**Exercise – 6 Library Management System**

**Linear Search Algorithm**

Linear search is a simple search technique where each element of the list is checked one by one until the target value is found or the end of the list is reached.

Steps:

1. Start from the first element.

2. Compare each element with the target.

3. If a match is found, return the positon

4. If the end is reached and no match is found, return -1 (not found).

**Binary Search Algorithm**

Binary search is a highly efficient search algorithm that works on **sorted arrays** by repeatedly dividing the search range in half.

Steps:

1. Start in the middle element of the sorted array

2. If it matches the target, return the position.

3. If the target is less than the middle, search the left half.

4. If greater, search the right half.

5. Repeat until found or range becomes empty.

**Main.java**

import java.util.\*;

public class Main{

    public static void main(String[] args){

        Book[] books = {

            new Book(101, "Java", "James Gosling"),

            new Book(102, "C","Dennis Ritchie"),

            new Book(103, "Python", "Guido van Rossum"),

            new Book(104,"JavaScript","Brendan Eich"),

            new Book(105, "C++","Bjarne Stroustrup")

        };

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter book title to search (linear search)");

        String t1 = sc.nextLine();

        linearSearch(books, t1);

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

        System.out.println("Enter book title to search (binary search)");

        String t2 = sc.nextLine();

        int idx = binarySearch(books, t2);

        if(idx!=-1)

        System.out.println("Book found");

        else

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

        sc.close();

    }

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

        boolean found = false;

        for(Book b: books){

            if(b.title.equals(t)){

                System.out.println("Book found");

                found = true;

                break;

            }

        }

        if(!found)

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

    }

    static int binarySearch(Book[] books, String t){

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

        while(low<=high){

            int mid = (low+high)/2;

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

            if(compare == 0)

            return mid;

            else if(compare<0)

            low = mid+1;

            else

            high = mid-1;

        }

        return -1;

    }

}

class Book{

    int bookId;

    String title;

    String author;

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

        this.bookId = bookId;

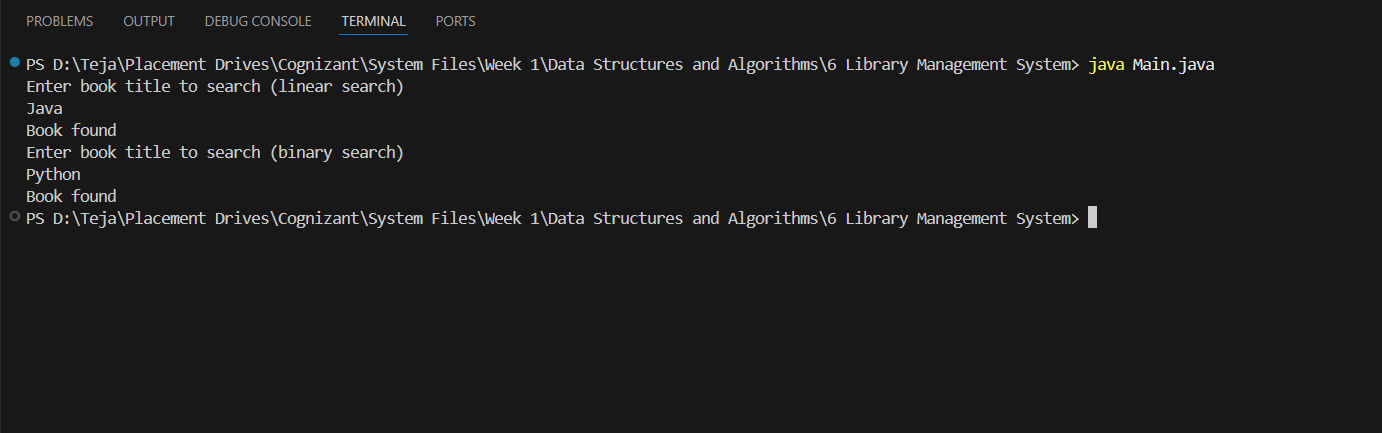
        this.title = title;

        this.author = author;

    }

}

**Output**



**Comparison of Time Complexities of Linear and Binary Search**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** | **Data Requirements** |
| Linear Search | O(1) | O(n) | O(n) | Works on unsorted data. |
| Binary Search | O(1) | O(nlogn) | O(nlogn) | Requires sorted data. |

**Use Case of Linear Search:**

1. The data is unsorted or randomly ordered

2. The data set is small

3. Simplicity is more important than performance

**Use case of Binary Search**

1. The data is sorted in either of ascending or descending order.

2. The data set is large and performance matters

3. Fast searching is required repeatedly on static data.