Graph Traversal: DFS & BFS – Student Notes

1. Introduction to Graph Traversal

Graph traversal is the process of visiting nodes in a graph following a specific strategy. Two fundamental graph traversal algorithms are Depth-First Search (DFS) and Breadth-First Search (BFS). Understanding these algorithms is essential for searching, pathfinding, cycle detection, networking, and many other applications.

2. Depth-First Search (DFS)

DFS explores a graph by moving as far as possible along one branch before backtracking. It uses a Last-In First-Out (LIFO) strategy, typically implemented with a stack. DFS is useful for exploring deep structures, detecting cycles, and performing topological sorting.

Key Characteristics of DFS:

- Uses a Stack (LIFO) - Moves deep into a branch before exploring siblings - May not find the shortest path - Works well for maze exploration, cycle detection, and recursion tracing

3. Breadth-First Search (BFS)

BFS explores a graph level by level, visiting all neighbors of a node before moving to the next layer. It uses a First-In First-Out (FIFO) queue. BFS is ideal for finding the shortest path in an unweighted graph.

Key Characteristics of BFS:

- Uses a Queue (FIFO) - Explores nodes in layers - Finds shortest paths in unweighted graphs - Ideal for routing and broadcasting problems

4. Explanation of the Simulation Code

The provided Python code creates an interactive graphical simulation of DFS and BFS using Tkinter. It visualizes node visitation, the frontier structure (stack or queue), and allows step-by-step execution.

4.1 Graph Definition

The graph is defined as an adjacency list in a Python dictionary, where each key is a node and each value is a list of neighboring nodes. Node positions are pre-defined to place them on the canvas.

4.2 DFS and BFS State Builders

Instead of running the algorithms directly, the program builds a list of 'states'. Each state records: - The current node - The frontier (stack for DFS, queue for BFS) - The visited nodes so far This allows step-by-step simulation.

4.3 GraphCanvas Class

Handles drawing nodes, edges, and updating node colors on the Tkinter canvas. Nodes turn orange when currently visited and green when marked as visited.

4.4 AlgorithmAnimator Class

This class manages user interaction. It loads the correct algorithm based on the dropdown selection, updates the UI labels, and animates traversal one step at a time. The 'Step' button executes exactly one state from the pre-built sequence.

4.5 Main Application

Creates the Tkinter window, loads the graph canvas, and displays the algorithm controls. Students interact with buttons and dropdown menus to visualize algorithm behavior.

5. Exercises

Exercise 1: Manual DFS

Given the graph in the simulation, write the DFS visitation order starting from node A. Show the stack contents after each step.

Exercise 2: Manual BFS

Using the same graph, write the BFS visitation order starting from node A. Show the queue contents after each step.

Exercise 3: Modify the Graph

Change the graph structure by adding a new node (G) or modifying edges. Predict how the DFS and BFS orders will change. Then run the code and compare your predictions with the simulation.

Exercise 4: Compare Algorithms

Write a short paragraph explaining why BFS always finds the shortest path in an unweighted graph, while DFS may not.

End of Notes