Understand Asymptotic Notation

Big O Notation

- Big O notation describes the **upper bound** of an algorithm's running time in the **worst-case** scenario.
- It allows us to evaluate how the algorithm scales as input size increases — focusing on **efficiency**.

Example Complexity Levels:

- O(1): Constant time
- O(log n): Logarithmic time (e.g., Binary Search)
- O(n): Linear time (e.g., Linear Search)
- O(n log n): Efficient sorts like Merge Sort
- O(n^2): Nested loops (e.g., Bubble Sort)

Search Time Scenarios

Case	Linear Search	Binary Search
Best	O(1) – first element	O(1) – middle element
Average	O(n/2) ≈ O(n)	O(log n)
Worst	O(n) – last or not found	O(log n) – recurse fully

Analysis

Time Complexity

Algorithm	Time Complexity	Suitable for
Linear Search	O(n)	Small datasets or unsorted data
Binary Search	O(log n)	Large datasets, sorted data

Which is Better?

- **Binary search is optimal** for **fast performance**, but only if data is **sorted**.
- **Linear search is flexible**, but **slower** on large datasets.

Recommendation for E-commerce Platform

- Use **Binary Search** (or even more advanced techniques like Trie or Search Indexing) if:
    - Products are pre-sorted by name or ID.
    - You need **fast retrieval** in a large catalog.

- Consider **Linear Search** for:
    - Small product datasets.
    - Flexible, on-the-fly filtering (e.g., by category, partial matches).