**ALGORITHMS DATASTRUCTURES**

**Exercise 6: Library Management System**

**Scenario:**

You are developing a library management system where users can search for books by title or author.

**Steps:**

1. **Understand Search Algorithms:**
   * Explain linear search and binary search algorithms.
2. **Setup:**
   * Create a class Book with attributes like bookId, title, and author.
3. **Implementation:**
   * Implement linear search to find books by title.
   * Implement binary search to find books by title (assuming the list is sorted).
4. **Analysis:**
   * Compare the time complexity of linear and binary search.
   * Discuss when to use each algorithm based on the data set size and order.

**IMPLEMENTATION BY LINEAR SEARCH:**

class 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;

}

}

public class LibraryLinearSearch {

public static void main(String[] args) {

Book[] books = {

new Book(1, "Java Programming", "James"),

new Book(2, "Python Basics", "Guido"),

new Book(3, "C Language", "Dennis"),

new Book(4, "Data Structures", "Narasimha"),

new Book(5, "Operating Systems", "Silberschatz")

};

System.out.println("Linear Search for 'C Language':");

linearSearch(books, "C Language");

}

static void linearSearch(Book[] books, String targetTitle) {

for (Book book : books) {

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

System.out.println(book);

return;

}

}

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

}

}

**OUTPUT:**

****

**IMPLEMENTATION BY BINARY SEARCH:**

import java.util.Arrays;

import java.util.Comparator;

class 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;

}

}

public class LibraryBinarySearch {

public static void main(String[] args) {

Book[] books = {

new Book(1, "Java Programming", "James"),

new Book(2, "Python Basics", "Guido"),

new Book(3, "C Language", "Dennis"),

new Book(4, "Data Structures", "Narasimha"),

new Book(5, "Operating Systems", "Silberschatz")

};

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

System.out.println("Binary Search for 'Python Basics':");

binarySearch(books, "Python Basics");

}

static void binarySearch(Book[] books, String targetTitle) {

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

while (low <= high) {

int mid = (low + high) / 2;

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

if (cmp == 0) {

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

return;

} else if (cmp < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

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

}

}

**OUTPUT:**

****

**ANALYSIS:**

**Linear Search:**

* Best Case: O(1) (match at first position)
* Worst/Average Case: O(n) (match at end or not found)
* Works on unsorted data

**Binary Search:**

* Best Case: O(1) (match at middle)
* Worst/Average Case: O(log n)
* Requires data to be sorted

**WHEN TO USE:**

* **Linear Search:**
  + Use for small datasets
  + Use when data is unsorted
  + Suitable for frequent insertions/deletions
* **Binary Search:**
  + Use for large datasets
  + Only when data is sorted
  + Best for fast and repeated lookups in static data