**Big O notation:**Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity. In computer science, it's used to classify algorithms by how their running time or space requirements grow as the input size grows. It provides an upper bound on the growth rate of an algorithm's complexity.

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

1. **Best-Case Scenario:** This is the most efficient execution of an algorithm. For search operations, it typically occurs when the target element is found at the very first comparison.
2. **Linear Search:** The best case is when the element is the first in the list. Time complexity: O(1).
3. **Binary Search:** The best case is when the element is the middle element of the array. Time complexity: O(1).

**2.Average-Case Scenario:** This describes the expected performance of an algorithm over many different inputs. It's often more difficult to calculate precisely as it involves probability distributions of inputs.

**(i) Linear Search:** On average, the element is found somewhere in the middle of the list. Time complexity: O(n).

**(ii) Binary Search:** The element is found after a logarithmic number of comparisons. Time complexity: O(logn).

**3.Worst-Case Scenario:** This is the least efficient execution of an algorithm. For search operations, it typically occurs when the target element is either not present in the data structure or is located at the very last position to be checked.

**(i) Linear Search:** The worst case is when the element is the last in the list, or not present at all. The algorithm has to check every element. Time complexity: O(n).

**(ii) Binary Search:** The worst case is when the element is not in the array, or when it takes the maximum number of divisions to find it (e.g., at an end of a sub-array). Time complexity: O(logn).