AMRITA SCHOOL OF COMPUTING

DESIGN AND ANALYSIS OF ALGORITHMS

(23CSE211)

Name: Y.V.S.Likesh

Roll No.: CH.SC.U4CSE24152

Class: BTech (CSE-B)

School: Amrita School of Computing,
Chennai Campus.

LAB-3

1) BFS

Code:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100
int queue[MAX], front = -1, rear = -1;
int visited[MAX];
void enqueue(int v){
    if(rear == MAX-1){
      return;
    }
    if(front == -1){
      front = 0;
    }
    queue[++rear] = v;
}
int dequeue(){
    if(front == -1){
      return -1;
    }
    int v = queue[front++];
    if(front > rear){
      front = rear = -1;
    }
    return v;
}
void BFS(int adj[MAX][MAX],int n,int start){
    for(int i=0;i<n;i++){
        visited[i] = 0;
    }
    enqueue(start);
    visited[start] = 1;
    while(front != -1){
        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;
           }
        }
    }
}
```

```
int main(){
    int n;
    int adj[MAX][MAX];
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix:\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 node: ");
    scanf("%d", &start);
    printf("BFS Traversal: ");
    BFS(adj, n, start);
    printf("\nCH.SC.U4CSE24152\n");
    return 0;
}
```

Output:

```
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Documents$ gcc bfs.c
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Documents$ ./a.out
Enter number of vertices: 3
Enter adjacency matrix:
0 1 1
1 0 0
1 0 0
Enter starting node: 0
BFS Traversal: 0 1 2
CH.SC.U4CSE24152
```

Space Complexity:

The space complexity of this program is O(n^2) – the graph is represented using an adjacency matrix.

Time Complexity:

The time complexity of this program is O(n^2). Because it is using adjacency matrix.

2) DFS

Code:

```c
#include <stdio.h>
#define MAX 100
int visited[MAX];
void DFS(int adj[MAX][MAX],int n,int start){
    printf("%d ", start);
    visited[start] = 1;
    for(int i=0;i<n;i++){
      if(adj[start][i] == 1 && !visited[i]){
        DFS(adj, n, i);
      }
    }
}
int main(){
    int n;
    int adj[MAX][MAX];
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix:\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);
    for(int i=0;i<n;i++){
      visited[i] = 0;
    }
    printf("DFS Traversal: ");
    DFS(adj, n, start);
    printf("\nCH.SC.U4CSE24152\n");
    return 0;
}
```

Output:

```
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Documents$ gcc Dfs.c
y-v-s-likesh@y-v-s-likesh-VirtualBox:~/Documents$ ./a.out
Enter number of vertices: 3
Enter adjacency matrix:
0 1 1
1 0 0
1 0 0
Enter starting vertex: 0
DFS Traversal: 0 1 2
CH.SC.U4CSE24152
```

Space Complexity:

The space complexity of this program is O(n) – using array and recursion stack in worst case.

Time Complexity:

The time complexity of this program is O(n^2). Because Adjacent matrix is used.