**Understanding Asymptotic Notation**

**Big O notation** is a mathematical concept used in computer science to describe the **performance or complexity of an algorithm** in terms of **time or space** as the size of the input (n) grows.

**Why Big O is Important in Analyzing Algorithms**

* It **abstracts away machine-dependent details**, focusing only on how the algorithm scales.
* Helps you compare different algorithms for efficiency.
* Allows you to choose the most appropriate algorithm/data structure for a task, especially with **large datasets**.

**Best, Average, and Worst Case Scenarios in Search Operations**

When analyzing search algorithms, it’s important to consider how they behave in **different scenarios**:

**1. Best Case**

* The search target is found in the **first attempt**.
* **Linear Search**: O(1) if the element is at the beginning.
* **Binary Search**: O(1) if the element is at the middle of the sorted array.

**2. Average Case**

* Represents the **expected performance** over many runs.
* **Linear Search**: O(n/2) → Simplifies to O(n)
* **Binary Search**: O(log n) on average in a sorted list

**3. Worst Case**

* The algorithm takes the **maximum number of steps** to find or fail.
* **Linear Search**: O(n) if the element is at the end or not found.
* **Binary Search**: O(log n) even in the worst case due to halving the list each step.