**E-commerce Platform Search Function**

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

**Understanding Asymptotic Notation:**

**Big O notation:** describes the growth rate of algorithm execution time with respect to input size. It classifies algorithms as:

O(1) – Constant time

O(n) – Linear time

O(log n) – Logarithmic time

O(n log n) – Linearithmic time

O(n²) – Quadratic time

**Describing the best, average, and worst-case scenarios for search operations.**

Best Case: The product is found immediately , at 0th index for linear search and (0+n)/2 for binary - O(1) for linear search; O(1) if binary search hits the middle.

Average Case: Product is equally likely at any position - O(n) for linear search; O(log n) for binary search.

Worst Case: Product is last or not found -O(n) for linear search; O(log n) for binary search.

**Product class**

We define a Product class with attributes:

* productId: Unique product identifier
* productName: Name of the product
* category: Category of the product

**Search Algorithms**

**Linear Search**: Scans each product one by one.

**Binary Search**: Uses a sorted list and repeatedly halves the search range.

**Comparison:**

Linear Search is simple and works without sorting the data. However, it is slow for large product lists.

Binary Search is much faster , especially as the list grows , but requires the product list to be sorted.

**Analysis:**

For an e-commerce platform with thousands of products and customers searching often, binary search is a good choice. Binary search can quickly find a product because it looks at the middle of the list and eliminates half the items at a time. However, it requires the product list to be sorted first. Sorting takes extra time whenever we add or remove a product. Even so, most platforms have more people searching than changing products, so sorting the list once and using binary search usually saves a lot of time in the long run.

OUTPUT :

