Linear Search:

Description: Linear search is a simple search algorithm that checks each element in the list sequentially until the desired element is found or the list ends.

Time Complexity:

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

Average Case: O(n/2) ≈ O(n)

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

Binary Search:

Description: Binary search is an efficient search algorithm that works on sorted lists. It repeatedly divides the search interval in half. If the target value is less than the middle element, the search continues in the left half; otherwise, it continues in the right half.

Time Complexity:

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

Average Case: O(log n)

Worst Case: O(log n) (when the target is not present)

Analysis

Time Complexity:

Linear Search:

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

Average Case: O(n/2) ≈ O(n)

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

Binary Search:

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

Average Case: O(log n)

Worst Case: O(log n) (when the target is not present)

When to Use Each Algorithm:

Linear Search:

Suitable for small datasets where the overhead of sorting is not justified.

Can be used on unsorted lists.

Simple to implement.

Binary Search:

Suitable for large datasets where the cost of sorting can be amortized over multiple searches.

Requires the list to be sorted beforehand.

More efficient for large datasets due to O(log n) time complexity.

For a library management system, linear search is straightforward and works well for small datasets or when the list is unsorted. Binary search, on the other hand, is more efficient for large, sorted datasets, providing faster search times. The choice between the two depends on the size of the dataset and whether it is sorted or not.