**Library Management System - Analysis**

**Linear Search**

* **Definition**: Linear search, also known as sequential search, involves scanning each element of the array sequentially until the target element is found or the end of the array is reached.
* **Time Complexity**: O(n)
  + **Best Case**: O(1) (if the target is the first element)
  + **Average Case**: O(n) (when the target is found randomly)
  + **Worst Case**: O(n) (if the target is the last element or not present)

**Binary Search**

* **Definition**: Binary search is an efficient algorithm for finding an item from a sorted array by repeatedly dividing the search interval in half. It compares the target value to the middle element of the array, reducing the search space by half each time.
* **Time Complexity**: O(log n)
  + **Best Case**: O(1) (if the target is the middle element)
  + **Average Case**: O(log n)
  + **Worst Case**: O(log n) (if the target is not present or at one of the ends)

**Analysis**

**Time Complexity Comparison**

* **Linear Search**: O(n) is linear and grows directly with the size of the dataset. It is straightforward but becomes inefficient as the dataset grows larger.
* **Binary Search**: O(log n) is logarithmic and grows much slower compared to linear search. It is highly efficient for large datasets but requires the array to be sorted.

**When to Use Each Algorithm**

* **Linear Search**:
  + **Use When**: The dataset is small or unsorted.
  + **Advantages**: Simplicity; no need for pre-sorting.
  + **Disadvantages**: Inefficient for large datasets due to its linear growth with data size.
* **Binary Search**:
  + **Use When**: The dataset is large and sorted.
  + **Advantages**: Much faster than linear search for large datasets due to its logarithmic time complexity.
  + **Disadvantages**: Requires sorted data and additional time/space for sorting if not already sorted.