Library Management System

1. Understand Search Algorithms

- **Linear Search**:
    - **Description**: Linear search iterates through each element in the list to find the target value. It checks each element until it finds the target or reaches the end of the list.
    - **Time Complexity**:
        - Best Case: O(1) (target is the first element)
        - Average Case: O(n)
        - Worst Case: O(n)
    - **Usage**: Suitable for unsorted lists or small datasets where sorting is not feasible.
- **Binary Search**:
    - **Description**: Binary search works on sorted lists by repeatedly dividing the search interval in half. It compares the target value to the middle element of the list and eliminates half of the list from the search.
    - **Time Complexity**:
        - Best Case: O(1) (target is the middle element)
        - Average Case: O(log n)
        - Worst Case: O(log n)
    - **Usage**: Suitable for large, sorted datasets where the overhead of maintaining a sorted list is justified.

Analysis

- **Time Complexity Comparison**:
    - **Linear Search**:
        - Best Case: O(1)
        - Average/Worst Case: O(n)
    - **Binary Search**:
        - Best Case: O(1)
        - Average/Worst Case: O(log n)
- **Discussion**:
    - **Linear Search**:
        - Advantages: Simple to implement, works on unsorted lists.
        - Disadvantages: Inefficient for large datasets.
        - Usage: When the list is small or unsorted, or when insertion/deletion happens frequently.
    - **Binary Search**:
        - Advantages: Efficient for large datasets.
        - Disadvantages: Requires the list to be sorted, additional overhead for maintaining sorted order.
        - Usage: When the list is large and relatively static, or when search operations are frequent compared to insertions/deletions.