Programming Project Report

COP 5536 Spring 2024

Advanced Data Structures

GatorGlide Delivery Co.

Name : Sri Meghana Vaishnapu
UFID : 15258339
Email : Vaishnapu.s@ufl.edu

Abstract:

GatorDelivery is a delivery management system designed to efficiently handle orders, prioritize deliveries, and estimate delivery times. Leveraging data structure AVL trees, GatorDelivery optimizes order processing, enabling businesses to streamline their delivery operations and enhance delivery management, serving on the basis of priority and ordering time. The key functionalities include, creating the order, Get the rank of the order (which gives the rank of at which the order will be delivered), update order time, cancel order, fetch the orders between any given time period. This report provides an overview of the GatorDelivery system, outlining its design, key features, implementation details, and performance evaluation.

Introduction:

The efficient delivery management systems are indispensable for businesses aiming to provide timely and reliable services to their customers. GatorDelivery is a sophisticated solution tailored to meet the demands of modern delivery operations. By integrating advanced data structures and algorithms, GatorDelivery optimizes order processing, prioritization, and delivery time estimation, thereby enhancing operational efficiency and customer experience.

This report serves as a comprehensive guide to the GatorDelivery system, offering insights into its architecture, functionality, and implementation. We delve into the core components of the system, including AVL trees for order prioritization and ETA tree for estimating delivery times within specified ranges. Through detailed explanations and examples, we illustrate how GatorDelivery empowers businesses to manage their delivery logistics with precision and agility.

Furthermore, we discuss the design principles underlying GatorDelivery. By employing modular design practices and leveraging object-oriented programming concepts, GatorDelivery ensures adaptability to evolving business requirements and technological landscapes.

In summary, this report offers a comprehensive overview of GatorDelivery, shedding light on its design philosophy, functionality, and performance characteristics. By embracing innovation and efficiency, GatorDelivery stands as a testament to the transformative power of technology in the realm of delivery management.

Problem Description:

In the heart of Florida, GatorGlide Delivery Co. emerged as a tech-savvy beacon of efficient logistics inspired by the swift movements of the alligator. It is planning to elevate its software infrastructure to meet the growing demands of customers. The new system aims to optimize order management, delivery routes, and enhance overall efficiency. They use AVL trees to intricately store the order priorities and ETAs to manage the orders efficiently.
Order management system:
● Order details are given whenever an order is created with the following fields: orderId, currentSystemTime, orderValue, deliveryTime
● We can perform operations like createOrder, cancelOrder, and searchOrder
● currentSystemTimes are relative to a fixed value. So, they can only be some integer values, representing time since an epoch (like 1, 2, 3, etc.).
● deliveryTime is the time required to process and deliver the order (from source to destination).
Orders are served according to their priorities. Priority of an order is calculated using the following equation:
priority = valueWeight * normalizedOrderValue - timeWeight * orderCreationTime
where valueWeight = 0.3, timeWeight = 0.7, and normalizedOrderValue = order.orderValue / 50

System Architecture:

The Program contains five classes:

a. Order.java
b. AVLTree.java
c. ETATree.java
d. gatorDelivery.java

1. **Order class**

   The ‘Order’ class represents an order within the Gator Delivery system. It encapsulates essential attributes such as the order ID, current system time, order value, and delivery time. Additionally, it provides methods to

calculate the priority of the order based on its value and current system time.

Function Prototypes and Explanations:

1. Constructor
   - public Order(int orderId, int currentSystemTime, int orderValue, int deliveryTime)
     - Initializes a new instance of the ‘order’ class with the specified attributes.

2. Method: calculatePriority()
   - public double calculatePriority()
     - Calculates the priority of the order based on its value and current system time, using a weighted formula.
   - Returns:The calculated priority of the order.

The ‘Order’ class forms a fundamental component of the Gator Delivery system, facilitating the representation and management of individual orders. Its methods allow for the determination of order priorities, aiding in efficient order processing and delivery scheduling.

2. AVLTree class

The AVLTree class represents a self-balancing binary search tree specifically designed for managing orders within the Gator Delivery system. It employs AVL (Adelson-Velsky and Landis) tree properties to ensure balanced and efficient insertion, deletion, and retrieval operations.

Function Prototypes and Explanations:

1. Node Class:
   - static class Node
     - Represents a node in the AVLTree, containing an `Order` object and references to left and right child nodes.
   - Attributes:

- Order order: The order stored in the node.
- Node left: Reference to the left child node.
- Node right: Reference to the right child node.
- int height: Height of the node in the tree.
- Constructor:
- Node(Order order): Constructs a new node with the specified order.

2. Height Calculation:
- int height(Node node)
- Calculates the height of a given node in the tree.
- Parameters: Node node - The node for which height is to be calculated.
- Returns: The height of the node.

3. Balance Factor Calculation:
- int balance(Node node)
- Calculates the balance factor of a given node in the tree.
- Parameters: Node node - The node for which the balance factor is to be calculated.
- Returns: The balance factor of the node.

4. Rotations:
- Node rightRotate(Node y) and Node leftRotate(Node x)
- Perform right and left rotations, respectively, on the given node.
- Parameters: Node y (or Node x) - The node to be rotated.
- Returns: The root node after rotation.

5. Insertion:
- Node insert(Node node, Order order)
- Inserts a new node with the given order into the AVL tree while maintaining balance.
- Parameters:
- Node node - The root node of the subtree.
- Order order - The order to be inserted.
- Returns: The root node of the modified subtree after insertion.

The AVLTree class serves as a critical component in the Gator Delivery system's order management module, ensuring efficient storage and retrieval of orders while adhering to AVL tree properties. Its methods enable seamless integration of orders into the tree structure, promoting optimal performance and scalability.

