E-commerce Platform Search Function

**1. Understanding Asymptotic Notation**

***Big O Notation (O):***Big O notation is a mathematical representation used to describe the upper bound of an algorithm’s running time as a function of input size n. It helps developers analyze the efficiency of algorithms regardless of hardware, allowing them to select the best algorithm for large-scale applications.

In this exercise, Big O was used to evaluate search algorithms (linear and binary) in different scenarios.

**2. Search Case Scenarios**

| Search Operation | Best Case | Average Case | Worst Case |
| --- | --- | --- | --- |
| *Linear Search* | O(1) – first match | O(n/2) ≈ O(n) | O(n) |
| *Binary Search* | O(1) – middle match | O(log n) | O(log n) |

Linear search checks every element sequentially. Binary search divides the search space in half on each step but requires sorted data.

**3. Setup**

*We created a* Product *class with the following attributes:*

* productId: Unique identifier for the product.
* productName: Name of the product (used in exact match search).
* productCategory: Category to which the product belongs (used for grouped search).
* price: Used for range-based queries (like min–max price filter).

15 sample Product objects were created to simulate a real-world catalog.

**4. Implementation Overview**

We implemented two search classes:

1. **ProductLinearSearch**
   * Implements simple sequential search across all products.
   * Supports search by ID, name, category, and price range.
2. **ProductBinarySearch**
   * Uses binary search to optimize:
     + Product ID search
     + Product name search
     + Price range queries (via a hybrid method)
   * Product array is pre-sorted by the required field before calling binary search.

**5. Price Range Search – Hybrid Approach**

Binary search alone is not suitable for range queries since it returns only a single match.  
To overcome this, we implemented a **hybrid approach**:

* Use **binary search** to find the first index with price ≥ min (ceil).
* Use **binary search** to find the last index with price ≤ max (floor).
* Perform a **linear scan between ceil and floor** to collect all valid products.

**Time Complexity:**

* Best Case: O(log n + k) where k is the number of matching products.
* Worst Case: O(n) when the range includes most or all products.

**6. Time Complexity Comparison**

|  |  |  |
| --- | --- | --- |
| **Operation** | **Linear Search** | **Binary Search** |
| Search by ID | O(n) | O(log n) |
| Search by Name | O(n) | O(log n) |
| Search by Category | O(n) | O(n) (linear fallback) |
| Search by Price Range | O(n) | O(log n + k) (hybrid) |