**Step 1: Understand Asymptotic Notation:**

* Explain Big O notation and how it helps in analyzing algorithms.
* Describe the best, average, and worst-case scenarios for search operations

**Ans:**

**Big O notation** is a way to measure how efficient an algorithm is, especially as the input size grows. It doesn't give exact timing but shows how the time (or space) increases based on the size of the data.

For example:

* O(1) means the operation takes the same time no matter how big the input is.
* O(n) means the time grows linearly with the number of elements.
* O(log n) means the time increases slowly even as the input grows.

It helps us compare different algorithms and choose the one that performs better as data grows.

**Best, Average, and Worst Cases in Search:**

**Linear Search:**

* Best Case: O(1) - if the element is right at the beginning.
* Average Case: O(n/2) - if it’s somewhere in the middle.
* Worst Case: O(n) - if it's at the end or not present at all.

**Binary Search:**

* Best Case: O(1) - if the target is right in the middle.
* Average & Worst Case: O(log n) - because we divide the list in half each time.

**Step 4: Analysis:**

* Compare the time complexity of linear and binary search algorithms.
* Discuss which algorithm is more suitable for your platform and why.

**Ans:**

**Time Complexity Comparison:**

|  |  |
| --- | --- |
| Search Method | Time Complexity |
| Linear Search | O(n) |
| Binary Search | O(log n) |

**Binary Search is clearly faster than Linear Search** for large data sets - **but only if the data is sorted.**

* If our product list is **already sorted**, Binary Search is the better option.
* If the data is **not sorted and constantly changing**, Linear Search might be simpler.

For an e-commerce platform, we usually want **fast search**, so sorting the product list or using efficient data structures makes **Binary Search** the more suitable choice.