README.md

Gator_Delivery_System

Author

- Dong-You Jhong
- UFID: 36484188
- jhong.dongyou@ufl.edu

AVL_unit.py

This file contains the implementation of an AVL tree data structure for managing orders. The AVL tree supports insertion, deletion, and various operations related to orders.

TreeNode class

- Represents a generic tree node with attributes for an order, including order ID, current system time, order value, delivery time, ETA (Estimated Time of Arrival), priority, left child, right child, and height.

AVL_Tree class

- Represents the AVL tree data structure with methods for insertion, deletion, rotation, and various order-related operations.

- **insert:** Inserts a new order into the AVL tree while maintaining balance.
- **leftRotate and rightRotate:** Perform left and right rotations to maintain balance in the AVL tree.
- **getHeight:** Calculates the height of a given node in the AVL tree.
- **getBalance:** Calculates the balance factor of a given node in the AVL tree.
- **getMinValueNode:** Retrieves the node with the minimum value in a subtree rooted at the given node.
- **preOrder and inOrder:** Perform pre-order and in-order traversals of the AVL tree.
- **get_path:** Retrieves the path from a node to the root in the AVL tree.
- **get_near_large_node:** Finds the nearest larger node in the AVL tree for a given node.
- **printHelper:** Helper function to print the AVL tree structure.

gatorDelivery.py

This file contains the main program logic for managing orders using the AVL tree data structure.

Order class

- Inherits from the TreeNode class and represents an order with additional functionalities.

OrderManagementSystem class

- Inherits from the AVL_Tree class and extends it to manage orders efficiently.

- Contains methods for creating orders, canceling orders, updating order delivery times, retrieving order ranks, printing orders within a specific time frame, and more.
- **create_order:** Creates a new order, inserts it into the AVL tree, and calculates its ETA.
- **cancel_order:** Cancels an existing order and updates the ETA of subsequent orders if necessary.
- **update_time:** Updates the delivery time of an existing order and adjusts the ETA of subsequent orders accordingly.
- **get_rank_of_order:** Retrieves the rank of a specific order based on its position in the AVL tree.
- **print_within_time:** Prints orders that will be delivered within a specified time frame.
- **quit:** Prints the list of orders that have been delivered and exits the program.
- **delete:** Deletes an order from the AVL tree while maintaining balance.

Usage

To use the program, execute the `gatorDelivery.py` script with a text file containing commands as input. The program parses the commands and performs the corresponding operations on the orders. Example commands include creating orders, canceling orders, updating delivery times, and printing orders within specific time frames.

```
python3 gatorDelivery.py test.txt
```

Command Format

Each line in the input text file represents a command to be executed. The commands follow a specific format:

- **Create Order:** `createOrder(order_id, current_system_time, order_value, delivery_time)`
- **Cancel Order:** `cancelOrder(order_id, current_system_time)`
- **Update Time:** `updateTime(order_id, current_system_time, new_delivery_time)`
- **Print Orders within Time Frame:** `print(time1, time2)`
- **Print Order Rank:** `getRankOfOrder(order_id)`
- **Quit:** `quit()`

Example Input File

```
createOrder(1001, 1, 100, 4)
createOrder(1002, 2, 150, 7)
print(2, 15)
```

Output

The program generates output based on the commands provided in the input file. The output includes information about the created, canceled, and delivered orders, as well as any updates to their ETA.

Dependencies

- Python 3.x
- argparse
- re
- sys