**Hands-On:** **E-commerce Platform Search Function**

**Asymptotic Notation:**

**Big O notation**is a powerful tool used in computer science to describe the time complexity or space complexity of algorithms. **Big-O** is a way to express the **upper bound**of an algorithm’s time or space complexity.

* Describes the asymptotic behavior (order of growth of time or space in terms of input size) of a function, not its exact value.
* Can be used to compare the efficiency of different algorithms or data structures.
* It provides an**upper limit** on the time taken by an algorithm in terms of the size of the input. We mainly consider the worst case scenario of the algorithm to find its time complexity in terms of Big O
* It’s denoted as**O(f(n))**, where**f(n)** is a function that represents the number of operations (steps) that an algorithm performs to solve a problem of size **n**.

### 

### ****Search Operation Cases****

* **Best Case:** When the item is found at the first position.
* **Average Case:** Item is somewhere in the middle.
* **Worst Case:** Item is not present or found at the end.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** |
| Linear Search | O(1) | O(n) | O(n) |
| Binary Search | O(1) | O(log n) | O(log n) |

## Analysis & Conclusion

### Time Complexity

* **Linear Search:** O(n) — not efficient for large product catalogs.
* **Binary Search:** O(log n) — much faster but needs sorted data.

### ****Which Algorithm is More Suitable and Why?****

* **Binary Search** is generally **more suitable** for an e-commerce platform where fast search performance is critical, especially as the number of products grows large.
* It provides **logarithmic time complexity**, making it highly efficient for large datasets.
* However, it requires that the product list be **sorted**, which may need additional processing or a sorted data structure (like TreeMap, SortedList, or database indexing).
* For **small datasets or unsorted arrays**, use **linear search**.
* For **large, sorted datasets**, use **binary search**.
* In real-world apps, use **hash maps** or **search trees** for faster average-case lookups (O(1) to O(log n)).

For **real-time platforms**, products are usually stored in **indexed databases**, where binary search or hashing is used under the hood to give near-instantaneous results