Assignment No. 06

Title:

Represent a given graph using an adjacency matrix/list to perform DFS and using an adjacency list to perform BFS. Use the map of the area around the college as the graph. Identify the prominent landmarks as nodes and perform DFS and BFS on that.

Objectives:

1. To understand the concept of Graph Traversal techniques.
2. To learn and implement Depth-First Search (DFS) and Breadth-First Search (BFS).

Learning Objectives:

1.Understand graph traversal techniques using DFS and BFS.

2.Implement adjacency matrix and adjacency list representations.

3.Apply DFS using an adjacency matrix or list.

4.Apply BFS using an adjacency list.

Learning Outcomes:

- Define a class for the graph using Object-Oriented Programming (OOP) concepts.
- Analyze the working of DFS and BFS functions.
- Represent real-world scenarios using graphs.

Theory:

Graphs are fundamental data structures used to model relationships between different entities. They consist of **nodes (vertices)** and **edges (connections between nodes).**

Graph Representation:

1. **Adjacency Matrix:**
    - A 2D array where matrix[i][j] is 1 if there is an edge between node i and node j, otherwise 0.
    - Best for dense graphs but consumes more space for sparse graphs.
2. **Adjacency List:**
    - Each node maintains a list of all its neighboring nodes.

- Efficient for sparse graphs, using less memory than an adjacency matrix.

Graph Traversal Techniques:

1. Depth-First Search (DFS)

DFS is a recursive graph traversal algorithm that explores as far as possible along one branch before backtracking. It uses **stack (or recursion)** for traversal.

Algorithm for DFS:

1. Start at a given node.
2. Mark the node as visited and explore an adjacent unvisited node.
3. Repeat the process for each unvisited neighbor until no more nodes remain.
4. Backtrack if no adjacent unvisited nodes are left.

Time Complexity: O(V + E), where V = vertices, E = edges.

2. Breadth-First Search (BFS)

BFS explores all neighbors of a node before moving to the next level of neighbors. It uses a **queue** for traversal.

Algorithm for BFS:

1. Start at a given node and enqueue it.
2. Dequeue a node, mark it as visited, and enqueue all its unvisited neighbors.
3. Repeat until the queue is empty.

Time Complexity: O(V + E), where V = vertices, E = edges.

Graph Representation for the Given Problem:

For this assignment, we consider a **real-world map around the college** where:

- **Nodes** represent prominent landmarks (e.g., College, Library, Canteen, Bus Stop, Hostel, etc.).
- **Edges** represent roads connecting these landmarks.

Example Graph (Landmarks & Paths):

- **Nodes (Landmarks):**
  - A: College
  - B: Library
  - C: Canteen
  - D: Bus Stop
  - E: Hostel
- **Edges (Connections between landmarks):**
  - A ↔ B
  - A ↔ C
  - B ↔ D
  - C ↔ D
  - D ↔ E

Graph Representation:

1. **Adjacency Matrix Representation (for DFS)**

```
   A  B  C  D  E
A  [0, 1, 1, 0, 0]
B  [1, 0, 0, 1, 0]
C  [1, 0, 0, 1, 0]
D  [0, 1, 1, 0, 1]
E  [0, 0, 0, 1, 0]
```

2. **Adjacency List Representation (for BFS)**

```
A -> B, C
B -> A, D
C -> A, D
D -> B, C, E
E -> D
```

Software Required:

- g++ / gcc compiler (Linux/Windows).
- IDE: Eclipse / CodeBlocks / VS Code.

Input:

1. Number of landmarks (nodes).
2. Connections between landmarks (edges).

Output:

- Adjacency matrix representation (for DFS).
- Adjacency list representation (for BFS).

- DFS and BFS traversal sequence.

Conclusion:

This program provides insights into graph traversal techniques and their real-world applications. DFS is implemented using an adjacency matrix or list, and BFS is implemented using an adjacency list for efficient memory usage.

Outcome:

Upon completion, students will be able to:

- Understand graph representations and traversal techniques.
- Implement adjacency matrix and adjacency list structures.
- Perform DFS and BFS traversals on real-world graphs.

Possible University Exam Questions:

1. What are the differences between DFS and BFS?
2. Explain adjacency matrix and adjacency list with an example.
3. What is the time complexity of DFS and BFS?