**Exercise 6: Library Management System**

Explain linear search and binary search algorithms.

Linear Search

Linear search is a simple searching algorithm that works by iterating through each element in the dataset one by one, checking if the target value matches the current element. If a match is found, the algorithm returns the index of the element. If no match is found after iterating through all elements, the algorithm returns a failure indicator.

How Linear Search Works

1. Start at the first element of the dataset.

2. Compare the target value with the current element.

3. If the values match, return the index of the current element.

4. If the values don't match, move to the next element and repeat steps 2-3.

5. If the end of the dataset is reached without finding a match, return a failure indicator.

Time Complexity

The time complexity of linear search is O(n), where n is the number of elements in the dataset. This means that the algorithm's running time increases linearly with the size of the dataset

Binary Search

Binary search is a more efficient searching algorithm that works by dividing the dataset in half with each iteration, searching for the target value in one of the two halves. This process continues until the target value is found or the dataset is exhausted.

How Binary Search Works

1. Sort the dataset in ascending order (if not already sorted).

2. Find the middle element of the dataset.

3. Compare the target value with the middle element.

4. If the values match, return the index of the middle element.

5. If the target value is less than the middle element, repeat steps 2-4 with the left half of the dataset.

6. If the target value is greater than the middle element, repeat steps 2-4 with the right half of the dataset.

7. If the dataset is exhausted without finding a match, return a failure indicator.

Time Complexity

The time complexity of binary search is O(log n), where n is the number of elements in the dataset. This means that the algorithm's running time increases logarithmically with the size of the dataset, making it much faster than linear search for large datasets.

//Java code

public class Book {

private int bookId;

private String title;

private String author;

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

this.bookId = bookId;

this.title = title;

this.author = author;

}

// Getters and setters

}

public class LinearSearch {

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

for (Book book : books) {

if (book.getTitle().equals(title)) {

return book;

}

}

return null;

}

}

public class BinarySearch {

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

Arrays.sort(books, Comparator.comparing(Book::getTitle));

int low = 0;

int high = books.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

if (books[mid].getTitle().equals(title)) {

return books[mid];

} else if (books[mid].getTitle().compareTo(title) < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return null;

}

}

Analysis

Compare the time complexity of linear and binary search.

- Linear Search:

- Time complexity: O(n)

- Suitable for small datasets or when the data is unsorted

- Binary Search:

- Time complexity: O(log n)

- Suitable for large datasets or when the data is sorted

Discuss when to use each algorithm based on the data set size and order.

When to use each algorithm:

- Linear Search:

- When the dataset is small (e.g., fewer than 100 elements)

- When the data is unsorted or cannot be sorted

- Binary Search:

- When the dataset is large (e.g., thousands or millions of elements)

- When the data is sorted or can be sorted