**Understand Asymptotic Notation**

**Big O Notation :**

Big O Notation describes the upper bound of the time (or space) complexity of an algorithm in terms of input size n. It helps us analyze how the runtime grows as the input grows.

* **O(1)**: Constant time
* **O(n)**: Linear time
* **O(log n)**: Logarithmic time
* **O(n log n)**: Log-linear time
* **O(n²)**: Quadratic time

**Best, Average, and Worst Case**

For **search operations**, these cases vary based on data distribution and algorithm:

| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** |
| --- | --- | --- | --- |
| Linear Search | O(1) | O(n/2) ≈ O(n) | O(n) |
| Binary Search | O(1) | O(log n) | O(log n) |

**Analysis**

**Time Complexity Comparison**

| **Algorithm** | **Time Complexity** | **Sorted Required?** | **Suitable For** |
| --- | --- | --- | --- |
| Linear Search | O(n) | No | Small or unsorted data |
| Binary Search | O(log n) | Yes | Large, sorted datasets |

**Which Is Better?**

* **Linear Search** is simple and works on **unsorted** data but is inefficient for large datasets.
* **Binary Search** is **faster** but requires sorting (which is O(n log n)).