**1. Understand the Problem**

Why data structures and algorithms matter for large inventories:

* Efficiency: Large inventories mean potentially millions of products. Efficient data structures allow fast search, insertion, and deletion.
* Scalability: Algorithms help scale operations so that performance doesn’t degrade as data grows.
* Memory usage: Proper data structures optimize memory and reduce overhead.
* Real-time updates: Warehouse inventory needs fast updates and queries, requiring efficient data handling.

Suitable Data Structures:

* ArrayList: Simple, good for ordered data and iteration. But searching for a product by ID is O(n).
* HashMap (or Hashtable): Offers O(1) average time complexity for add, update, delete, and search by key (productId). Ideal for fast lookup by productId.
* TreeMap: Maintains sorted order of keys, O(log n) time for operations. Useful if sorted retrieval is required.
* LinkedList: Useful for frequent insertions/deletions at ends but slow search (O(n)).

Best fit: HashMap is generally best for inventory by productId due to its constant-time average access.

**4. Analysis**

|  |  |  |
| --- | --- | --- |
| **OPERATION** | **TIME COMPLEXITY** | **EXPLANATION** |
| Add product | Average: O(1); Worst: O(n) | Constant time insertion; worst case due to hash collisions and resizing. |
| Update Product | Average: O(1); Worst: O(n) | Replace the value for a key. |
| Delete Product | Average: O(1); Worst: O(n) | Remove key-value pair. |
| Get product | Average: O(1); Worst: O(n) | Lookup by key. |

How to optimize further:

* Use good hash functions: Minimize collisions.
* Load factor tuning: Adjust HashMap load factor to reduce resizing overhead.
* Use concurrent data structures: For multi-threaded access (e.g., ConcurrentHashMap).
* Indexing for other queries: If searching by other attributes (like productName), maintain secondary indexes (maps).
* Batch updates: Minimize lock contention if in concurrent environment.
* Use databases: For very large inventories, consider a DB with indexing.