**1.**

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

**Ans :** Data structures and algorithms are quite important in the case of huge inventories since they determine the efficiency of data storage, retrieval, and manipulation. Efficient data structures and algorithms ensure that:

* **Quick Information Access**: Fast retrieval times improve performance, especially when dealing with large datasets.
* **Easy Updation & Modifications**: Efficient insertion, deletion, and updating operations help maintain accurate and up-to-date inventory data.
* **Optimal use of Memory**: Proper data structures minimize memory usage, ensuring the system can handle large volumes of data without performance degradation.
* **Discuss the types of data structures suitable for this problem.**

**Ans :** Types of Data Structures That Are Appropriate for Inventory Management are :

* **Array :** Appropriate to use with dynamic arrays when there is a high need for fast access to elements and iteration. This will also be appropriate in inventories with an equal measure of updates.
* **HashMap :** This is the best approach for getting fast look-up, insertion , and deletion operations based on unique keys (eg. product ids, index number etc.). Well suited to big inventories where fast retrieval and update operations are important.

**4.**

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

**Ans :** Every Operation’s Time Complexity Analysis

* **Add Product (addProduct):** Time Complexity: O(1) (on average) Explanation: Generally, insertion in HashMap is the mathematic function of 1 through direct access by hash code.
* **Update Product (updateProduct):** Time Complexity: O(1) (on average) Explanation: In order to obtain and adjust the product, HashMap enables O(1) for its users.
* **Delete Product (deleteProduct):** Time Complexity: O(1) (on average) Explanation: Similarly to adding elements to a HashMap; removing one also works in O(1).
* **Retrieve Product (getProduct):** Time Complexity: O(1) (on average) Explanation: It’s possible to retrieve information from a hash-table in constant time.
* **Discuss how you can optimize these operations.**

**Ans :** Optimizing HashMap operations involves using a good hash function to minimize collisions and maintaining an appropriate load factor to avoid frequent resizing. These practices ensure efficient O(1) average time complexity for add, update, and delete operations.