Big O Notation:

Big O notation describes the **performance and efficiency** of an algorithm as the **input size increases**. It helps analyse the **scalability** of algorithms regardless of hardware.

- **O(1):** Constant time (best case)
- **O(n):** Linear time
- **O(log n):** Logarithmic time (binary search)
- **O(n^2):** Quadratic time

Using Big O, we can **compare search methods** and choose the optimal one for large data sets.

Search Operation Cases:

Linear Search:

- **Best Case:** The element is found at the **beginning** of the array. This takes only **one comparison**, so the time complexity is **O(1)**.
- **Average Case:** The element is found **somewhere in the middle** of the array. On average, it takes about **n/2 comparisons**, resulting in **O(n)** time complexity.
- **Worst Case:** The element is found at the **end** of the array or is **not present at all**. In both cases, the entire array is scanned, so the time complexity is **O(n)**.

Binary Search:

- **Best Case:** The element is found in the **middle** of the array on the first comparison. This gives a time complexity of **O(1)**.
- **Average Case:** The element is found after dividing the array **$\log_2(n)$** times. The time complexity is **O(log n)**.
- **Worst Case:** The element is **not present** or located in one of the far ends after multiple divisions. The search still goes through **$\log_2(n)$** divisions, so the time complexity is **O(log n)**.