**DATASTRUCTURES AND ALGORITHMS:**

**Exercise-6: Library Management System**

**Understand Search Algorithms**

**1. Linear Search:**  
Linear search is the most basic type of search. It checks each element in the list one by one from the beginning to the end until the target value is found or the list ends.

* **Works on:** Unsorted or sorted data.
* **Time Complexity:**
  + Best Case: O(1) (if the element is at the beginning)
  + Worst Case: O(n) (if at the end or not found)
* **Example:** Searching for a book titled *"Java Programming"* will go through every book in the list until it finds a match.

**2. Binary Search:**  
Binary search is an efficient method that works only on **sorted** data. It divides the list into halves and checks the middle element. Based on comparison, it continues searching in one half only.

* **Works on:** Sorted data only.
* **Time Complexity:**
  + Best Case: O(1)
  + Worst Case: O(log n)
* **Example:** Searching for *"Python Guide"* in a sorted list of book titles uses the midpoint logic to reduce comparisons.

**Book.java**

class Book {

int bookId;

String title;

String author;

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

this.bookId = bookId;

this.title = title;

this.author = author;

}

void display() {

System.out.println("ID: " + bookId + ", Title: " + title + ", Author: " + author);

}

}

**Library.java**

import java.util.Arrays;

import java.util.Comparator;

public class Library {

Book[] books;

int count;

public Library(int size) {

books = new Book[size];

count = 0;

}

// Add book to array

public void addBook(Book b) {

if (count < books.length) {

books[count++] = b;

} else {

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

}

}

// Linear search by title

public Book linearSearch(String title) {

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

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

return books[i];

}

}

return null;

}

// Binary search by title (only after sorting)

public Book binarySearch(String title) {

Arrays.sort(books, 0, count, Comparator.comparing(b -> b.title.toLowerCase()));

int left = 0;

int right = count - 1;

while (left <= right) {

int mid = (left + right) / 2;

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

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

else if (result < 0) left = mid + 1;

else right = mid - 1;

}

return null;

}

// Display all books

public void displayBooks() {

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

books[i].display();

}

}

public static void main(String[] args) {

Library library = new Library(5);

library.addBook(new Book(1, "C Programming", "Dennis Ritchie"));

library.addBook(new Book(2, "Java Programming", "James Gosling"));

library.addBook(new Book(3, "Python Guide", "Guido van Rossum"));

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

library.displayBooks();

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

Book found1 = library.linearSearch("Java Programming");

if (found1 != null) found1.display();

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

System.out.println("\nBinary Search for 'Python Guide':");

Book found2 = library.binarySearch("Python Guide");

if (found2 != null) found2.display();

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

}

}

**Step 4: Analysis**

| **Search Type** | **Time Complexity** | **Works On** | **Best Used When** |
| --- | --- | --- | --- |
| Linear Search | O(n) | Sorted or unsorted | Data is small or not sorted |
| Binary Search | O(log n) | Sorted only | Large dataset and already sorted |

**When to Use What?**

* Use **Linear Search** if:
  + The list is small
  + The list is **unsorted**
  + Simplicity is more important than speed
* Use **Binary Search** if:
  + The list is **large**
  + It is **sorted**
  + Speed and performance are priorities