1. Understand Asymptotic Notation

What is Big O Notation?

Big O notation is a mathematical notation used to describe the time complexity or space complexity of an algorithm in terms of input size (n). It tells us how an algorithm's performance grows as the input grows, helping us evaluate efficiency.

Why is Big O useful?

- It helps us compare algorithms independently of hardware or implementation.
- It guides design decisions, especially in performance-critical systems like an e-commerce platform.

Best, Average, and Worst-Case Scenarios in Search

Scenario	Linear Search	Binary Search (sorted)
Best Case	O(1) - first element	O(1) - middle match
Average Case	O(n/2) → O(n)	O(log n)
Worst Case	O(n) - last/absent	O(log n)

4. Analysis of Time Complexity

Linear Search

- Time Complexity:
    - Best: O(1)
    - Average: O(n)
    - Worst: O(n)
- Pros:
    - Works on unsorted data
    - Simple to implement
- Cons:
    - Slower for large datasets

Binary Search

- Time Complexity:
    - Best: O(1)
    - Average & Worst: O(log n)

- Pros:
    - Much faster on large datasets
- Cons:
    - Requires sorted data
    - Overhead of maintaining sorted structure

Which is more suitable for your platform?

For an e-commerce platform, where users perform frequent and fast product searches:

Binary Search is more suitable, assuming product data can be kept sorted (or stored in structures like balanced trees or indexed DBs). It's significantly faster (O(log n)) and scales well.

However, if the dataset is:

- small, or
- frequently updated with unsorted data (e.g., real-time inventory),

then Linear Search might still be useful for simplicity in specific parts of the system.