# Exercise 6: Library Management System

### **Understand Search Algorithms**

**Linear Search:**

* **Description:** Linear search involves checking each element in the list sequentially until the desired element is found or the list ends.
* **Time Complexity:**
  + Best Case: O(1)O(1)O(1) (if the element is the first one)
  + Average Case: O(n)O(n)O(n)
  + Worst Case: O(n)O(n)O(n)

**Binary Search:**

* **Description:** Binary search is an efficient algorithm for finding an item from a sorted list of items. It repeatedly divides the search interval in half. If the value of the search key is less than the item in the middle of the interval, it narrows the interval to the lower half. Otherwise, it narrows it to the upper half. The process continues until the value is found or the interval is empty.
* **Time Complexity:**
  + Best Case: O(1)O(1)O(1) (if the middle element is the target)
  + Average Case: O(log⁡n)O(\log n)O(logn)
  + Worst Case: O(log⁡n)O(\log n)O(logn)

### **Step 2: Setup**

**Book Class:**

class Book {

int bookId;

String title;

String author;

Book(int bookId, String title, String author) {

this.bookId = bookId;

this.title = title;

this.author = author;

}

// Getters

public int getBookId() {

return bookId;

}

public String getTitle() {

return title;

}

public String getAuthor() {

return author;

}

@Override

public String toString() {

return "Book [bookId=" + bookId + ", title=" + title + ", author=" + author + "]";

}

}

### **Step 3: Implementation**

**Linear Search to Find Books by Title:**

public class LibraryManagementSystem {

private Book[] books;

private int size;

public LibraryManagementSystem(int capacity) {

books = new Book[capacity];

size = 0;

}

// Method to add a book

public void addBook(Book book) {

if (size < books.length) {

books[size++] = book;

} else {

System.out.println("Library is full.");

}

}

// Method to search for books by title using linear search

public Book linearSearchByTitle(String title) {

for (int i = 0; i < size; i++) {

if (books[i].getTitle().equalsIgnoreCase(title)) {

return books[i];

}

}

return null; // Book not found

}

// Method to display all books

public void displayBooks() {

for (int i = 0; i < size; i++) {

System.out.println(books[i]);

}

}

}

**Binary Search to Find Books by Title (Assuming the List is Sorted):**

import java.util.Arrays;

public class LibraryManagementSystem {

private Book[] books;

private int size;

public LibraryManagementSystem(int capacity) {

books = new Book[capacity];

size = 0;

}

// Method to add a book

public void addBook(Book book) {

if (size < books.length) {

books[size++] = book;

Arrays.sort(books, 0, size, (b1, b2) -> b1.getTitle().compareToIgnoreCase(b2.getTitle()));

} else {

System.out.println("Library is full.");

}

}

// Method to search for books by title using binary search

public Book binarySearchByTitle(String title) {

int left = 0;

int right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

int compareResult = books[mid].getTitle().compareToIgnoreCase(title);

if (compareResult == 0) {

return books[mid];

} else if (compareResult < 0) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return null; // Book not found

}

// Method to display all books

public void displayBooks() {

for (int i = 0; i < size; i++) {

System.out.println(books[i]);

}

}

}

### **Step 4: Analysis**

**Time Complexity:**

* **Linear Search:**
  + Best Case: O(1)O(1)O(1) (if the book is the first one)
  + Average Case: O(n)O(n)O(n)
  + Worst Case: O(n)O(n)O(n)
* **Binary Search:**
  + Best Case: O(1)O(1)O(1) (if the middle element is the target)
  + Average Case: O(log⁡n)O(\log n)O(logn)
  + Worst Case: O(log⁡n)O(\log n)O(logn)

**Comparison and Suitability:**

* **Linear Search:**
  + Suitable for small datasets or unsorted data.
  + Simple to implement and does not require sorting.
  + Performance degrades linearly with the increase in data size.
* **Binary Search:**
  + Suitable for large datasets that are sorted.
  + Much more efficient than linear search for large datasets due to logarithmic time complexity.
  + Requires the dataset to be sorted, which may involve additional overhead for sorting.

**Conclusion:** For the library management system, if the list of books is small or unsorted, **linear search** can be used due to its simplicity and no requirement for sorting. However, for large, sorted lists, **binary search** is preferable due to its much faster search time, making it ideal for scenarios where efficient search operations are crucial.