3. ETATree class

The ETATree class serves as a specialized tree structure within the Gator Delivery system, primarily designed for managing Estimated Time of Arrival (ETA) values associated with orders. It facilitates efficient association between ETAs and corresponding nodes in the AVLTree, enabling streamlined retrieval and management of order delivery timings.

Function Prototypes and Explanations:

1. Node Class:
   - static class Node
     - Represents a node in the ETATree, containing an ETA value and a reference to the corresponding AVLTree node.
     - Attributes:
       - int eta: The Estimated Time of Arrival value.
       - AVLTree.Node avlNode: Reference to the corresponding node in the AVLTree.
       - Node left: Reference to the left child node.
       - Node right: Reference to the right child node.
       - int height: Height of the node in the tree.
     - Constructor:
       - Node(int eta, AVLTree.Node avlNode): Constructs a new node with the specified ETA value and AVLTree node reference.

2. Height Calculation:
   - int height(Node node)
     - Calculates the height of a given node in the tree.

- Parameters: Node node - The node for which height is to be calculated.
- Returns: The height of the node.

3. Balance Factor Calculation:
- int balance(Node node)
- Calculates the balance factor of a given node in the tree.
- Parameters: Node node - The node for which the balance factor is to be calculated.
- Returns: The balance factor of the node.

4. Rotations:
- Node rightRotate(Node y) and Node leftRotate(Node x)
- Perform right and left rotations, respectively, on the given node.
- Parameters: Node y (or Node x) - The node to be rotated.
- Returns: The new root node after rotation.

5. Insertion:
- Node insert(Node node, int eta, AVLTree.Node avlNode)
- Inserts a new node with the given ETA and AVLTree node references into the ETATree while maintaining balance.
- Parameters:
- Node node - The root node of the subtree.
- int eta - The ETA value to be inserted.
- AVLTree.Node avlNode - The corresponding AVLTree node reference.
- Returns: The root node of the modified subtree after insertion.

4. gatorDelivery class

The code provided seems to be a Java implementation of a delivery management system named GatorDelivery. Here's a breakdown of its key components and functionality:

1. Data Structures Used:
- AVL Tree: Used for order prioritization based on certain criteria.

- ETA Tree: Utilized for efficient retrieval of orders within specified time ranges.

2. Main Class: The GatorDelivery class serves as the main entry point and orchestrator of the delivery management system.

3. Input Processing:
- Reads commands from an input file, processes them, and writes output to another file.
- Commands include operations like creating orders, canceling orders, updating delivery times, printing order details, etc.

4. Order Management:
- Orders are represented as objects containing attributes such as order ID, current system time, order value, and delivery time.
- Orders are inserted, updated, or canceled in both the AVL and ETA trees as per the specified commands.
- ETA calculations are performed based on the current system time and delivery times of orders, considering their priority.

5. Priority Handling:
- Orders are prioritized using AVL trees, allowing for efficient retrieval of orders based on priority.
- The system dynamically adjusts priorities when orders are canceled or updated.

6. ETA Calculation and Management:
- ETAs are calculated and managed using the ETA tree data structure.
- The system ensures that ETAs are updated accurately when orders are created, canceled, or delivery times are modified.

7. Output Generation:
- Outputs generated include order details, ETA updates, ranks of orders, etc.
- Outputs are written to a designated output file.

8. Error Handling:

- The system handles invalid commands and ensures robust error reporting for scenarios like order not found, delivery already completed, etc.

9. Debugging and Logging:
- Debugging statements are included for tracking AVL tree structures, order search operations, etc.
- Methods like printAVLTreeStructure, size, and minValueNode aid in debugging and analyzing AVL tree operations.

10. File I/O:
- The system reads commands from an input file and writes outputs to another file.

Functions and explanation:

1. main(String[] args):
- Entry point of the program.
- Reads input from a file, processes commands, and writes output to another file.

2. processCommand(String command):
- Parses and executes commands read from the input file.
- Routes commands to specific methods based on the command type.

3. print(int orderId):
- Prints the details of a specific order.
- Calculates the Estimated Time of Arrival (ETA) for the order and prints the details if the order exists.

4. print(int time1, int time2):
- Prints orders within a specified time range.
- Traverses the ETA tree to find orders within the given time range and prints their IDs.

5. printOrdersWithinRange(ETATree.Node node, int time1, int time2, List<Integer> ordersWithinRange):

- Helper method for printing orders within a time range.
- Recursively traverses the ETA tree to find orders within the specified time range.

6. getRankOfOrder(int orderId):
- Retrieves the rank of a specific order based on its priority in the AVL tree.
- Prints the order's rank if found.

7. getRankOfOrder(AVLTree.Node node, int orderId):
- Helper method for retrieving the rank of an order.
- Recursively traverses the AVL tree to find the rank of the order.

8. size(AVLTree.Node node):
- Calculates the size of the AVL tree rooted at the given node.
- Used for debugging purposes.

9. printAVLTreeStructure(AVLTree.Node node):
- Prints the structure of the AVL tree.
- Used for debugging AVL tree operations.

10. createOrder(int orderId, int currentSystemTime, int orderValue, int deliveryTime):
- Creates a new order and inserts it into the AVL and ETA trees.
- Updates the ETA of affected orders if necessary.

Conclusion:

In conclusion, the GatorDelivery project demonstrates the implementation of a delivery management system using AVL trees for order prioritization and ETA trees for efficient retrieval of orders within specified time ranges. The system efficiently processes commands such as creating orders, canceling orders, updating delivery times, retrieving order ranks, and printing order details.

By leveraging AVL trees, the system ensures that orders are prioritized based on their values and delivery times, facilitating efficient delivery management. The ETA trees enable quick retrieval of orders within specified time intervals, enhancing the system's responsiveness to customer queries and requests.