Big O Notation:

Definition: Big O notation describes the upper bound of the time complexity of an algorithm, representing the worst-case scenario. It helps in understanding how the runtime of an algorithm grows with the input size.

Purpose: Big O notation provides a high-level understanding of the efficiency of an algorithm, allowing for comparison between different algorithms based on their scalability.

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

Best Case: The scenario where the algorithm performs the minimum number of operations. For linear search, this occurs when the target element is at the first position (O(1)). For binary search, the best case is also O(1), when the middle element is the target.

Average Case: The scenario representing the expected number of operations performed, typically based on assumptions about the distribution of input data. For linear search, it's O(n/2), simplifying to O(n). For binary search, it's O(log n).

Worst Case: The scenario where the algorithm performs the maximum number of operations. For linear search, this occurs when the target element is at the last position or not in the list at all (O(n)). For binary search, the worst case is O(log n).

Analysis

Time Complexity Comparison:

Linear Search:

Best Case: O(1) (target is the first element)

Average Case: O(n/2) = O(n)

Worst Case: O(n) (target is the last element or not present)

Binary Search:

Best Case: O(1) (target is the middle element)

Average Case: O(log n)

Worst Case: O(log n) (searching through all levels)