NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
(KARACHI CAMPUS)
FAST School of Computing
Fall 2023

DATA STRUCTURES PROJECT REPORT

Online E-commerce Store

Group Members:
Laiba Mohsin - 22K-4246
Aniqa Azhar - 22K-4228
Aisha Jalil - 22K-4649

Section - CS-3K
Lecturer : Sobia Iftikhar

Introduction

Our project aims to implement an efficient order management and delivery system for an online shop. The project utilizes various data structure concepts to handle order processing, deliveries, and optimize the overall system's efficiency. The project will use data structures and algorithms to manage the store's inventory and process orders efficiently, while also providing customers with a user-friendly interface.

Problem Statement

The primary problem addressed by the project is to efficiently manage orders and deliveries for an online shop. This involves creating a robust system that handles customer orders, calculates bills, and optimizes delivery routes based on geographical locations. This project highlights how different data structures and algorithms are practically used to create an online store management system. It focuses on developing efficient solutions for tasks like storing data, retrieving information, and finding optimal paths within the system.

Data Structure Concepts Used

1. AVL Tree

The AVLTreeforOrders Class is implemented to manage takeaway orders using an AVL tree, ensuring balance after each insertion or deletion to maintain its height. This AVL tree plays a crucial role in efficiently handling and organizing takeaway orders for each shop. The tree's balancing mechanism facilitates quick retrieval and insertion of orders, enhancing the overall efficiency of managing the store's inventory. Operations such as searching, deletion, and insertion are optimized using the AVL tree, with the order ID provided by the user serving as a key parameter for insertion and deletion processes.

2. Graphs (for Dijkstra's Algorithms)

Graphs are a key element in our project, visually depicting delivery areas and presented through an adjacency matrix. To enhance system navigation, we employ Dijkstra's algorithm. This algorithm excels in determining the shortest path between two nodes, with nodes representing cities and areas in the cities in our case. We utilize it to calculate distances between cities, optimizing our home delivery routes and enhancing the efficiency of the delivery process.

3. Linked List

As linked lists allow efficient insertion and deletion of elements, our project employs them to handle customer details and order information, storing user preferences (Delivery, Take Away) in the linked list. Linked lists are chosen over arrays due to their dynamic size, providing flexibility in managing and adapting to varying amounts of data.

4. Heaps

A min-heap is used to prioritize home delivery orders based on their urgency. The priority is determined by the priority attribute of the **DeliveryOrders** class. The **heapify_up** and **heapify_down** functions ensure that the min-heap properties are maintained during insertion and removal operations. Urgent orders, indicated by lower priority values, are moved to the front of the heap, ensuring that they are processed first. This allows for efficient retrieval of the highest-priority order when deliveries are being made.

5. Hash Table

The hash table is used to efficiently store and search home delivery orders based on their order ID. The insert method is used to insert a new delivery order into the hash table based on its order ID. The search method is used to search for a delivery order by its order ID. The delete_val method is used to delete a delivery order from the hash table based on its order ID. The hashFunction method calculates the hash key based on the order ID and the size of the hash table.

Efficiency Analysis

Time Complexity

The algorithm utilized in the project, such as Dijkstra's, is selected for their low time complexity.

Heap Operations (insert and remove function in HomeDeliveryHeap class, Min Heap for Dijkstra's):

The insertion operation and removal operation in a minheap (used as a priority queue) has a time complexity of O(n log n) for n orders. Single insertion or removal is O(log n).

The heapify_up and heapify_down functions utilized for a single insertion or removal have time complexity of O(log n), where n is the number of elements in the heap.

AVL Tree Operations (insert, deleteOrders, and tree traversal functions):
AVL tree insertion and deletion operations have a time complexity of O(log n) in the worst case, where n is the number of elements in the tree.
Tree traversal functions (e.g., preOrder) also has a time complexity of O(n), where n is the number of nodes in the tree/number of orders.

Dijkstra's Algorithm for Shortest Paths:
Dijkstra's algorithm has a time complexity of O((V + E) log V) with an adjacency matrix and min heap implementation, where V is the number of vertices and E is the number of edges.

Displaying Orders and Deliveries:
Printing each order or delivery: O(N) where N is the number of orders or deliveries.

Hash Table implementation:
The insert, search, and delete_val operations in this code have an average-case time complexity of O(1) because the hash function is used to directly access the location in the hash table.

Space Complexity

Home Delivery Heap (deliveries array in HomeDeliveryHeap class):
The space complexity of the home delivery queue is O(1) for the array itself, but O(n) for the dynamically allocated DeliveryOrders objects. Each DeliveryOrders object holds information about an order, and the number of such objects depends on the number of home delivery orders.

AVL Tree (nodes and related data**):**
The space complexity of the AVL tree is O(n), where n is the number of nodes in the tree. Each node in the AVL tree contains information about an order.

Min Heap (used in Dijkstra's):
The space complexity of the min heap is O(V) for the array of MinHeapNode objects.

Hash Table implementation:
The space complexity of the hash table is O(n), where n is the number of unique order IDs.

Comparison with Existing Systems

The project differentiates itself from existing systems through its emphasis on optimizing time and space complexity. By incorporating Dijkstra's and minheap, the system ensures faster and more efficient route planning, leading to timely deliveries. The minheap further enhances the project's capability to handle urgent orders with minimal delay .The project uses data structures and algorithms to manage the store's inventory and process orders efficiently, while also providing customers with a user-friendly interface. The system is designed to facilitate both in-store and home deliveries, providing features for efficient order processing and tracking. It incorporates geographical information, allowing users to select delivery areas within cities, and it calculates delivery distances and charges accordingly.

Conclusion

In conclusion, this project highlights the challenges of order management and delivery optimization using various data structure concepts. The implementation of AVL Trees, Priority Queues, and Graph algorithms results in an efficient and scalable system. The project's differentiation from existing systems lies in its commitment to minimizing time and space complexity, providing a competitive edge in the domain of order and delivery management.