**E-commerce Platform Search Function**

1. **Understand Asymptotic Notation:**

**Big O Notation** is a standard way to describe how an algorithm performs as the size of the input increases. It represents the **upper limit** of an algorithm's runtime or memory usage, helping us understand how it behaves under scaling. By using Big O, we can assess and compare the **efficiency** of different algorithms in a consistent and meaningful way.

**Case-Based Performance Analysis**

* **Best Case:**  
  This reflects the most favourable condition, where the algorithm completes in the **least number of steps**. For example, in a search operation, the best case occurs when the target element is found at the **very beginning** of the list.
* **Average Case:**  
  This considers a **typical situation**, calculating the average number of steps required across all possible input configurations. It offers a balanced view of how the algorithm is likely to perform in general usage.
* **Worst Case:**  
  This scenario accounts for the **maximum effort** the algorithm may need to make. In search operations, this happens when the element is either located at the **very end** of the list or not present at all—forcing the algorithm to scan the entire array.

**2. Setup:**

Create a new project in Java programming language for E-commerce Platform Search Function.

**3. Implementation:**

Step 1: Creating a class Product with attributes **productId, productName** and **category**.

Step 2: Implementation of Linear Search: Linear search iterates through each element in an array until it finds the target. It works on both sorted and unsorted arrays.

Step 3: Implementation of Binary Search: Binary search requires the array to be sorted. It repeatedly divides the search interval in half, checking if the target is in the left or right half.

**4. Analysis:**

**Time Complexity Comparison**

* **Linear Search:**
  + Best Case: O(1) (element found at the beginning)
  + Average Case: O(n) (element found in the middle)
  + Worst Case: O(n) (element not found or at the end)
* **Binary Search:**
  + Best Case: O(1) (element found at the middle)
  + Average Case: O(log n) (element found after several divisions)
  + Worst Case: O(log n) (element not found)

**Algorithm Suitability:**

**Binary Search is more suitable because**

1. **Large Product Catalog**  
   E-commerce platforms typically have thousands of products. Binary search performs better on large datasets due to its O(log n) efficiency.
2. **Products are Frequently Browsed, Not Frequently Changed**  
   Since the product list doesn't change every second, sorting it once and using binary search makes future searches faster.
3. **Improved User Experience**  
   Fast search results improve the speed of the platform, leading to better user satisfaction.
4. **Scalability**  
   Binary search scales well as the number of products increases, making it future-proof for growing platforms.
5. **Can Combine with Advanced Structures**  
   Binary search forms the basis for more advanced data structures (like search trees or search indices), which are often used in real-world e-commerce systems.

Output

