Exercise 6: Library Management System

1. Understanding the Problem

In a library management system where users can search for books by **title** or **author**, choosing the right search algorithm is essential for speed and efficiency — especially when the library contains **hundreds or thousands of books**.

Some book lists may be **unsorted**, while others might be **alphabetically sorted** by title. Depending on this, we choose between **linear search** and **binary search**.

Why Are Search Algorithms Important?

➤ Quick Access to Book Details

In a large database of books, users expect to find information instantly. The search algorithm determines how quickly the system responds.

➤ Better User Experience

A fast search engine behind the scenes ensures smooth navigation for users who want to look up books by title or author.

➤ Performance as Data Grows

With time, the number of books grows. A poor search method will slow down the system, while the right algorithm keeps it efficient.

2. Understanding Search Algorithms

To implement effective searching, we must understand two basic algorithms:

1. Linear Search

This method checks **each book one by one** until it finds a match.

Works **even if the list is unsorted**.

Simple, but **slow for large lists**.

Example:

```
for (Book b : books) {
    if (b.title.equals("Harry Potter")) {
        return b;
    }
}
```

2. Binary Search

This method works only on a **sorted list**.

It divides the search space into halves repeatedly.

Much faster than linear search.

Example (assuming list is sorted by title):

```
int left = 0, right = books.length - 1;
while (left <= right) {
    int mid = (left + right) / 2;
    int comparison = books[mid].title.compareTo("Harry Potter");
```

```
    if (comparison == 0) {
        return books[mid];
    } else if (comparison < 0) {
        left = mid + 1;
    } else {
        right = mid - 1;
    }
}
```

In binary search, instead of checking every book, we eliminate **half of the remaining books** in each step.

3. When to Use Which Search Algorithm

➤ Use **linear search** when:

The book list is **unsorted**.

The dataset is **small** (fewer than 100 items).

You want a simple solution with minimal setup.

➤ Use **binary search** when:

The list is **sorted** by title or author.

You need **high speed** with a large number of books.

You want better performance in scalable systems.

4. Time Complexity Analysis

➤ Linear Search

Best Case: O(1) (if the book is at the beginning)

Worst Case: O(n) (if the book is at the end or not found)

No need for sorting, but becomes slow as n increases.

➤ Binary Search

Best Case: O(1) (if the book is at the middle)

Worst Case: O(log n) (log base 2 of number of books)

Requires the list to be **pre-sorted**, but much faster for large n.

5. Optimizations and Best Practices

Keep Lists Sorted (if possible)

If your system frequently searches by title, it's good to maintain a sorted list. This allows you to use binary search consistently.

Use Case-Based Logic

Check if the data is sorted before deciding on a search method. Use linear search as a fallback.

Encapsulate Search Logic

Place search code in reusable functions like findBookByTitle(String title) for clean and maintainable code.

Preprocess Frequently Searched Data

If users often search for popular books or authors, cache their positions or use indexing.