**Exercise 6: Library Management System**

* + **Explain linear search and binary search algorithms.**

**Linear Search**

**Linear Search** is a straightforward search algorithm that checks each element in a list sequentially until the target element is found or the end of the list is reached.

**Algorithm:**

1. Start from the first element of the list.
2. Compare the current element with the target element.
3. If the current element matches the target, return its index.
4. If the current element does not match, move to the next element.
5. Repeat steps 2-4 until the target element is found or the end of the list is reached.

**Time Complexity:**

* **Best Case**: O(1) (when the target element is the first element)
* **Average Case**: O(n)
* **Worst Case**: O(n) (when the target element is the last element or not present)

**Binary Search**

**Binary Search** is a more efficient search algorithm that works on sorted arrays. It repeatedly divides the search interval in half, comparing the target element to the middle element of the current interval.

**Algorithm:**

1. Start with the entire array as the search interval.
2. Find the middle element of the current interval.
3. Compare the middle element with the target element.
4. If the middle element matches the target, return its index.
5. If the middle element is greater than the target, repeat the search on the left half of the interval.
6. If the middle element is less than the target, repeat the search on the right half of the interval.
7. Repeat steps 2-6 until the target element is found or the interval is empty.

**Time Complexity:**

* **Best Case**: O(1) (when the target element is the middle element)
* **Average Case**: O(log n)
* **Worst Case**: O(log n)
  + **Compare the time complexity of linear and binary search.**
  + **Discuss when to use each algorithm based on the data set size and order.**

**Time Complexity Comparison: Linear Search vs. Binary Search**

**Linear Search**

* **Best Case**: O(1) - When the target element is the first element in the list.
* **Average Case**: O(n) - On average, the search will check half of the elements.
* **Worst Case**: O(n) - When the target element is the last element or not present in the list.

**Binary Search**

* **Best Case**: O(1) - When the target element is the middle element.
* **Average Case**: O(log n) - The search space is halved with each step.
* **Worst Case**: O(log n) - When the target element is not present, and the search space is reduced to zero.

**When to Use Each Algorithm**

**Linear Search**

* **Unsorted Data**: Linear search is suitable for unsorted datasets because it does not require any prior ordering of elements.
* **Small Data Sets**: For small datasets, the simplicity of linear search can be advantageous, as the overhead of sorting the data for binary search may not be justified.
* **Single or Few Searches**: If you only need to perform a single search or a few searches, linear search can be more efficient due to its simplicity and lack of setup requirements.

**Binary Search**

* **Sorted Data**: Binary search requires the dataset to be sorted. It is highly efficient for large, sorted datasets.
* **Large Data Sets**: For large datasets, binary search is much faster due to its logarithmic time complexity.
* **Frequent Searches**: If you need to perform multiple searches on the same dataset, it is worth sorting the data once and then using binary search for efficient lookups.