Library Management System with Linear and Binary Search – EX 6

1. Understanding Search Algorithms

Linear Search:

- **Definition**: Linear search involves checking each element in a list sequentially until the desired element is found or the list ends.
- **Time Complexity**: O(n), where n is the number of elements in the list. This is because each element may need to be checked.

Binary Search:

- **Definition**: Binary search is a more efficient algorithm that can only be used on sorted lists. It works by repeatedly dividing the list in half and comparing the target value with the middle element.
- **Time Complexity**: O(log n), where n is the number of elements in the list. This is due to the division of the list into halves at each step.

2. Setup

Class Definition: Book

The Book class includes attributes such as bookId, title, and author.

```
class Book {
    private String bookId;
    private String title;
    private String author;

    public Book(String bookId, String title, String author) {
        this.bookId = bookId;
        this.title = title;
        this.author = author;
    }

    public String getBookId() {
        return bookId;
    }
```

```
    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    @Override
    public String toString() {
        return "Book ID: " + bookId + ", Title: " + title + ", Author: " + author;
    }
}
```

3. Implementation

Linear Search Implementation

```
class LinearSearch {
    public static int linearSearch(Book[] books, String title) {
        for (int i = 0; i < books.length; i++) {
            if (books[i].getTitle().equalsIgnoreCase(title)) {
                return i;
            }
        }
        return -1;
    }
}
```

Binary Search Implementation

```
class BinarySearch {
    public static int binarySearch(Book[] books, String title) {
```

```
        int left = 0;
        int right = books.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;
            int result = title.compareToIgnoreCase(books[mid].getTitle());

            if (result == 0) {
                return mid;
            }

            if (result > 0) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return -1;
    }
}
```

Sorting Books for Binary Search

To use binary search, the list of books must be sorted.

```
class SortBooks {
    public static void sortBooks(Book[] books) {
        for (int i = 0; i < books.length - 1; i++) {
            for (int j = 0; j < books.length - 1 - i; j++) {
                if (books[j].getTitle().compareToIgnoreCase(books[j + 1].getTitle()) > 0) {
                    Book temp = books[j];
                    books[j] = books[j + 1];
```

```
                books[j + 1] = temp;
            }
        }
    }
}
```

Main Class: LibraryManagementSystem

```
public class LibraryManagementSystem {
    public static void main(String[] args) {
        Book[] books = {
            new Book("1", "Think like a monk", "Jayshetty"),
            new Book("2", "Atomic habits", "James clear")
        };

        Scanner scanner = new Scanner(System.in);

        System.out.println("Choose Search Method:");
        System.out.println("1. Linear Search");
        System.out.println("2. Binary Search");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter the title of the book to search: ");
        String title = scanner.nextLine();

        if (choice == 1) {
            System.out.println("Linear Search:");
            int index = LinearSearch.linearSearch(books, title);
            if (index != -1) {
```

```
            System.out.println("Book found: " + books[index]);
        } else {
            System.out.println("Book not found.");
        }
    } else if (choice == 2) {
        SortBooks.sortBooks(books);

        System.out.println("Binary Search:");
        int index = BinarySearch.binarySearch(books, title);
        if (index != -1) {
            System.out.println("Book found: " + books[index]);
        } else {
            System.out.println("Book not found.");
        }
    } else {
        System.out.println("Invalid choice.");
    }
  }
}
```

4. Analysis

Time Complexity Comparison:

- **Linear Search**: O(n) because it may involve checking each element until the desired one is found.
- **Binary Search**: O(log n) because the list is repeatedly divided in half, making it significantly faster for large, sorted datasets.

When to Use Each Algorithm:

- **Linear Search**: Ideal for small datasets or unsorted data. It can also be used if the dataset size is unknown or if maintaining a sorted list is impractical due to frequent insertions or deletions.
- **Binary Search**: Best suited for large datasets where the data is sorted. The efficiency of binary search makes it preferable for applications requiring frequent and fast lookups.