Khwopa College of Engineering

DSA Lab Sheet

II Year/ II Part

Faculty: Computer and Electronics

Labsheet #9

Objectives:

1. Implementation of Breadth-First Search (BFS) and Depth-First Search (DFS) to traverse a graph.

Theory:

Graph Basics

- A Graph consists of vertices (nodes) and edges (connections).
- Graphs can be:
    - Directed or undirected
    - Weighted or unweighted
    - Connected or disconnected

BFS (Breadth-First Search)

- Traverses level by level.
- Uses Queue data structure.
- Suitable for finding the shortest path in unweighted graphs.

DFS (Depth-First Search)

- Traverses as deep as possible before backtracking.
- Uses Stack (can be recursive or manual stack).
- Explores depth before breadth.

Algorithm:

Breadth-First Search (BFS) Algorithm

```
Step 1: Initialize all nodes as unvisited.
Step 2: Start from the source node.
Step 3: Visit the node and enqueue it.
Step 4: While queue is not empty:
    a. Dequeue a node
    b. For each unvisited adjacent node:
        i. Mark it visited
ii. Enqueue it
```

Depth-First Search (DFS) Algorithm

Step 1: Start from the source node.

Step 2: Visit the node and mark it as visited.

Step 3: For each unvisited adjacent node:

a. Recursively apply DFS

Execution Code:

```
#include <stdio.h>
#define SIZE 10

int adj[SIZE][SIZE], visited[SIZE];
int queue[SIZE], front = -1, rear = -1;

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

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

// BFS function
void bfs(int start, int n) {
    for (int i = 0; i < n; i++) visited[i] = 0;
    enqueue(start);
    visited[start] = 1;
    printf("BFS Traversal: ");

    while (front <= rear) {
        int node = dequeue();
        printf("%d ", node);
        for (int i = 0; i < n; i++) {
            if (adj[node][i] && !visited[i]) {
                enqueue(i);
                visited[i] = 1;
            }
        }
    }
    printf("\n");
}

// DFS function (recursive)
void dfs(int node, int n) {
    visited[node] = 1;
    printf("%d ", node);
    for (int i = 0; i < n; i++) {
        if (adj[node][i] && !visited[i])
            dfs(i, n);
    }
}
```

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

    // Initializing adjacency matrix
    for (int i = 0; i < edges; i++) {
        printf("Enter edge (u v): ");
        scanf("%d%d", &u, &v);
        adj[u][v] = adj[v][u] = 1;
    }

    printf("Enter starting vertex for BFS & DFS: ");
    scanf("%d", &start);

    bfs(start, n);

    for (int i = 0; i < n; i++) visited[i] = 0;
    printf("DFS Traversal: ");
    dfs(start, n);
    printf("\n");

    return 0;
}
```

Tasks:

1. Create an undirected graph with minimum of 5 nodes.
2. Perform BFS from node 0.
3. Perform DFS from node 0.
4. Print visited order in both BFS and DFS.