Phase 2: Final Project

Avinna Bhattarai

005028547

MSCS-532-A01

Project Phase 2 Deliverable 2: Proof of Concept Implementation

Dr. Vanessa Ruth Cooper

09/23/2024

Phase 2: Final Project

**Partial Implementation Overview**

The project focuses on designing data structures to optimize various aspects of a supply chain system, such as inventory management, route optimization, and scheduling deliveries. The key data structures implemented include:

* **Graph**: Models the transportation network between suppliers, warehouses, and customers.
* **Priority Queue (Heap)**: Manages the order of shipments based on urgency or cost.
* **Hash Table**: Stores inventory data for fast lookups on stock levels.
* **Balanced Tree (AVL)**: Maintains a sorted list of inventory levels and costs, enabling efficient dynamic updates.

**Fit into Overall Design:**

* The **Graph** enables route optimization using algorithms like Dijkstra's for shortest path calculations.
* The **Priority Queue** ensures urgent or cost-effective shipments are prioritized in scheduling.
* The **Hash Table** provides real-time inventory tracking.
* The **Balanced Tree** efficiently manages sorted data, crucial for inventory levels and route costs that require frequent updates.

**Demonstration and Testing**

Several test cases were executed to validate the performance and functionality of the key operations in Python.

**Test Case 1:** Shortest Route Calculation (Graph)**A screen shot of a computer code

Description automatically generated**

**Result:** The graph correctly calculates the shortest routes from Supplier1 to both warehouses using the adjacency list structure.

**Test Case 2:** Shipment Priority (Priority Queue)A computer screen with green and white text

Description automatically generated

**Result:** The priority queue correctly processed the most urgent shipment first, demonstrating proper heap management.

**Test Case 3:** Inventory Lookup (Hash Table)A black screen with white text

Description automatically generated

**Result:** The hash table returned correct product quantities and efficiently handled non-existent entries.

**Implementation Challenges and Solutions**

**1. Scalability:**

* **Challenge:** As the size of the supply chain network increases, graph traversal algorithms such as Dijkstra’s become computationally expensive.
* **Solution:** For large datasets, optimization strategies such as implementing A\* search algorithms or parallelizing the graph traversal were considered for future scalability improvements.

**2. Real-Time Data Handling:**

* **Challenge:** Handling real-time updates to inventory levels and shipment routes requires consistent and efficient data management.
* **Solution:** Hash tables and AVL trees were used to support fast lookups and maintain balance during frequent updates. In future iterations, using distributed databases for real-time data consistency across multiple locations is a potential solution.

**3. Complexity of AVL Trees:**

* **Challenge:** Implementing AVL trees for maintaining sorted inventory data added complexity.
* **Solution:** While AVL trees ensure O(log n) operations, simpler alternatives like binary search trees (BSTs) could be used in scenarios with less frequent updates.

**4. Next Steps**

To complete the project, the following steps are required:

* **Implement Full Route Optimization:** Complete the implementation of Dijkstra’s algorithm in the graph to support the full shortest path functionality.
* **Real-Time Shipment Scheduling:** Expand the priority queue implementation to handle real-time updates in shipment priorities.
* **Inventory Forecasting:** Incorporate machine learning models for demand forecasting and dynamic inventory updates, which will integrate with the existing data structures.
* **Optimization for Large-Scale Data:** Parallelize graph traversal algorithms and integrate cloud-based storage solutions to handle large-scale data efficiently.

**5. Code Snippets and Documentation**

Key portions of the implementation include the graph structure and priority queue management:

**Graph Implementation (Adjacency List):**

**A computer screen with white text and red and blue text

Description automatically generated**

**Explanation:** This graph class models the transportation network between supply chain entities, storing routes as edges with costs.

**Priority Queue Implementation (Heap):**

**A screen shot of a computer code

Description automatically generated**

**Explanation:** The shipment queue ensures that the most urgent deliveries are processed first, using Python’s heapq module for efficient priority handling.

**GitHub Link:** https://github.com/abhattarai28547/MSCS532\_FinalProject

References

Chang, Y., & Xu, J. (2020). Optimization of Supply Chain Networks Using Graph Theory. *Journal of Supply Chain Management*, 12(3), 245-267.

Gupta, P., & Kumar, A. (2021). Data Structures for Efficient Inventory Management. *International Journal of Computer Science*, 19(2), 45-59.

Li, X., & Zhang, Y. (2019). A Study on Real-Time Supply Chain Route Optimization. *Journal of Logistics Research*, 8(4), 101-123.