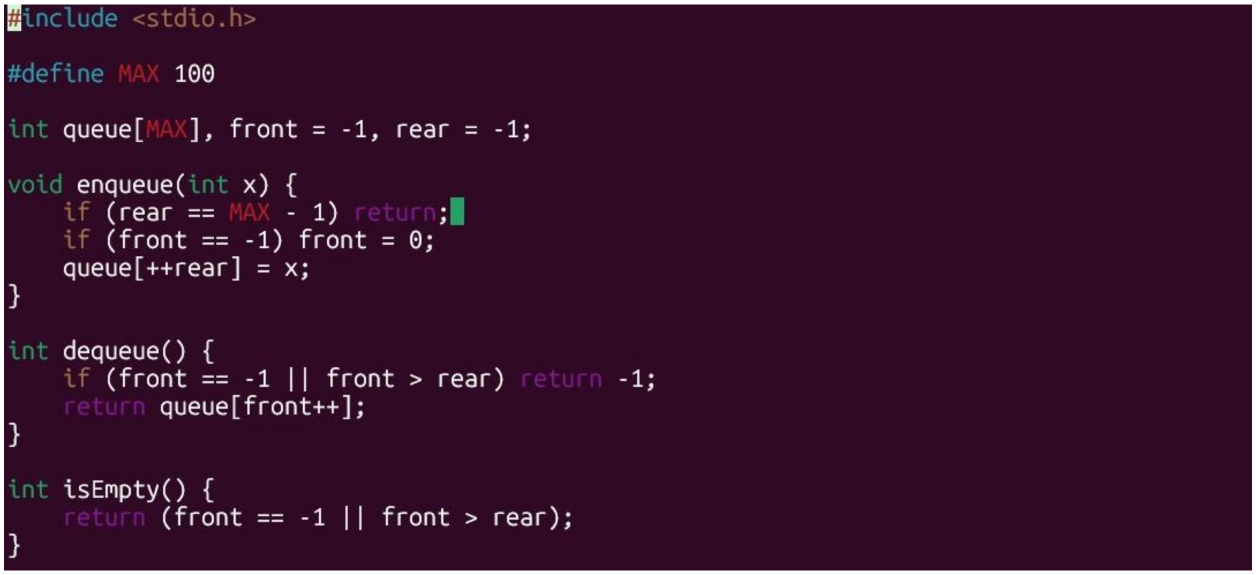
23CSE211-DFS AND BFS

NAME : UPPU MANJU SRI NATH

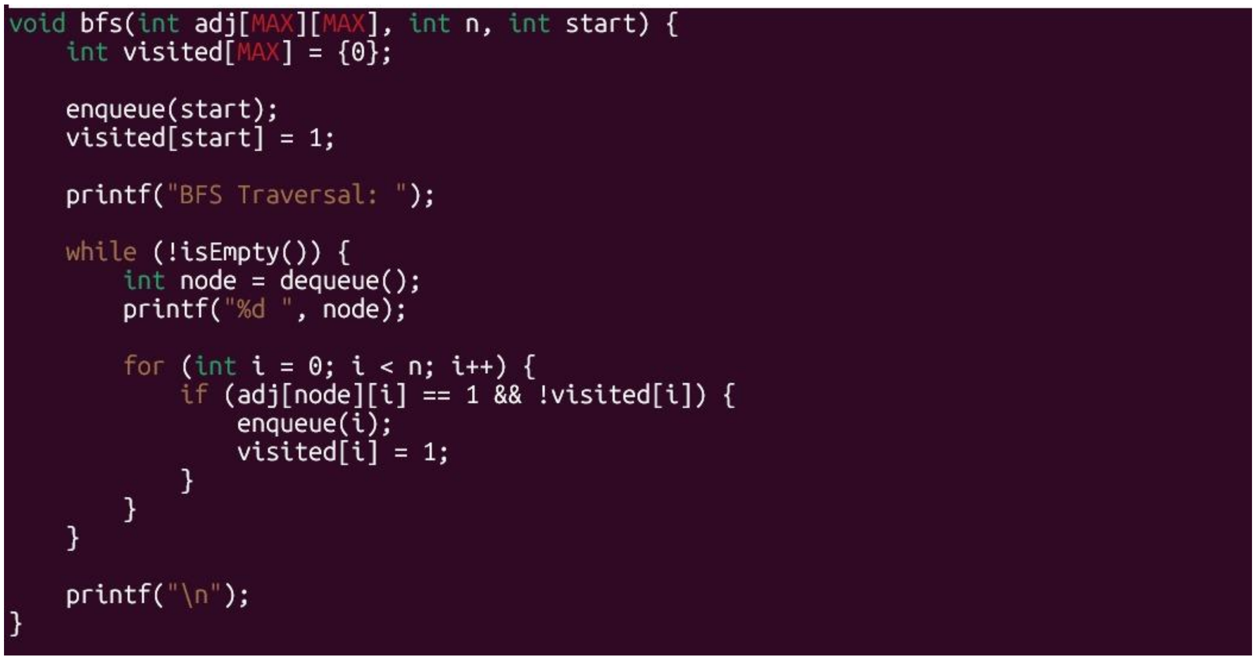
ROLL NO : CH.SC.U4CSE24249

1) **BREADTH FIRST SEARCH (BFS):**

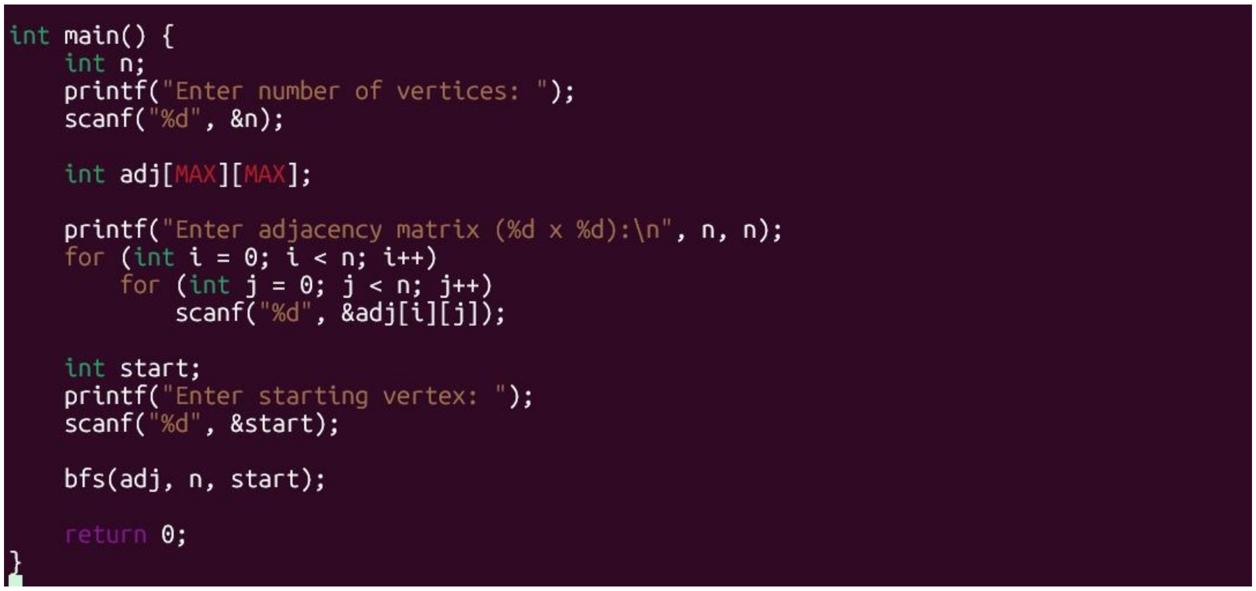
CODE:



```c
#include <stdio.h>

#define MAX 100

int queue[MAX], front = -1, rear = -1;

void enqueue(int x) {
    if (rear == MAX - 1) return;
    if (front == -1) front = 0;
    queue[++rear] = x;
}

int dequeue() {
    if (front == -1 || front > rear) return -1;
    return queue[front++];
}

int isEmpty() {
    return (front == -1 || front > rear);
}
```

