**Exercise 6: Library Management System**

**Understanding Search Algorithms**

**🔹 Linear Search**

Linear search checks each element in the list **one by one** until it finds the required item or reaches the end.

* Simple and works on **unsorted data**.
* Not efficient for **large datasets**.
* Used when data is small or rarely searched.

**Time Complexity**:  
Best Case – O(1), Worst Case – O(n)

**🔹 Binary Search**

Binary search works only when the list is **sorted**. It compares the middle element with the target and eliminates half of the data on each step.

* Faster for **large, sorted datasets**.
* Cannot be used on **unsorted data**.

**Time Complexity**:  
Best Case – O(1), Worst Case – O(log n)

**Source Code:**

**Book.java**

public class Book {

private final 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 bookId + " | " + title + " | " + author;

}

}

**LibraryManager.java**

import java.util.Arrays;

import java.util.Comparator;

public class LibraryManager {

private Book[] books;

private int count;

public LibraryManager(int size) {

books = new Book[size];

count = 0;

}

public void addBook(Book book) {

if (count < books.length) books[count++] = book;

}

public Book linearSearch(String title) {

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

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

return books[i];

}

}

return null;

}

public Book binarySearch(String title) {

Arrays.sort(books, 0, count, Comparator.comparing(Book::getTitle, String.CASE\_INSENSITIVE\_ORDER));

int low = 0, high = count - 1;

while (low <= high) {

int mid = (low + high) / 2;

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

if (cmp == 0) return books[mid];

else if (cmp < 0) low = mid + 1;

else high = mid - 1;

}

return null;

}

public void listBooks() {

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

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

}

}

}

**Demo.java**

public class Demo {

public static void main(String[] args) {

LibraryManager library = new LibraryManager(10);

library.addBook(new Book("B001", "The Hobbit", "J.R.R. Tolkien"));

library.addBook(new Book("B002", "1984", "George Orwell"));

library.addBook(new Book("B003", "Clean Code", "Robert C. Martin"));

library.addBook(new Book("B004", "The Alchemist", "Paulo Coelho"));

System.out.println("All books:");

library.listBooks();

System.out.println("\nLinear Search for '1984':");

Book result1 = library.linearSearch("1984");

System.out.println(result1 != null ? result1 : "Not found");

System.out.println("\nBinary Search for 'Clean Code':");

Book result2 = library.binarySearch("Clean Code");

System.out.println(result2 != null ? result2 : "Not found");

}

}

**Output:**

**A computer screen shot of a computer

AI-generated content may be incorrect.**

**Analysis of Linear vs Binary Search**

**🔹 Time Complexity**

* **Linear Search**:
  + Simple but **slow** for large lists
  + Checks every element → O(n)
* **Binary Search**:
  + Fast, but works only on **sorted lists**
  + Divides list each step → O(log n)

**🔹 When to Use**

* Use **linear search** when the list is **small** or **unsorted**.
* Use **binary search** when the list is **sorted** and **large**.