COP5536 - Advanced Data Structures
Project Assignment

Name: Vishal Prakash
UFID: 43884150
Email ID: vishalprakash@ufl.edu

Project Overview

The GatorGlide Delivery Co. Management System is a sophisticated software solution designed to revolutionize the way delivery services operate. By leveraging advanced AVL trees, the system optimally manages order priorities and delivery times, ensuring maximum efficiency in the delivery process. This technology-forward approach not only streamlines order handling but also significantly enhances delivery performance, setting a new standard for logistics operations.

Instructions to run the code

- Extract the contents of "Prakash_Vishal.zip" and navigate to the project directory.
- Open a terminal or command prompt in this directory.
- Paste the input text file in this directory .
- To run the program, in the terminal, type the command shown below:
  python gatorDelivery.py <input_file_name>.txt
  Eg: python gatorDelivery.py test1.txt
- The output will be generated in a file named <input_file_name_output_file.txt>, detailing the order deliveries and system operations.

Key Features

- **AVL Tree for Order Management:** Utilizes AVL trees for storing and managing orders based on their priorities, ensuring efficient access and modification operations.
- **Dynamic Priority Calculation:** Employs a formula that balances order value and creation time to calculate priorities, facilitating fair and efficient order processing.
- **Efficient Delivery Scheduling:** The system calculates and updates the Estimated Time of Arrival (ETA) for orders, dynamically adjusting to new orders and cancellations.
- **Single Delivery Agent Model:** Simulates a delivery system with a single agent, focusing on optimizing the delivery route and schedule for peak efficiency.
- **Comprehensive System Operations:** Supports creating, updating, and cancelling orders, along with sophisticated querying features like printing order details and determining order delivery ranks.

Functional Prototypes and Structure

The system is modularly designed, comprising several interconnected components:

1. **avl.py - AVL Tree Implementation**

   This module implements an AVL tree, a self-balancing binary search tree. Each node in the tree represents an order, keyed by its priority or ETA for efficient retrieval and updates. The AVL tree guarantees O(log n) complexity for insertion, deletion, and search operations, essential for maintaining system performance even as the number of orders grows.

   Key Methods:

   a) insert(self, key, value):

   Purpose: Inserts a new node with the given key and value into the AVL tree, maintaining the balanced property.
   Parameters:
   key: The priority or ETA of the order.
   value: The order details.
   Returns: None.

   b) delete(self, key):

   Purpose: Removes the node with the specified key from the tree, rebalancing it if necessary.
   Parameters:
   key: The priority or ETA of the order to be removed.
   Returns: None.

   c) search(self, key):

   Purpose: Finds and returns the node with the given key, if it exists.
   Parameters:
   key: The priority or ETA to search for.
   Returns: The node with the specified key or None if not found.

2. **gatorDelivery.py - Main Program**

   Serves as the entry point for the application. It parses the input file, orchestrates the execution of system operations based on commands within the file, and handles output file generation.

Main Functionality:

- Parses command-line arguments to identify the input file.
- Reads commands from the input file, executing operations such as order creation, cancellation, and updating.
- Manages output file generation, recording the results of operations and the state of the delivery schedule.

3. **order_management_system.py - Order Management**

   Central to handling the logic for managing orders, calculating priorities, updating ETAs, and managing the delivery schedule.

   Key Functions:

   a) createOrder(self, order_id, current_system_time, orderValue, deliveryTime):

      Purpose: Creates a new order with the given details, calculates its priority, inserts it into the priority AVL tree, and updates ETAs as necessary.
      Parameters: Order details including ID, system time, value, and delivery time.

   b) cancelOrder(self, order_id, current_system_time):

      Purpose: Cancels an existing order, removing it from the AVL tree and updating ETAs for remaining orders.
      Parameters: The ID of the order to cancel and the current system time.

   c) updateTime(self, order_id, current_system_time, newDeliveryTime):

      Purpose: Updates the delivery time for an existing order and adjusts ETAs accordingly.
      Parameters: Order ID, current system time, and the new delivery time.

4. **priority_queue.py - (If Implemented) Priority Queue for Pre-Processing Orders**

   While not explicitly detailed in the project description, a priority queue could be used for preprocessing or batch handling of orders before they are managed within the AVL trees, especially if orders are received in quick succession or in batches.

   Potential Methods:

   - enqueue(self, order): Adds an order to the queue based on its priority.
   - dequeue(self): Removes and returns the highest priority order from the queue.