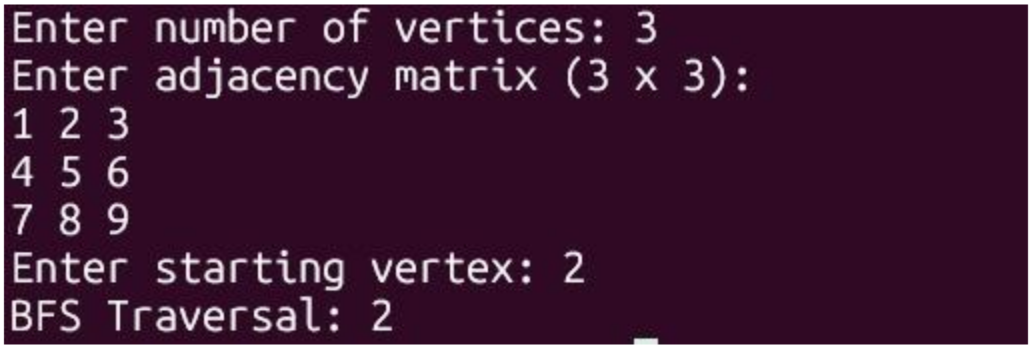


```c
void bfs(int adj[MAX][MAX], int n, int start) {
    int visited[MAX] = {0};

    enqueue(start);
    visited[start] = 1;

    printf("BFS Traversal: ");

    while (!isEmpty()) {
        int node = dequeue();
        printf("%d ", node);

        for (int i = 0; i < n; i++) {
            if (adj[node][i] == 1 && !visited[i]) {
                enqueue(i);
                visited[i] = 1;
            }
        }
    }

    printf("\n");
}
```



```c
int main() {
    int n;
    printf("Enter number of vertices: ");
    scanf("%d", &n);

    int adj[MAX][MAX];

    printf("Enter adjacency matrix (%d x %d):\n", n, n);
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &adj[i][j]);

    int start;
    printf("Enter starting vertex: ");
    scanf("%d", &start);

    bfs(adj, n, start);

    return 0;
}
```

OUTPUT:



```
Enter number of vertices: 3
Enter adjacency matrix (3 x 3):
1 2 3
4 5 6
7 8 9
Enter starting vertex: 2
BFS Traversal: 2
```

Time Complexity

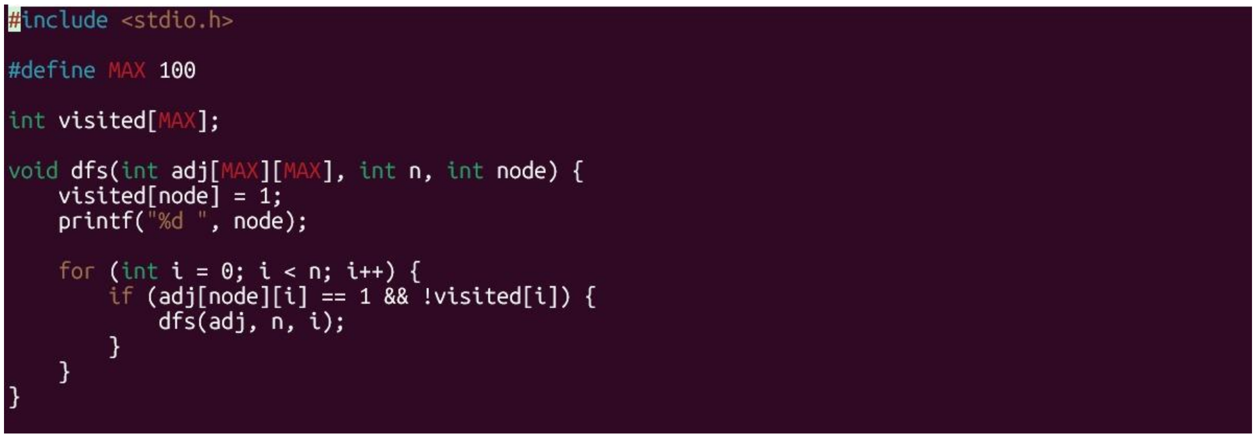
O(V + E)
BFS visits every vertex once and checks all edges connected to them.

