**Big O Notation**

**Big O notation** describes the upper bound of an algorithm's runtime or space complexity, helping compare efficiency. It focuses on how the runtime or space grows with input size (n), ignoring constants and lower-order terms.

**Common Big O Notations**

* **O(1):** Constant time - The runtime does not change with the input size.
* **O(log n):** Logarithmic time - The runtime grows logarithmically with the input size.
* **O(n):** Linear time - The runtime grows linearly with the input size.
* **O(n log n):** Linearithmic time - The runtime grows in proportion to n log n.
* **O(n^2):** Quadratic time - The runtime grows quadratically with the input size.
* **O(2^n):** Exponential time - The runtime grows exponentially with the input size.

**Search Operation Scenarios**

**Linear Search**

* **Best-Case:** Target is the first element. Time: O(1)
* **Average-Case:** Target is in the middle. Time: O(n)
* **Worst-Case:** Target is the last or not present. Time: O(n)

**Binary Search (in a sorted array)**

* **Best-Case:** Target is the middle element. Time: O(1)
* **Average-Case:** Generally found after log(n) comparisons. Time: O(log n)
* **Worst-Case:** Target is not present. Time: O(log n)

**Analysis:**

### Time Complexity Comparison of Linear and Binary Search

#### Linear Search

**Linear search** examines each element in the array sequentially until the target element is found or the end of the array is reached.

* **Best-Case:** The target element is the first element.  
  **Time Complexity:** O(1)
* **Average-Case:** The target element is expected to be in the middle of the array.  
  **Time Complexity:** O(n/2) ≈ O(n)
* **Worst-Case:** The target element is the last element or not present.  
  **Time Complexity:** O(n)

#### Binary Search

**Binary search** repeatedly divides the sorted array in half, eliminating half of the remaining elements with each comparison.

* **Best-Case:** The target element is the middle element.  
  **Time Complexity:** O(1)
* **Average-Case:** The target element is generally found after log(n) comparisons.  
  **Time Complexity:** O(log n)
* **Worst-Case:** The target element is not present, and the array is reduced to a single element.  
  **Time Complexity:** O(log n)

In my case as am working on a working on the search functionality of an e-commerce platform and the search needs to be optimized for fast performance, **binary search algorithm** is more suitable and efficient and will be preferred over the linear search algorithm.