**Exercise 6: Library Management System**

**Linear Search**:

* **Description**: Iterates through each element of the list until it finds the target element or reaches the end.
* **Time Complexity**: O(n)O(n)O(n) where nnn is the number of elements in the list.
* **When to Use**: Suitable for unsorted data or small datasets.

**Binary Search**:

* **Description**: Repeatedly divides the search interval in half. Requires that the list be sorted.
* **Time Complexity**: O(log n)O(\log n)O(log n) where n n n is the number of elements in the list.
* **When to Use**: Efficient for large, sorted datasets.

**Implementation:**

Please refer the code.

**Analysis**

**Linear Search**:

* **Time Complexity**: O(n)O(n)O(n) - In the worst case, it has to check each element.
* **When to Use**: Ideal for small or unsorted datasets where sorting is not feasible.

**Binary Search**:

* **Time Complexity**: O(log⁡n)O(\log n)O(logn) - Much faster than linear search for large datasets if the data is sorted.
* **When to Use**: Efficient for large, sorted datasets. Sorting may be required before performing binary search.

**Advantages and Limitations**:

* **Linear Search**:
  + **Advantages**: Simple to implement, no need for data to be sorted.
  + **Limitations**: Slower for large datasets compared to binary search.
* **Binary Search**:
  + **Advantages**: Much faster for large datasets if sorted.
  + **Limitations**: Requires sorted data, which involves additional time and complexity if the dataset is not sorted.

In summary, use linear search for smaller or unsorted datasets and binary search for larger, sorted datasets to optimize search performance.