# Exercise 6: Library Management System

## 1. Understand Search Algorithms

**Linear Search:**

**- Description:** Scans each element in a list sequentially until the target value is found or the end of the list is reached.

**- Time Complexity:**

- Best Case: O(1) (if the target is the first element in the list)

- Average Case: O(n) (where `n` is the number of elements in the list)

- Worst Case: O(n) (if the target is the last element or not present in the list)

- Space Complexity: O(1) (constant space usage)

**- Characteristics:**

- Simple and easy to implement.

- Can work on unsorted data.

- May be inefficient for large datasets due to its linear time complexity.

**Binary Search:**

- Description: Works on sorted data by repeatedly dividing the search interval in half. It compares the target value with the middle element and eliminates half of the remaining elements from consideration.

**- Time Complexity:**

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

- Average Case: O(log n) (where `n` is the number of elements)

- Worst Case: O(log n) (if the target is not present or is at the end)

- Space Complexity: O(1) for iterative implementation; O(log n) for recursive implementation (due to recursion stack)

**- Characteristics:**

- Much faster for large datasets compared to linear search.

- Requires the data to be sorted before searching.

## 2. Setup

**Create a Class `Book`:**

- Purpose: Represents a book with attributes such as ID, title, and author.

**- Methods:**

- Constructor: Initializes a `Book` instance with a `bookId`, `title`, and `author`.

- Getters and Setters: Provide access to and modification of `bookId`, `title`, and `author`.

- `toString()` Method: Returns a string representation of the book.

## 3. Implementation

**Linear Search Implementation:**

- Purpose: To find a book by its title using linear search.

- Method: `linearSearchByTitle(List<Book> books, String title)`

- Process:

- Iterate through the list of books.

- Compare each book's title with the target title (ignoring case).

- Return the book if a match is found.

- Return `null` if no matching book is found.

**Binary Search Implementation:**

- Purpose: To find a book by its title using binary search (assuming the list is sorted by title).

- Method: `binarySearchByTitle(List<Book> books, String title)`

- Process:

- Initialize `low` and `high` pointers to the start and end of the list, respectively.

- While `low` is less than or equal to `high`:

- Compute the middle index `mid`.

- Compare the middle book's title with the target title.

- If the middle book’s title matches the target, return the book.

- Adjust the `low` or `high` pointers based on the comparison result.

- Return `null` if the book is not found.

## 4. Analysis

**Time Complexity Comparison:**

**- Linear Search:**

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

- Average Case: O(n) (target is somewhere in the middle)

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

- Space Complexity: O(1) (constant space usage)

**- Binary Search:**

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

- Average Case: O(log n) (target is somewhere in the middle, requiring log(n) comparisons)

- Worst Case: O(log n) (target is not present or at the end)

- Space Complexity: O(1) for iterative implementation; O(log n) for recursive implementation (due to recursion stack)

**When to Use Each Algorithm:**

- Linear Search:

- Use When:

- The dataset is small.

- The dataset is unsorted or changes frequently.

- Simplicity and ease of implementation are preferred over performance.

- Advantages:

- Can be used on unsorted data.

- Simple to implement and understand.

**- Binary Search:**

**- Use When:**

- The dataset is large.

- The data is sorted (or can be sorted).

- Faster search performance is required.

**- Advantages:**

- Much faster than linear search for large datasets.

- Efficient for searching in sorted data.

This summary highlights the methods and their implications for search operations in a library management system, detailing when and why each search algorithm might be used.