E-commerce Platform Search

1️⃣ Big O Notation and Its Role in Analyzing Algorithms

Big O notation is a mathematical way to describe how the performance of an algorithm changes as the size of the input data grows. It focuses on the asymptotic behavior — meaning how fast the runtime or space usage increases when dealing with very large inputs. Big O helps developers compare algorithms by providing a high-level performance measure that ignores constant factors and small inputs. For example, an algorithm with time complexity O(n) grows linearly with input size, while an algorithm with O(log n) grows much slower and is more efficient for large data. Using Big O helps us choose the most efficient algorithm when performance matters.

2️⃣ Best, Average, and Worst-Case Scenarios for Search Operations

When analyzing search algorithms, we often think about three scenarios:

Best case: The search target is found immediately.

Average case: The target is found somewhere in the middle.

Worst case: The target is not found or is at the end of the list.

For Linear Search:

Best case: O(1) — target is the first item.

Average case: O(n/2), simplified to O(n).

Worst case: O(n) — target is not present or last.

For Binary Search:

Best case: O(1) — target is at the middle.

Average case: O(log n).

Worst case: O(log n) — reduces search space by half each step.

These scenarios help us understand how an algorithm behaves in real situations, not just in ideal ones.

3️⃣ Comparing Time Complexity of Linear and Binary Search

Linear Search is simple and works on unsorted data. Its time complexity is:

Best case: O(1)

Average case: O(n)

Worst case: O(n)

Binary Search is much faster but only works on sorted data. Its time complexity is:

Best case: O(1)

Average and Worst case: O(log n)

Clearly, binary search performs far better for large datasets — but with the extra requirement of sorted data.

4️⃣ Which Algorithm is More Suitable and Why

For an e-commerce platform where fast search performance is critical (users expect quick results), Binary Search is generally more suitable — assuming we can maintain a sorted product list (which is often the case in databases or in-memory structures). Binary Search’s O(log n) performance allows the platform to scale to thousands or millions of products. However, for dynamic, unsorted data (such as a new or unsorted feed), Linear Search can still be useful because it works without requiring sorting.

In summary: for the core product catalog (which can be sorted), Binary Search is ideal. For smaller or unsorted datasets (like recent views, or temporary lists), Linear Search can still be effective.

Output🡺

