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

**Scenario:** You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Steps:**

1. **Understand Asymptotic Notation:**
   * Explain Big O notation and how it helps in analyzing algorithms.
   * Describe the best, average, and worst-case scenarios for search operations.
2. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.
3. **Implementation:**
   * Implement linear search and binary search algorithms.
   * Store products in an array for linear search and a sorted array for binary search.
4. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

**Solution :**

1. **Understanding :**

**Big O Notation**

Big O notation is a mathematical notation used to describe the upper bound of an algorithm's running time or space requirements in terms of input size. It provides a high-level understanding of the algorithm's efficiency and scalability. Big O notation helps in comparing different algorithms and understanding their behavior as the input size grows.

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

* **Best Case**: The scenario where the algorithm performs the minimum number of operations. For search algorithms, this often means finding the target element on the first attempt.
* **Average Case**: The scenario that represents the expected performance over a large number of inputs. It considers all possible inputs and their probabilities.
* **Worst Case**: The scenario where the algorithm performs the maximum number of operations. This provides a guarantee on the upper bound of the running time.

1. **Analysis :**

#### Time Complexity

* **Linear Search:**
  + **Best Case**: O(1) (Product is at the first position)
  + **Average Case**: O(n/2) ≈ O(n) (Product is somewhere in the middle)
  + **Worst Case**: O(n) (Product is at the last position or not present)
* **Binary Search**:
  + **Best Case**: O(1) (Product is at the middle position)
  + **Average Case**: O(log n)
  + **Worst Case**: O(log n) (Product is at either end or not present)