**Library Management System**

Linear Search: Linear search is a simple search algorithm that checks each element of the array one by one until the desired element is found or the end of the array is reached.

- Time Complexity:

- Best Case: O(1) (Element found at the first position)

- Average Case: O(n)

- Worst Case: O(n) (Element not found or found at the last position)

- Space Complexity: O(1)

Binary Search : Binary search is an efficient algorithm for finding an element in a sorted array. It works by repeatedly dividing the search interval in half. If the value of the search key is less than the item in the middle of the interval, the search continues in the lower half; otherwise, it continues in the upper half.

- Time Complexity:

- Best Case: O(1) (Element found at the middle)

- Average Case: O(log n)

- Worst Case: O(log n) (Element not found)

- Space Complexity: O(1)

**Analysis**

1. Linear Search

- Time Complexity: O(n) in all cases (best, average, worst).

- Linear search goes through each element one by one, making it less efficient for large datasets.

2. Binary Search

- Time Complexity: O(log n) in all cases (best, average, worst).

- Binary search is much more efficient for large datasets as it divides the search space in half with each step.

**When to Use Each Algorithm**

Linear Search:

- Use linear search when the dataset is small or unsorted.

- Useful when searching for multiple occurrences of the same element.

Binary Search:

- Use binary search when the dataset is large and sorted.

- Provides much faster search times, especially for large datasets.

- Requires the dataset to be sorted, so if the dataset changes frequently, the cost of maintaining the sorted order must be considered.

In conclusion, for large and sorted datasets, binary search is preferred due to its efficiency. For small or unsorted datasets, linear search is simpler and can be more practical.