# E-commerce Platform Search Functionality Analysis

1. **Understanding Asymptotic Notation**

## Big O Notation

Big O notation is a way to measure how the running time or resource usage of an algorithm grows as the input size increases. It helps us compare algorithms by focusing on their efficiency, especially for large datasets. For example, an algo- rithm with O(n) time takes roughly twice as long if the input size doubles, while O(log n) grows much slower. This allows developers to choose faster algorithms for e-commerce platforms where quick searches improve user experience.

## Best, Average, and Worst-Case Scenarios for Search Operations

Search algorithms have different performance depending on the situation:

## Linear Search:

* + - * **Best Case**: O(1) – The target is at the first position, so only one check is needed.
      * **Average Case**: O(n) – On average, half the list is checked to find the target or determine it’s not there.
      * **Worst Case**: O(n) – The target is at the end or not in the list, requiring a check of all n items.
    - **Binary Search** (requires sorted data):
      * **Best Case**: O(1) – The target is at the middle of the list, found in one step.
      * **Average Case**: O(log n) – The list is halved repeatedly, so about log n steps are needed.
      * **Worst Case**: O(log n) – The target is at an extreme end or not in the list, still taking log n steps.

# Setup

## Product Class

The Product class represents an item in the e-commerce platform. It has three attributes for searching:

* + - **productId**: A unique integer identifier for each product.
    - **productName**: A string describing the product’s name.
    - **category**: A string indicating the product’s category (e.g., Electronics).

This class is used to create product objects stored in arrays for searching. For linear search, the array can be unsorted. For binary search, the array must be sorted by productId.

# Implementation

## Linear and Binary Search Algorithms

Two search algorithms were implemented:

* + - **Linear Search**: Checks each product in the array one by one until it finds a matching productId or reaches the end. It works on unsorted arrays and returns the product or null if not found.
    - **Binary Search**: Works on a sorted array by repeatedly dividing the search range in half. It compares the target productId with the middle element, then searches the left or right half depending on whether the target is smaller or larger. It returns the product or null if not found.

## Product Storage

Products are stored in two arrays:

* + - **Unsorted Array**: Used for linear search, containing products in any order.
    - **Sorted Array**: Used for binary search, with products sorted by productId to enable efficient halving of the search space.

# Analysis

## Time Complexity Comparison

* + - **Linear Search**: Always O(n) because it may need to check every product. For an e-commerce platform with 1,000,000 products, it could take up to 1,000,000 steps.
    - **Binary Search**: O(log n) because it halves the search space each step. For 1,000,000 products, it takes about log(1,000,000) 20 steps, much faster than linear search.

## Suitability for E-commerce Platform

For an e-commerce platform, search speed is critical to ensure a smooth user experience. Linear search is simple and works well for small product lists (e.g., under 100 items) or when the product list changes frequently, making sorting impractical. However, its O(n) time complexity makes it too slow for large plat- forms with thousands or millions of products.

Binary search is far more suitable for large e-commerce platforms because its O(log n) time complexity ensures fast searches even with millions of products. The requirement for a sorted array is manageable since productIds can be in- dexed and sorted in a database. If the product list is updated often, maintaining

a sorted order may add overhead, but this can be mitigated with efficient data structures like balanced search trees or database indexing.

**Recommendation**: Binary search is the better choice for a large e-commerce platform due to its speed. For optimal performance, use a database with an index on productId to mimic binary search efficiency without manual sorting.