**Exercise 6: Library Management System**

**Steps: Understand Search Algorithms**

**Linear Search**

Linear search is a straightforward search algorithm that examines each element in a list sequentially until it finds the desired element or reaches the end of the list.

* **Time Complexity:** O(n) in the worst case, where n is the number of elements in the list.
* **Best Case:** O(1), when the desired element is the first one in the list.
* **Worst Case:** O(n), when the desired element is the last one or not present in the list.
* **Advantages:** Simple to implement, does not require the list to be sorted.
* **Disadvantages:** Inefficient for large lists due to its linear time complexity.

**Binary Search**

Binary search is a more efficient search algorithm that operates on sorted lists. It repeatedly divides the list in half, comparing the target element with the middle element, and eliminates half of the list until it finds the target element or the list cannot be divided further.

* **Time Complexity:** O(log n) in the worst case, where n is the number of elements in the list.
* **Best Case:** O(1), when the middle element is the target element.
* **Worst Case:** O(log n), when the search goes through all levels of division.
* **Advantages:** Much more efficient for large lists compared to linear search.
* **Disadvantages:** Requires the list to be sorted.

**Setup**

Create a class Book with attributes like bookId, title, and author.

public 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; }

// Getters

public String getBookId() {

return bookId; }

public String getTitle() {

return title;}

public String getAuthor() {

return author;}

}

**Implementation**

**Linear Search to Find Books by Title**

public class LibraryManagement {

private Book[] books;

public LibraryManagement(Book[] books) {

this.books = books;}

public Book linearSearchByTitle(String title) {

for (Book book : books) {

if (book.getTitle().equalsIgnoreCase(title)) {

return book;}}

return null;

}

public static void main(String[] args) {

Book[] books = {

new Book("001", "The Great Gatsby", "F. Scott Fitzgerald"),

new Book("002", "1984", "George Orwell"),

new Book("003", "To Kill a Mockingbird", "Harper Lee"),

new Book("004", "Moby-Dick", "Herman Melville")

};

LibraryManagement library = new LibraryManagement(books);

// Linear search

String searchTitle = "1984";

Book foundBook = library.linearSearchByTitle(searchTitle);

if (foundBook != null) {

System.out.println("Book found: " + foundBook.getTitle() + " by " + foundBook.getAuthor());

} else {

System.out.println("Book not found.");}

}

}

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

import java.util.Arrays;

import java.util.Comparator;

public class LibraryManagement {

private Book[] books;

public LibraryManagement(Book[] books) {

this.books = books;}

public Book linearSearchByTitle(String title) {

for (Book book : books) {

if (book.getTitle().equalsIgnoreCase(title)) {

return book;}}

return null;

}

public Book binarySearchByTitle(String title) {

int left = 0;

int right = books.length - 1;

while (left <= right) {

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

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

if (comparison == 0) {

return books[mid];

} else if (comparison < 0) {

left = mid + 1;

} else {

right = mid – 1; }}

return null;}

public static void main(String[] args) {

Book[] books = {

new Book("001", "1984", "George Orwell"),

new Book("002", "Moby-Dick", "Herman Melville"),

new Book("003", "The Great Gatsby", "F. Scott Fitzgerald"),

new Book("004", "To Kill a Mockingbird", "Harper Lee")

};

// Sort books by title for binary search

Arrays.sort(books, Comparator.comparing(Book::getTitle));

LibraryManagement library = new LibraryManagement(books);

// Binary search

String searchTitle = "1984";

Book foundBook = library.binarySearchByTitle(searchTitle);

if (foundBook != null) {

System.out.println("Book found: " + foundBook.getTitle() + " by " + foundBook.getAuthor());

} else {

System.out.println("Book not found."); }}}

**Step 4: Analysis**

**Time Complexity Comparison**

* **Linear Search:**
  + **Best Case:** O(1) - The desired book is the first element.
  + **Average Case:** O(n) - The desired book is somewhere in the middle.
  + **Worst Case:** O(n) - The desired book is the last element or not present.
* **Binary Search:**
  + **Best Case:** O(1) - The middle element is the desired book.
  + **Average Case:** O(log n) - The search space is halved each time.
  + **Worst Case:** O(log n) - The desired book is at the last possible position or not present.

**When to Use Each Algorithm**

* **Linear Search:**
  + Suitable for small datasets or when the list is not sorted.
  + Simple and requires no preprocessing of the list.
* **Binary Search:**
  + More efficient for large datasets but requires the list to be sorted.
  + The sorting operation can be costly, but the efficiency of the search makes up for it in the long run.