**1.Inventory Management System**

**Understanding the problem**

Efficient inventory management requires fast data storage and retrieval, especially in large warehouses. Data structures and algorithms help achieve this by optimizing operations like search, update, and delete. A HashMap is ideal for this system as it provides constant-time access using unique keys like product IDs. In contrast, an ArrayList requires more time to search and update, making it less efficient for large datasets. Choosing the right data structure ensures better performance and scalability.

**Analysis**

In this inventory management system, I used a HashMap to store and manage products using their unique product IDs. This is because HashMap allows very fast access to data using keys, making it suitable for systems where efficiency matters.

1.Add Operation

When a product is added using put(), it gets stored directly by its product ID. This operation has an average time complexity of O(1).

2 .Update Operation

To update a product, I retrieve it using get() and then modify its fields like quantity or price. Since retrieval using a key is also O(1), the update remains efficient.

3 .Delete Operation

Deleting a product using remove() by its ID is also done in constant time, which makes deletion quick.

The only slightly slower part is displaying all products, which requires looping through the values of the HashMap — making it an O(n) operation, where *n* is the number of products.

If this system were to scale up, or support multiple users at the same time, I could use more advanced versions like ConcurrentHashMap for thread safety or add extra indexing for faster search by product name.

Overall, HashMap helps keep the core operations very efficient, and the system performs well for a warehouse-type inventory model.

**Output**

**A screenshot of a computer program

AI-generated content may be incorrect.**