Flight Management System (FMS)

Objective

To develop a robust Flight Management System (FMS) that efficiently manages and optimizes flight routes, schedules, and costs between cities. The system will be built using advanced data structures and algorithms, enabling both administrators and users to interact with the system for effective flight management and optimal route recommendations.

System Overview

The FMS will leverage:

- Graph-based data structure to model cities as nodes and flight routes (with associated costs) as edges.
- Dijkstra's Algorithm to calculate the shortest and most economical route between cities.
- Binary Search Trees (BST) to organize flight schedules for efficient time-based flight searches.
- Greedy algorithms, recursion, sorting, and searching techniques to optimize performance.

System Features

1. **User Roles and Menus**

Upon login, users will choose their role, unlocking specific functionalities:

a) **Admin Menu:**

Admins can manage the graph structure representing cities and flight routes:

1. Add City: Add a city (node) to the graph.
2. Remove City: Remove a city and its associated routes.
3. Add Route: Create a route (edge) between two cities with a specified cost.
4. Remove Route: Delete an existing route between two cities.

5. Update Route Cost: Change the cost of a specific route.

6. View Current Graph: Visualize the current graph data structure.

7. Exit: Log out of the admin interface.

b) **User Menu:**

Users can search for flights and view available cities:

1. Search Flight: Find the shortest cost route between two cities using Dijkstra's Algorithm.

2. View All Cities: List all cities in the system.

3. Flight Schedule Search: Use a Binary Search Tree (BST) to search flight schedules based on a preferred departure time and find the earliest available flight.

4. Exit: Log out of the user interface.

Concepts and Algorithms Used

1. Graph-Based Data Structure:

- Cities represented as nodes.
- Flight routes and costs represented as weighted edges.

2. Dijkstra's Algorithm:

- Computes the shortest and most economical flight route.
- Prioritizes efficiency and accuracy using a priority queue.

3. Binary Search Tree (BST) for Flight Schedules:

- Flights are stored as nodes in a BST, where each node contains:
- Departure time.
- Arrival time.
- Associated route and cost.
- Searching for the earliest flight after a given time is achieved in O(log n) time.

4. Queues:

- Used in BFS (Breadth-First Search) for graph traversal or implementing priority queues in Dijkstra's Algorithm.

5. Sorting and Searching:

- To organize and retrieve flight schedules efficiently.

6. Greedy Algorithms:

- Applied in Dijkstra's Algorithm to find optimal flight routes by making the locally optimal choice at each step.

7. Recursion:

- Used in BST operations like insertion, deletion, and searching.

Development Scope and Benefits

1. For Admins:

- Comprehensive tools for graph data management.
- Real-time updates to the flight network.

2. For Users:

- Accurate and optimized flight recommendations.
- Flexible and efficient search functionality for departure and arrival schedules.
- Simple and intuitive interface for booking decisions.

Implementation Plan

1. **Phase 1**: Graph Data Structure Development

- Implement nodes (cities) and edges (routes with costs).
- Develop functionalities for adding, removing, and updating graph elements.

2. **Phase 2:** Dijkstra's Algorithm Integration

- Build the shortest path algorithm for route optimization.

3. **Phase 3**: Binary Search Tree for Flight Schedules

- Implement BST to store and manage flight departures and arrivals.

4. **Phase 4**: User Interface Development

- Design role-based menus for Admin and User.
- Implement input handling, menu navigation, and data validation.

5. **Phase 5**: Testing and Optimization