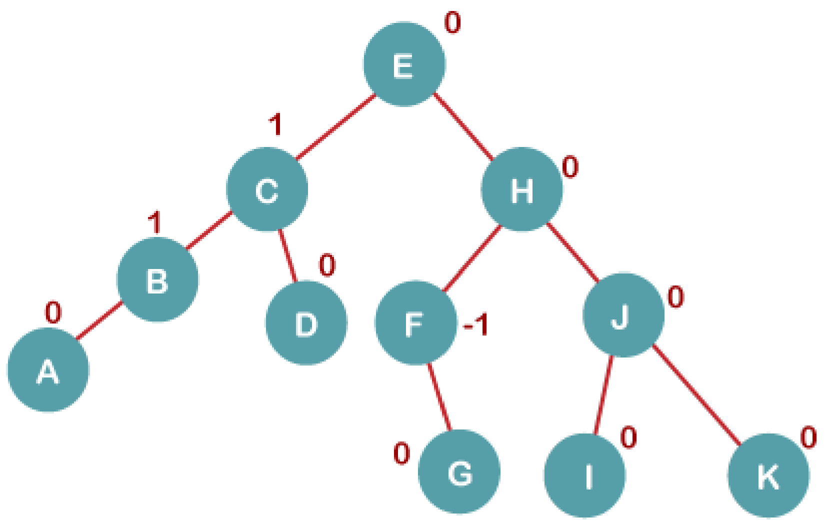

(Balanced)

GatorGlide Delivery co.

Name : Hari Krishna Reddy

ID : 79915077

UF Email : golamari.h@ufl.edu

Project Details

The **GatorGlide delivery management system**, an optimized order delivery management system, is represented by this project and is intended to manage the different tasks involved in creating orders, updating times, cancelling and prioritizing orders as the priority metrics. The main elements and features of the system are described in detail in this report.

The system is structured in such a way that each class serves a specific purpose in the overall delivery management process. The main classes are **AVLTree**, **Order**, **OrderManagementSystem**.

Key Components

1. Priority Tree

The ETA (Estimated Time of Arrival) for an order is calculated based on the ETA of the order that needs to be delivered immediately before it, along with the delivery time of the current order and previous order(The driver return time is added to ETA). This calculation uses this AVL tree with order priorities as keys and order numbers as values

2. Eta Tree

This process involves delivering and removing all orders with an Estimated Time of Arrival (ETA) less than or equal to the order creation time. To efficiently handle this, This AVL tree is utilized with order numbers as keys. When an order is canceled, it is removed from the system, and the ETAs of all orders with lower priority are updated

3. Order Management system

Ensures the effective management of Order delivery and prioritization using the AVL trees

Key Methods:

- ❖ func_check_order_deliveries
- ❖ func_update_eta
- ❖ func_create_order
- ❖ func_cancel_order
- ❖ func_get_rak_of_order
- ❖ func_deliver_remainig_orders

4. Output

The system is intended to be run with a specific input file **(.txt file)** containing a series of commands. Each command corresponds to a order operation like creation of order, update, delivering…. The results are output to a file with filename as **"InputFile Name_OutputFileName.txt"**

Execution of Program

1. Unzip the zip file with name Golamari_harikrishnareddy.zip
2. Navigate to the folder and run any of the following commands in the terminal
   - ❖ python3 gatorDelivery.py <InputFileName.txt>
3. The output of the program will be stored in a file with name "InputFileName_OutputFileName.txt"

Function Prototype/Explanation of Code

As we have seen above the whole code is majorly based on the implementation of 5 classes which are:

- ➔ class TreeNode
- ➔ class AVLTree
- ➔ class Ordersystem

TreeNode Class

- Purpose: Represents a node in the AVL tree.
- Attributes:
  - key: The key associated with the node.
  - val: The value associated with the key.
  - left: Reference to the left child.
  - right: Reference to the right child.
  - height: The height of the node.

AVLTree Class

- ➢ Initialization:
  - __init__(): Initializes an empty AVL tree.
- ➢ Insertion:
  - insert(root, key, val): Inserts a key-value pair into the AVL tree.
    - Time Complexity: O(log n), where n is the number of nodes in the tree.

- Deletion:
    - delete(root, key): Deletes a key from the AVL tree.
        - Time Complexity: O(log n), where n is the number of nodes in the tree.
- Rotation Operations:
    - leftRotate(z): Performs a left rotation around node z.
    - rightRotate(y): Performs a right rotation around node y.
- **Utility Functions:**
    - getHeight(root): Returns the height of a node.
    - getBalance(root): Returns the balance factor of a node.
    - getMinValueNode(root): Finds the node with the minimum key value.
    - preOrder(root): Performs a pre-order traversal of the AVL tree.
    - inOrder(root): Performs an in-order traversal of the AVL tree and returns the sorted key-value pairs.
    - getSortedItems(): Returns all key-value pairs in sorted order.
    - getNode(root, key): Retrieves a value by its key.
    - update(root, key, new_val): Updates the value of a key.
    - reverseInOrder(root): Performs a reverse in-order traversal of the AVL tree and returns the key-value pairs in reverse sorted order.
    - getReverseSortedItems(): Returns all key-value pairs in reverse sorted order.
    - countNodes(root): Counts the number of nodes in the AVL tree.
    - getNumberOfNodes(): Returns the total number of nodes in the AVL tree.
- **Properties**
- Balanced: Ensures that the tree remains balanced after every insertion and deletion operation.
- Search Time Complexity: O(log n), where n is the number of nodes in the tree.
- Insertion and Deletion Time Complexity: O(log n), due to the self-balancing nature of the AVL tree.

Class: Ordersystem

The OrderSystem class represents an order management system that uses AVL trees to organize orders based on their priority and estimated time of arrival (ETA).

- **Initialization**:
  __init__(file): Initializes the order management system with two AVL trees for order priorities and orders with meta information. It also sets the current system time, first order flag, driver return time, and last order ETA.
- **Order Delivery Status**:
  func_check_order_deliveries(): Checks and updates the status of order deliveries

based on the current system time and updates the AVL trees accordingly.

- **ETA Update**:

  func_update_eta(order_id): Updates the ETA for all orders available before the driver returns from the last delivery. It also prints the ETA for a specific order and any updated ETAs for other orders.
- **Order Creation:**

  func_create_order(order_id, creation_time, order_value, delivery_time): Creates a new order and updates the AVL trees accordingly. It also checks whether the driver has returned from the last delivery and pushes the new order out for delivery accordingly.
- **Order Cancellation:**

  func_cancel_order(order_id, current_system_time): Cancels an existing order only if it's not out for delivery. It checks for the out_for_delivery field and relays messages accordingly. It also updates the ETAs if necessary.
- **Delivery Time Update**:

  func_update_time(order_id, current_system_time, new_delivery_time): Updates the delivery time of an existing order and updates the AVL trees accordingly, only if the order is not out for delivery. It also updates the ETAs if necessary.
- **Printing Orders in Time Range**:

  func_double_print(time1, time2): Prints the orders within the specified time range that will be delivered.
- **Printing ETA**:

  func_print_eta(): Prints the ETA for each item in the AVL tree. The printing is conditional based on a debug flag.
- **Printing Single Order Details**:

  func_single_print(order_id): Prints the details of a single order.
- **Getting Order Rank**:

  func_get_rak_of_order(order_id): Gets the rank of an order in the AVL tree for the given order ID.
- **Delivering Remaining Orders**:

  func_deliver_remainig_orders(): Delivers all the remaining orders in the AVL tree once the program receives a quit command.