Final lab test

AI

1. **Problem Statement:** You are given a graph where nodes represent cities and edges represent roads between them. Each road has a cost (like distance or travel time). Your task is to find the shortest path from a starting city to a destination city using both DFS and Best-First Search (BFS).

Tasks:

1. **Graph Representation:**
   - Choose a data structure (e.g., adjacency list) to represent the graph.
   - Create a graph with at least 10 cities and the corresponding roads with costs.
2. **Algorithm Implementation:**
   - Implement the DFS algorithm to find a path from the starting city to the destination city.
   - Implement the Best-First Search algorithm using a heuristic based on distance to prioritize paths.
3. **Pathfinding:**
   - For both algorithms, output the found path and the total cost.
   - If no path exists, indicate this clearly.
4. **Performance Analysis:**
   - Compare the two algorithms based on:
     - Path cost (length)
     - Number of nodes explored
     - Memory usage
5. **Discussion:**
   - Discuss the strengths and weaknesses of DFS and Best-First Search.
   - In which situations might you choose one algorithm over the other?

Submission Requirements:

- A report detailing your methods, findings, and performance analysis.
- Code for both DFS and Best-First Search implementations.
- Optional: Visuals or diagrams of the graph and paths.

2. **Problem Statement:** You need to create an AI algorithm to solve the map coloring problem. Given a map as a graph where regions are nodes and edges represent borders, assign colors to each region such that no two adjacent regions share the same color, using no more than three colors.

Algorithm Design

a. Briefly describe the algorithm you would use to solve the map coloring problem (e.g., greedy algorithm). Outline the main steps involved.

Implementation

b. Write a Python function that accepts a graph as an adjacency list and returns a valid coloring of the graph using a maximum of three colors. Include error handling for cases where coloring is not possible.

Complexity Analysis

c. Provide a brief analysis of the time and space complexity of your algorithm. Discuss how it scales with the number of nodes and edges in the graph.

Test Cases

d. Create three test cases, including one case that cannot be colored with three colors. Explain the expected output for each case.

Grading Criteria:

- Correctness and efficiency of implementations (40%)
- Clarity of performance analysis (30%)
- Organization of the report (20%)
- Quality of visuals (10%)