**Report for ADS project: gatorDelivery**

Zhiyu Wang

04.02.2024

UFID: 31397456 Email: [wangzhiyu@ufl.edu](mailto:wangzhiyu@ufl.edu)

**Overview**

This project was written using Java language and consistent with four separate files including AvlTree, AvlNode, Order and gatorDelivery. The goal is to using AVL tree structure to realize a order management system with function print(orderId), print(time1, time2), getRankOfOrder(orderId), createOrder(order\_id, current\_system\_time, orderValue, deliveryTime), cancelOrder(order\_id, current\_system\_time) and updateTime(order\_id, current\_system\_time, newDeliveryTime). The function, structure and time complexity for each file and function will be introduced below.

**Function, structure and time complexity**

* **Order.java**

----Function

The Order class itself does not use any specific data structure internally. It simply defines the structure of an order object with its attributes (orderId, createTime, orderValue, deliveryTime, ETA) and provides methods to access and modify these attributes.

orderId: An integer representing the order ID.

createTime: An integer representing the time when the order was created.

orderValue: An integer representing the value of the order.

deliveryTime: An integer representing the time it takes to deliver the order.

ETA: An integer representing the estimated time of arrival for the order.

----Time complexity:

public Order(int orderId, int currentSystemTime, int orderValue, int deliveryTime): This constructor initializes the orderId, createTime, orderValue, and deliveryTime variables. It has a time complexity of O(1) as it performs a constant number of operations to initialize the object.

public double getPriority(): This method calculates and returns the priority of the order based on its orderValue and createTime. It has a time complexity of O(1) as it performs a constant number of arithmetic operations to calculate the priority.

public int getOrderId(): This method returns the orderId of the order. It has a time complexity of O(1) as it simply returns the value of a variable.

public int getOrderValue(): This method returns the orderValue of the order. It has a time complexity of O(1) as it simply returns the value of a variable.

public int getDeliveryTime(): This method returns the deliveryTime of the order. It has a time complexity of O(1) as it simply returns the value of a variable.

public int getETA(): This method returns the ETA of the order. It has a time complexity of O(1) as it simply returns the value of a variable.

public int getCreateTime(): This method returns the createTime of the order. It has a time complexity of O(1) as it simply returns the value of a variable.

Setters (setOrderId, setOrderValue, setDeliveryTime, setETA, setCreateTime): These setter methods set the values of the respective instance variables. They have a time complexity of O(1) as they simply assign values to variables.

* **AvlNode.java and AvlTree.java**

These two classes, AvlTree and AvlNode, work together to implement an AVL tree, a self-balancing binary search tree.

-----Function

The AvlNode class: represents a node in the AVL tree. It contains fields for the element value, the height of the node, and references to the left and right child nodes.

Constructor: The no-argument constructor initializes the node with the given element and sets the height to 0. The full-argument constructor initializes the node with the given element, left child, and right child.

The AvlTree class represents the AVL tree data structure. It contains a reference to the root node and a comparator for comparing elements if provided.

Constructor: The default constructor initializes an empty AVL tree. The constructor with a comparator parameter allows custom comparison logic for elements.

----Time complexity:

insert(T x) method:

Inserts a new element x into the AVL tree while maintaining balance.

Time Complexity: O(log n) average case, O(n) worst case (when the tree becomes unbalanced and requires rotations).

remove(T x) method:

Removes an element x from the AVL tree while maintaining balance.

Time Complexity: O(log n) average case, O(n) worst case (when the tree becomes unbalanced and requires rotations).

tolist() method:

Returns a list of elements in the AVL tree using in-order traversal (sorted order).

Time Complexity: O(n) (visits each node once).

findMin() method:

Finds and returns the minimum element in the AVL tree.

Time Complexity: O(log n) average case, O(n) worst case (traverse left child until reaching the leaf).

findMax() method:

Finds and returns the maximum element in the AVL tree.

Time Complexity: O(log n) average case, O(n) worst case (traverse right child until reaching the leaf).

makeEmpty() method:

Empties the AVL tree by setting the root to null.

Time Complexity: O(1).

isEmpty() method:

Checks if the AVL tree is empty.

Time Complexity: O(1).

balance(AvlNode<T> t) method:

Balances the subtree rooted at node t to maintain AVL tree property.

Time Complexity: O(1) for each node (constant time operations).

Rotation methods (rotateWithLeftChild, rotateWithRightChild, doubleWithLeftChild, doubleWithRightChild):

Perform single or double rotations to balance the tree.

Time Complexity: O(1) for each rotation operation.

* **gatorDelivery**

The gatorDelivery class represents a program for managing delivery orders using an AVL tree data structure. It contains methods for creating, canceling, updating, and retrieving information about orders. Below is the analysis of its functionalities and the time complexity of each method:

Main Variables:

priorityTree: An AVL tree that stores orders based on their priority (highest priority first).

etaPriorityTree: An AVL tree that stores orders based on their estimated time of arrival (ETA).

canceled: A set that logs canceled order IDs.

delivered: A set that logs delivered order IDs.

deliveringOrder: An order object representing the current order being delivered.

preDeliveringOrderEndTime: The end time of the previous delivering order.

Main Method (main):

Reads input from a file, executes commands, and writes the output to another file.

Command Execution Method (executeLine):

Parses a command from a string and uses reflection to call the corresponding method.

Time Complexity: O(n) where n is the number of characters in the input command.

Order Management Methods:

print(int orderId): Retrieves order details by order ID.

print(int time1, int time2): Retrieves order IDs for orders with ETAs within a given time range.

getRankOfOrder(int orderId): Determines the rank of an order based on its ETA compared to other orders.

createOrder(int orderId, int currentSystemTime, int orderValue, int deliveryTime): Creates a new order, updates ETAs of other orders, and manages the delivering order.

cancelOrder(int orderId, int currentSystemTime): Cancels an order, updates ETAs of other orders, and manages the delivering order.

updateTime(int orderId, int currentSystemTime, int newDeliveryTime): Updates the delivery time of an order, updates ETAs of other orders, and manages the delivering order.

Quit(): Outputs the delivered orders and ends the program.

**----Time Complexity**

The time complexity of each method depends on the operations performed and the size of the AVL trees.

Insertion, deletion, and searching in AVL trees have an average time complexity of O(log n) and a worst-case time complexity of O(n) if the tree becomes unbalanced.

Traversing the trees to convert them to lists for printing has a time complexity of O(n) where n is the number of nodes in the tree.

Iterating over lists for searching or updating has a time complexity of O(n) where n is the number of elements in the list.

Overall, the time complexity of the program is dominated by the AVL tree operations, making it efficient for managing orders.