Space Complexity

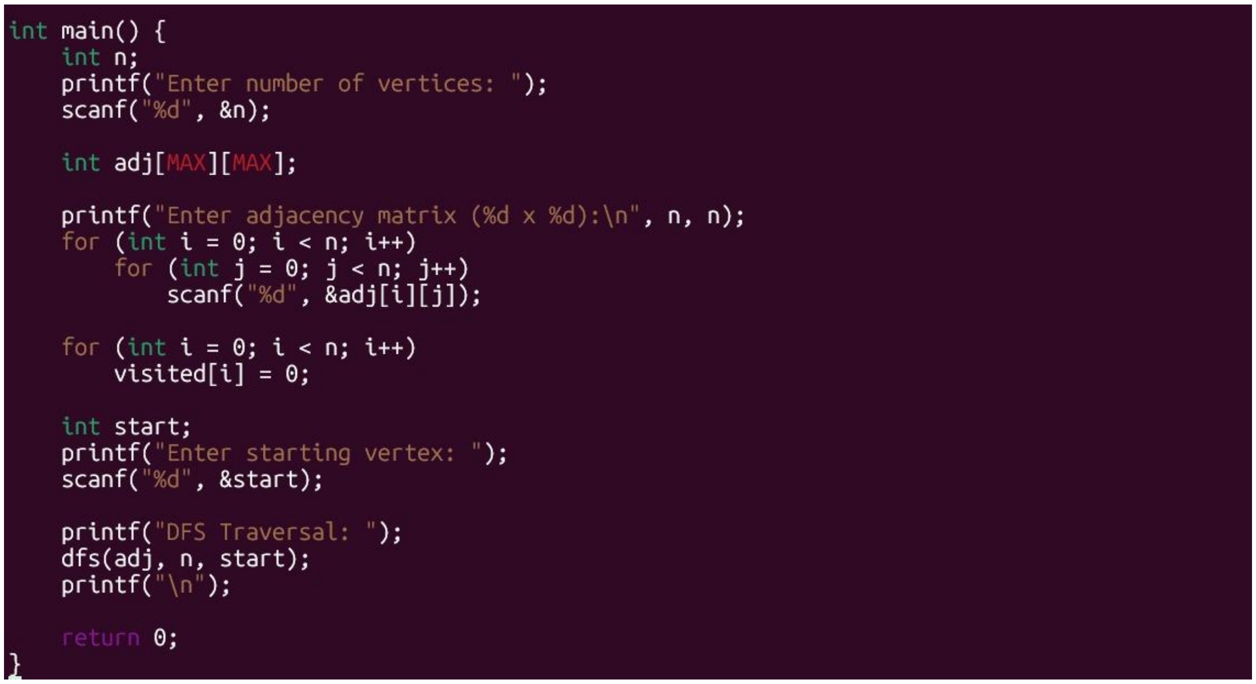
O(V) for the visited array and the queue.
O(V^2) for adjacency matrix storage (since your code uses a matrix).

2) **DEPTH FIRST SEARCH (DFS):**

CODE:

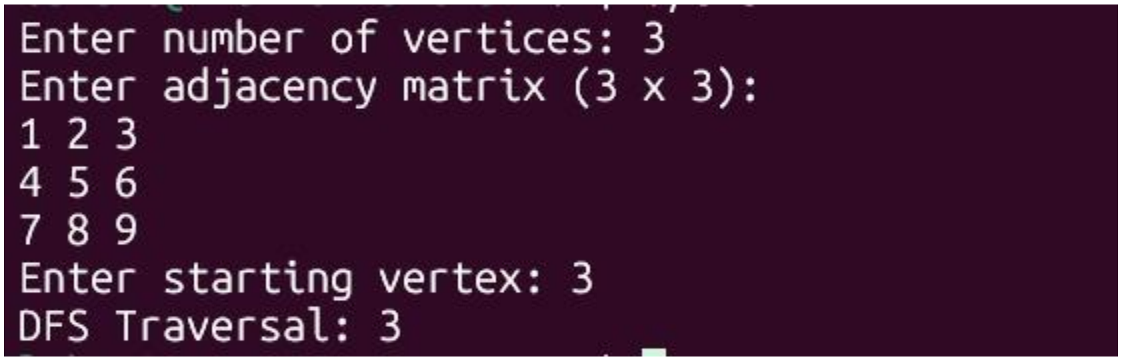


```c
#include <stdio.h>

#define MAX 100

int visited[MAX];

void dfs(int adj[MAX][MAX], int n, int node) {
    visited[node] = 1;
    printf("%d ", node);

    for (int i = 0; i < n; i++) {
        if (adj[node][i] == 1 && !visited[i]) {
            dfs(adj, n, i);
        }
    }
}
```



```c
int main() {
    int n;
    printf("Enter number of vertices: ");
    scanf("%d", &n);

    int adj[MAX][MAX];

    printf("Enter adjacency matrix (%d x %d):\n", n, n);
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &adj[i][j]);

    for (int i = 0; i < n; i++)
        visited[i] = 0;

    int start;
    printf("Enter starting vertex: ");
    scanf("%d", &start);

    printf("DFS Traversal: ");
    dfs(adj, n, start);
    printf("\n");

    return 0;
}
```

OUTPUT:



```
Enter number of vertices: 3
Enter adjacency matrix (3 x 3):
1 2 3
4 5 6
7 8 9
Enter starting vertex: 3
DFS Traversal: 3
```

Time Complexity

O(V + E)
DFS visits every vertex once and explores all edges exactly one time.

Space Complexity

O(V) for the recursion stack + visited array.
O(V^2) for adjacency matrix storage (used by your code).