**Exercise 6: Library Management System**

1. Explain linear search and binary search algorithms.

**Linear Search:**

* **Description:** Linear search involves checking each element in the list sequentially until the desired element is found or the list ends.
* **Time Complexity:**
  + Best Case: O(1) (if the element is the first one)
  + Average Case: O(n)
  + Worst Case: O(n)

**Binary Search:**

* **Description:** Binary search is an efficient algorithm for finding an item from a sorted list of items. It repeatedly divides the search interval in half. If the value of the search key is less than the item in the middle of the interval, it narrows the interval to the lower half. Otherwise, it narrows it to the upper half. The process continues until the value is found or the interval is empty.
* **Time Complexity:**
  + Best Case: O(1) (if the middle element is the target)
  + Average Case: O(logn)
  + Worst Case: O(logn)

1. Create a class **Book** with attributes like **bookId**, **title**, and **author**.

public class Book {

private int id;

private String name;

private String author;

Book(int id,String name,String author){

this.id = id;

this.name = name;

this.author = author;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public String toString(){

return this.id + " " + this.name + " " + this.author;

}

}

1. Implement linear search to find books by title and Implement binary search to find books by title (assuming the list is sorted).

public class Search {

public Book linearSearch(Book[] books,String title){

for(Book book : books){

if(book.getName().toLowerCase().compareTo(title.toLowerCase()) == 0)

return book;

}

return null;

}

public Book binarySearch(Book[] sortedArray,String title){

int left = 0;

int right = sortedArray.length - 1;

int mid = -1;

while (left <= right){

mid = left + ((right - left) / 2);

String currTitle = sortedArray[mid].getName();

int res = currTitle.toLowerCase().compareTo(title.toLowerCase());

if(res == 0)

return sortedArray[mid];

else if(res < 0)

left = mid + 1;

else

right = mid - 1;

}

return null;

}

}

1. Analysis

**Time Complexity:**

* **Linear Search:**
  + Best Case: O(1) (if the book is the first one)
  + Average Case: O(n)
  + Worst Case: O(n)
* **Binary Search:**
  + Best Case: O(1) (if the middle element is the target)
  + Average Case: O(logn)
  + Worst Case: O(logn)

1. Comparison and Suitability:

* **Linear Search:**
  + Suitable for small datasets or unsorted data.
  + Simple to implement and does not require sorting.
  + Performance degrades linearly with the increase in data size.
* **Binary Search:**
  + Suitable for large datasets that are sorted.
  + Much more efficient than linear search for large datasets due to logarithmic time complexity.
  + Requires the dataset to be sorted, which may involve additional overhead for sorting.