**Module 2 - Data Structures and Algorithms**

**Exercise 6: Library Management System**

**1. Understand Search Algorithms**

**Linear Search:**

* **How it works:** Scans each element in the list sequentially until it finds the target.
* **Time Complexity:**
  + Best Case: O(1)
  + Average/Worst Case: O(n)
* **Use Case:** Works on unsorted or small datasets.

**Binary Search:**

* **How it works:** Repeatedly divides the sorted list in half and checks the middle value.
* **Time Complexity:**
  + Best Case: O(1)
  + Average/Worst Case: O(log n)
* **Use Case:** Only works on **s**orted datasets and is efficient for large datasets.

**2,3). Setup and Implementation:**

**Code:**

import java.util.Arrays;

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 int compareTo(Book other) {

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

    }

}

public class Library {

    static Book[] books = {

        new Book(101, "Java Basics", "Sundar"),

        new Book(102, "Data Structures", "Vikas"),

        new Book(103, "Algorithms", "Ravi")

    };

    public static void linearSearch(String title) {

        for (Book b : books) {

            if (b.title.equals(title)) {

                System.out.println(b.bookId + " " + b.title + " " + b.author);

                return;

            }

        }

    }

    public static void binarySearch(String title) {

        Arrays.sort(books);

        int low = 0, high = books.length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

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

            if (cmp == 0) {

                System.out.println(books[mid].bookId + " " + books[mid].title + " " + books[mid].author);

                return;

            } else if (cmp < 0) {

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

    }

    public static void main(String[] args) {

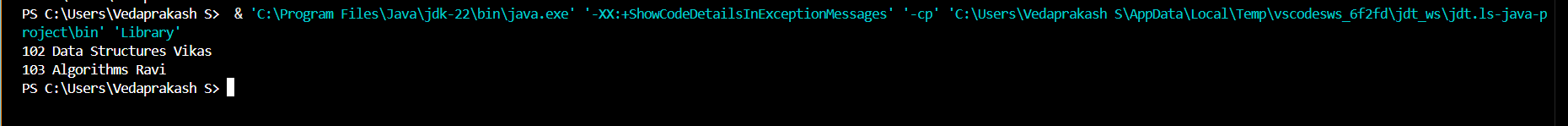
        linearSearch("Data Structures");

        binarySearch("Algorithms");

    }

}

**Output:**



**4. Analysis and Comparison**

| **Criteria** | **Linear Search** | **Binary Search** |
| --- | --- | --- |
| **Time Complexity** | O(n) | O(log n) |
| **Sorted Required** | No | Yes |
| **Performance** | Slower on large data | Much faster on large sorted data |
| **Flexibility** | Can be used on any list | Needs pre-sorting |

**When to Use What:**

* **Linear Search:** Small datasets, unsorted data, single/few queries.
* **Binary Search:** Large datasets, already sorted data, multiple queries.