# **Exercise 2: E-commerce Platform Search Function**

## **1. Understand Asymptotic Notation**

**Big O Notation:**

**- Definition:** Describes the upper bound of an algorithm's runtime or space requirements based on input size, helping to gauge its efficiency in the worst-case scenario.

**- Purpose:** Facilitates comparison between algorithms by focusing on their performance growth relative to input size.

**Best, Average, and Worst-Case Scenarios:**

**- Best Case:**

**- Linear Search:** O(1) if the target element is found at the start.

**- Binary Search:** O(1) if the target element is exactly at the midpoint.

**- Average Case:**

- **Linear Search:** O(n) on average, as it may require examining half of the elements.

**- Binary Search:** O(log n) due to the halving of the search space with each comparison.

**- Worst Case:**

**- Linear Search:** O(n) if the target is at the end or not present.

**- Binary Search:** O(log n), provided the data is sorted, with exponential reduction in search space.

**2. Setup**

**Create a Class `Product`:**

**- Purpose:** Represents a product with attributes such as product ID, name, and category.

**- Methods:**

**- Constructor:** Initializes a `Product` instance with specified attributes.

**- Getters and Setters:** Methods to access and modify product attributes.

**3. Implementation**

**Linear Search Algorithm:**

- Method `linearSearch(Product[] products, String searchTerm)`

- Purpose: Searches through an array of `Product` objects to find one with a matching name.

- Functionality: Iterates through the array, comparing each product's name with the search term. If a match is found, the corresponding product is returned; otherwise, `null` is returned.

**Binary Search Algorithm (requires sorted array by productName):**

- Method `binarySearch(Product[] products, String searchTerm)`

- Purpose: Efficiently searches for a `Product` in a sorted array based on product names.

- Functionality: Uses a binary search approach by repeatedly dividing the search space into halves. It compares the middle product's name with the search term and adjusts the search range accordingly. If a match is found, the product is returned; if not, `null` is returned.

**- Method `sortProductsByName(Product[] products)`**

- Purpose: Prepares the array for binary search by sorting products alphabetically by their names.

- Functionality: Uses a sorting algorithm to arrange products in ascending order based on their names, ensuring that binary search can be applied effectively.

**4. Analysis**

**Time Complexity Comparison:**

**- Linear Search:**

- Time Complexity: O(n)

- Space Complexity: O(1)

- Advantages: Simple implementation, no need for sorted data.

- Disadvantages: Inefficient for large datasets as it may require scanning every element.

**- Binary Search:**

- Time Complexity: O(log n)

- Space Complexity: O(1)

- Advantages: Efficient for large datasets, as it reduces the search time significantly.

- Disadvantages: Requires the dataset to be sorted, which adds preprocessing time with a complexity of O(n log n).

**Suitability for E-commerce Platform:**

- Binary Search is generally preferable for large datasets where the data is sorted, as it provides faster search times.

- Linear Search is suitable for smaller or unsorted datasets but becomes less efficient with increasing size.