**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.

**Big O Notation**: Describes the upper bound of an algorithm's running time or space requirements in the worst-case scenario, helping analyze how performance scales with input size (n).

* **O(1)**: Constant time - independent of input size.
* **O(n)**: Linear time - grows linearly with input size.
* **O(log n)**: Logarithmic time - grows logarithmically with input size.
* **O(n^2)**: Quadratic time - grows quadratically with input size.

**Best, Average, and Worst-Case Scenarios for Search Operations**:

* **Best Case**: Search takes the least time (target is first in the array).
* **Average Case**: Expected time averaged over all inputs.
* **Worst Case**: Search takes the most time (target is last or not in the array).

1. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.
2. **Implementation:**
   * Implement linear search and binary search algorithms.
   * Store products in an array for linear search and a sorted array for binary search.
3. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

**Time Complexity of Linear and Binary Search Algorithms**

* **Linear Search**:
  + **Time Complexity**: O(n)
  + **Explanation**: Checks each element until the target is found or the end of the array is reached.
* **Binary Search**:
  + **Time Complexity**: O(log n)
  + **Explanation**: Divides the array into halves and discards one half each step. Requires a sorted array.

**Comparison and Suitability**:

* **Linear Search**:
  + **Best Case**: O(1) (target is first element).
  + **Average Case**: O(n).
  + **Worst Case**: O(n) (target is last or not present).
  + **Suitability**: Best for small or unsorted datasets.
* **Binary Search**:
  + **Best Case**: O(1) (target is middle element).
  + **Average Case**: O(log n).
  + **Worst Case**: O(log n) (target not present).
  + **Suitability**: Best for large, sorted datasets due to logarithmic time complexity.

**E-commerce Platform Suitability**:

* For critical search performance, binary search is preferred over linear search due to its superior time complexity (O(log n) vs. O(n)), making it more efficient for large, sorted datasets. Linear search is only efficient for small or unsorted datasets.