Exercise 1: Inventory Management System

Why Data Structures and Algorithms Are Essential

Efficient data storage and retrieval are crucial for handling large inventories due to the following reasons:

**Performance**: Proper data structures ensure that operations like searching, adding, updating, and deleting products are performed quickly, which is vital when dealing with large datasets.

**Scalability**: As the inventory grows, the chosen data structures should handle increased load without significant performance degradation.

**Memory Efficiency**: Efficient data structures minimize memory usage while ensuring quick access to data.

Types of Data Structures Suitable for Inventory Management

**ArrayList**: Provides dynamic array capabilities with efficient indexed access. Suitable for small to medium-sized inventories with infrequent updates.

**HashMap**: Offers fast access, insertion, and deletion of elements based on keys. Ideal for large inventories where quick lookup and updates are necessary.

**LinkedList**: Useful when frequent additions and deletions are required, especially if these operations occur at both ends of the list.

I have chosen hashmap instead of arraylist as the hashmap is much faster than arraylist.

**Time Complexity Analysis**

*Add Product*: O(1) on average, since HashMap insertions are typically O(1) due to hash function distribution.

*Update Product*: O(1) on average, as updating a value in a HashMap involves overwriting the existing entry.

*Delete Product*: O(1) on average, since removing a key-value pair from a HashMap is generally O(1).

*Retrieve Product*: O(1) on average, as HashMap lookups are O(1) when keys are evenly distributed.

Optimization Suggestions

Load Factor and Initial Capacity: Adjust the load factor and initial capacity of the HashMap to minimize rehashing overhead when the number of products grows.

Concurrent HashMap: If the inventory system is accessed by multiple threads, consider using ConcurrentHashMap for thread-safe operations.

Indexing: For more complex queries (e.g., searching by product name), consider additional indexing mechanisms or secondary data structures.