Exercise 6: Library Management System

Search Algorithms

Linear Search

Scans each element in a list.

Works even when the list is unsorted. Time Complexity: O(n)

Binary Search

Works only on sorted data.

Divides the search interval in half each time. Time Complexity: O(log n)

Java Code

import java.util.Arrays; import java.util.Scanner;

class Book implements Comparable<Book> { int bookId;

String title; String author;

public Book(int bookId, String title, String author) { this.bookId = bookId;

this.title = title; this.author = author;

}

public String toString() {

return "ID: " + bookId + ", Title: " + title + ", Author: " + author;

}

@Override

public int compareTo(Book other) {

return this.title.compareToIgnoreCase(other.title);

}

}

public class LibraryManagementSystem { static Book[] books = new Book[5];

static Scanner sc = new Scanner(System.in);

public static void main(String[] args) {

// Adding sample books

books[0] = new Book(101, "The Alchemist", "Paulo Coelho"); books[1] = new Book(102, "Clean Code", "Robert C. Martin"); books[2] = new Book(103, "Java: The Complete Reference", "Herbert

Schildt");

books[3] = new Book(104, "The Pragmatic Programmer", "Andrew Hunt"); books[4] = new Book(105, "Introduction to Algorithms", "CLRS");

while (true) {

System.out.println("\n1. Linear Search by Title"); System.out.println("2. Binary Search by Title"); System.out.println("3. Display All Books"); System.out.println("4. Exit"); System.out.print("Choose option: ");

int choice = sc.nextInt(); sc.nextLine(); // consume newline

switch (choice) {

case 1 -> linearSearch(); case 2 -> binarySearch(); case 3 -> displayBooks(); case 4 -> System.exit(0);

}

}

}

public static void linearSearch() { System.out.print("Enter title to search: "); String title = sc.nextLine();

boolean found = false; for (Book b : books) {

if (b.title.equalsIgnoreCase(title)) { System.out.println("Found: " + b); found = true;

break;

}

}

if (!found) System.out.println("Book not found.");

}

public static void binarySearch() { Arrays.sort(books); // sort by title System.out.print("Enter title to search: "); String title = sc.nextLine();

int low = 0, high = books.length - 1; while (low <= high) {

int mid = (low + high) / 2;

int cmp = books[mid].title.compareToIgnoreCase(title); if (cmp == 0) {

System.out.println("Found: " + books[mid]); return;

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

} else {

high = mid - 1;

}

}

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

}

public static void displayBooks() { System.out.println("\nLibrary Books:"); for (Book b : books) {

System.out.println(b);

}

}

}

Analysis

Algorithm Time Complexity Sorted Required? Use Case

Linear Search O(n) datasets

Binary Search O(log n)

✅ Yes

❌ No

Small or unsorted Large and sorted datasets

✅Use Linear Search when:

Data is unsorted List is small

✅Use Binary Search when:

Data is sorted

You need fast search on large datasets

▶ Option 1: Linear Search by Title

Input:

Enter title to search: Clean Code Output:

Found: ID: 102, Title: Clean Code, Author: Robert C. Martin

▶ Option 2: Binary Search by Title

Input:

Enter title to search: The Pragmatic Programmer Output:

Found: ID: 104, Title: The Pragmatic Programmer, Author: Andrew Hunt

▶ Option 3: Display All Books

Output:

Library Books:

ID: 101, Title: The Alchemist, Author: Paulo Coelho ID: 102, Title: Clean Code, Author: Robert C. Martin

ID: 103, Title: Java: The Complete Reference, Author: Herbert Schildt ID: 104, Title: The Pragmatic Programmer, Author: Andrew Hunt

ID: 105, Title: Introduction to Algorithms, Author: CLRS

▶ Option 1/2 with Nonexistent Book

Input:

Enter title to search: Python Programming Output:

Book not found.