**1. Understand the Problem**

**Why Data Structures and Algorithms are Essential**

When dealing with large inventories, efficient data storage and retrieval become crucial. The choice of data structures and algorithms directly impacts the performance of the system. Here's why they are important:

* **Efficient Searching:** With increase in the count of products, an efficient search algorithm reduces the time required to find a specific item.
* **Optimal Storage:**  Data structures ensure that data is stored in a way that minimizes memory usage and maximizes access speed.
* **Fast Updates:** Inventory systems require frequent updates. Efficient data structures allow for quick insertions, deletions, and modifications**.**
* **Scalability:** As the inventory grows, the system should be able to handle increased load.

**Types of Data Structures Suitable for Inventory Management**

Different data structures can be used depending on the specific needs of the inventory management system:

* **ArrayList:** Useful for maintaining a simple list of products. Provides fast access and iteration but slow in operations like insertions and deletions.
* **HashMap (Dictionary):** Ideal for situations where quick access to elements via unique keys (e.g., productId) is required. Provides average O(1) time complexity for add, delete, and update operations.
* **LinkedList:** Useful when frequent insertions and deletions are needed. However, it has slower access times compared to ArrayList.
* **Set:** Used to ensure that there are no duplicate items in the inventory.

**2. Analysis**

**Time Complexity Analysis**

* **Add Product:** O(1) on average because HashMap provides constant time for put operation.
* **Update Product:** O(1) on average because HashMap provides constant time for put operation.
* **Delete Product**: O(1) on average because HashMap provides constant time for remove operation.
* **Display Products: :** O(n), where n is the number of products in the inventory because we need to iterate over all products.

**Optimization**

* **Optimize Add Product Operation:** 
  + **Batch Insertions:** If you have a list of products to add, consider batch insertions to reduce the overhead of multiple method calls and operations.
  + **Return Type:** Return a boolean indicating success or failure, instead of printing messages. This allows the caller to handle the output more flexibly.
* **Optimize Update Product Operation:** 
  + **Avoid Unnecessary Re-insertion**: Only update fields that have changed to reduce unnecessary operations.
  + **Return Type**: Return a boolean indicating whether the update was successful.