

Exercise 1: Inventory Management System

a. Understand the Problem:

1. Explain why data structures and algorithms are essential in handling large inventories.

Ans: Data structures and algorithms are essential in handling large inventories because they help make software more efficient and effective. Some of its other uses are:

1. Efficient: data structures and algorithms enhance the performance of software by reducing resource consumption.
2. Problem-Solving: Algorithms provide a systematic approach to solving programming problems. Whether it's searching data, sorting inputs, or navigating through complex data relationships, algorithms offer step-by-step procedures that help programmers tackle challenges more easily.
3. Memory Management: Effective use of data structures leads to more efficient memory usage, reducing the overall footprint of an application. This is crucial for high-performance computing and for devices with limited memory resources, like mobile phones.

2. Discuss the types of data structures suitable for this problem.

The types of data structures suitable for this problem are the following:

- Array List: It allows indexed access and is suitable for cases where order matters and frequent additions and deletions are not a concern.
- HashMap: It Provides average $O(1)$ time complexity for insertions, deletions, and lookups, making it ideal for inventory systems where fast access by unique identifiers (e.g., product ID) is required.
- LinkedList: It is Useful if frequent insertions and deletions at arbitrary positions are needed, but less efficient for indexed access.
- Binary Search Tree (BST): It's useful for maintaining a sorted inventory, but average operations are $O(\log n)$.

b. Analysis

1. Analyse the time complexity of each operation (add, update, delete) in your chosen data structure.

- Add Product: Inserting into a HashMap is on average $O(1)$ due to hashing.
- Update Product: Accessing and modifying a value in a HashMap is $O(1)$.
- Delete Product: Removing a key-value pair from a HashMap is $O(1)$.

2. Discuss how you can optimize these operations.

- Load Factor and Rehashing: It ensures the HashMap has an appropriate load factor to minimize rehashing, which can be costly.

- **Concurrency:** For a multi-threaded environment, consider using Concurrent HashMap to handle concurrent access and modifications safely.
- **Memory Usage:** It regularly monitors and manages memory usage to prevent excessive memory consumption, especially if the inventory grows large.
- **Indexing:** For fast retrieval of products based on attributes other than productid, consider additional indexing mechanisms or secondary data structures like trees or lists.