Big O notation is a mathematical concept used to describe the upper limit of an algorithm's running time or space requirements in terms of the size of the input. It helps in understanding the performance and scalability of algorithms by focusing on their growth rates rather than specific constants or lower-order terms.

* **O(1):** Constant time complexity, where the running time does not change with the input size.
* **O(log n):** Logarithmic time complexity, where the running time increases logarithmically with the input size.
* **O(n):** Linear time complexity, where the running time increases linearly with the input size.
* **O(n log n):** Log-linear time complexity, typical for efficient sorting algorithms like merge sort.
* **O(n^2):** Quadratic time complexity, where the running time increases quadratically with the input size, common in algorithms like bubble sort.

**Best, Average, and Worst-Case Scenarios for Search Operations:**

1. **Best Case:**
   * The scenario where the search operation completes in the minimum possible time.
   * Example: In linear search, the best case is finding the target element at the first position, O(1).
2. **Average Case:**
   * The expected time complexity considering all possible inputs.
   * Example: In linear search, on average, the target element is found in the middle of the array, O(n/2), which simplifies to O(n).
3. **Worst Case:**
   * The scenario where the search operation takes the maximum possible time.
   * Example: In linear search, the worst case is when the target element is at the last position or not present at all, O(n).