**1\_E-commerce Platform Search Function**

**a. What is Big O Notation?**

* When we write code, especially for things like searching through thousands of products, it's important to know how fast it runs as the number of items increases.
* That’s where asymptotic notation comes in, it helps us describe the performance of an algorithm without actually running it, based on input size.

Big O notation specifically tells us the worst-case growth rate, how the time or space needed increases as data grows. For example:-

* O(1) means it always takes the same amount of time, no matter how big the list is.
* O(n) means the time increases linearly with the number of items.
* O(log n) means it gets only slightly slower as the data increases, a huge performance win!

It ignores constants and focuses only on the most impactful term, so something like O(3n + 8) simplifies to O(n).

**b. Best, Average, and Worst-Case Scenarios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario** | **Description** | **Linear Search** | **Binary Search** |
| **Best Case** | The item is right at the start | O(1) | O(1) |
| **Average Case** | The item is somewhere in the middle | O(n/2) ≈ O(n) | O(log n) |
| **Worst Case** | The item is at the end or not in the list at all | O(n) | O(log n) |

* **Linear Search** checks every item one by one, great for small lists but slow for large ones.
* **Binary Search** quickly cuts the list in half each time, but the data must be **sorted** first.

**c. Comparison - Linear vs Binary Search**

|  |  |  |  |
| --- | --- | --- | --- |
| **Search Type** | **Time Complexity** | **Data Requirement** | **Suitable When** |
| **Linear Search** | O(n) | Unsorted data | Small data size, no overhead |
| **Binary Search** | O(log n) | Requires sorted array | Large datasets, frequent queries |

**Which is better for e-commerce platforms?**

* **Binary search** is **much faster** for large product catalogues, ideal when data is **sorted and mostly read-only**.
* However, if data is **unsorted or frequently changing**, **linear search** or **HashMap-based lookup** might be better.