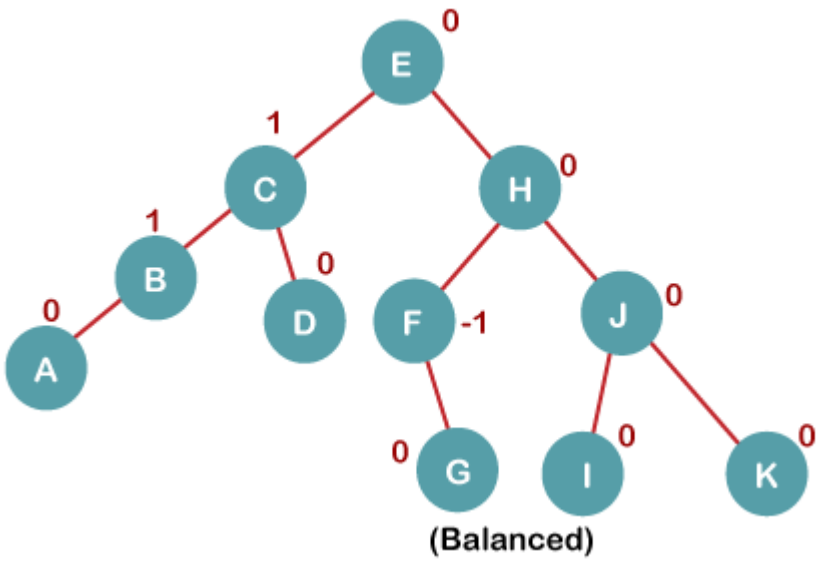

GatorGlide Delivery co.

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. This calculation uses this AVL tree with order priorities as keys, allowing for duplicate keys.

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 ETAs as keys (without duplicates). 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:

- Create_order
- Update_time
- Deliver_orders
- Cancle_order
- Calculate_eta
- Update_eta

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 library operation like adding, borrowing books, etc. The results are output to a file with filename as **"InputFile Name_OutputFileName.txt"**

Execution of Program

1. Unzip the zip file with name Hari_golamari.zip
2. Navigate to the folder and run any of the following commands in the terminal
   - ❖ python3 order.py <InputFileName.txt> <outputFileName.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 AVLNode
- ➔ class AVLTree
- ➔ class Order
- ➔ class OrderManagementSystem

Class: AVLNode

- **__init__(self, key, value)**: Initializes an AVL tree node with a key and value pair, setting its left and right children to **None**, and its height to 1.

Class: AVLTree

- **__init__(self)**: Initializes the root of the AVL tree as **None**, indicating an empty tree.
- **insert(self, key, value)**: (Time complexity: O(log n)) Inserts a node with a given key and value into the AVL tree and balances the tree if necessary.
- **_insert(self, node, key, value)**: (Time complexity: O(log n)) Recursively inserts a node into the AVL tree at the correct position and balances the tree.
- **delete(self, key)**: (Time complexity: O(log n)) Deletes a node with a given key from the AVL tree and balances the tree if necessary.
- **_delete(self, node, key)**: (Time complexity: O(log n)) Recursively deletes a node from the AVL tree and balances the tree.
- **search(self, key)**: (Time complexity: O(log n)) Searches for a node with a given

key in the AVL tree and returns the node if found.

- **_search(self, node, key)**: (Time complexity: O(log n)) Recursively searches for a node in the AVL tree based on the key.
- **update_height(self, node)**: Updates the height of a given node based on the heights of its children.
- **balance(self, node)**: Balances the AVL tree by performing appropriate rotations on the given node.
- **get_height(self, node)**: Returns the height of a given node.
- **get_balance(self, node)**: Returns the balance factor of a given node.
- **left_rotate(self, z)**: (Time complexity: O(1)) Performs a left rotation on the given node **z** to rebalance the tree.
- **right_rotate(self, y)**: (Time complexity: O(1)) Performs a right rotation on the given node **y** to rebalance the tree.
- **min_value_node(self, node)**: (Time complexity: O(log n)) Returns the node with the minimum key value in the subtree rooted at the given node.
- **inorder_traversal(self, node, result)**: Performs an inorder traversal of the AVL tree starting from the given node and appends the values of the nodes to the **result** list.
- **get_inorder_values(self)**: Returns a list of values of the nodes in the AVL tree in inorder.

Class: Order

- **__init__(self, orderId, currentSystemTime, orderValue, deliveryTime)**: Initializes an order with an order ID, current system time, order value, delivery time, calculates its priority, and sets its ETA to **None**.
- **calculate_priority(self)**: Calculates the priority of the order based on its value and the current system time.

Class: OrderManagementSystem

- **__init__(self)**: Initializes the order management system with two AVL trees for managing orders by priority and ETA, and a dictionary to store orders by their IDs.
- **create_order(self, orderId, currentSystemTime, orderValue, deliveryTime)**: (Time complexity: O(log n)) Creates a new order, calculates its ETA, inserts it into the priority and ETA AVL trees, and adds it to the orders dictionary.
- **update_time(self, orderId, currentSystemTime, newDeliveryTime)**: (Time complexity: O(log n)) Updates the delivery time and ETA of an existing order and adjusts the AVL trees accordingly.

- **deliver_orders(self, currentSystemTime)**: (Time complexity: O(k log n), k is number of orders) Delivers orders whose ETAs are less than or equal to the current system time, removing them from the AVL trees and the orders dictionary.
- **cancel_order(self, orderId, currentSystemTime)**: (Time complexity: O(log n)) Cancels an existing order, removes it from the AVL trees and the orders dictionary, and updates the ETAs of remaining orders.
- **search_order(self, orderId)**: Searches for an order by its ID and prints its details if found. If the order is not found, it prints a message indicating so.
- **calculate_eta(self, order)**: Calculates the ETA of an order based on the current system time and the last delivered order time.
- **update_etas(self)**: Updates the ETAs of all orders in the system to reflect changes in the delivery schedule.
- **print_orders_in_range(self, time1, time2)**: Prints the IDs of orders whose ETAs are within a specified time range.
- **get_rank_of_order(self, orderId)**: Prints the rank of an order based on its ETA compared to other orders in the system.