**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.

**Concept**

**Linear Search**

**Algorithm:**

* **Definition:** Linear search is a straightforward search algorithm that checks each element of the list sequentially until the desired element is found or the list ends.
* **Steps:**
  1. Start from the first element of the list.
  2. Compare the target value with the current element.
  3. If the target value matches the current element, return the index or the element.
  4. If the target value does not match, move to the next element.
  5. Repeat steps 2-4 until the target value is found or the end of the list is reached.

**Time Complexity:**

* **Best Case:** O(1) (When the target element is at the beginning of the list)
* **Worst Case:** O(n) (When the target element is at the end of the list or not present in the list)
* **Average Case:** O(n) (On average, the target element might be in the middle of the list)

**Binary Search**

**Algorithm:**

* **Definition:** Binary search is an efficient search algorithm that works on sorted lists. It repeatedly divides the list into halves to find the target element.
* **Steps:**
  1. Start with the entire list.
  2. Find the middle element of the list.
  3. Compare the target value with the middle element.
  4. If the target value matches the middle element, return the index or the element.
  5. If the target value is less than the middle element, repeat the search on the left half of the list.
  6. If the target value is greater than the middle element, repeat the search on the right half of the list.
  7. Repeat steps 2-6 until the target value is found or the sublist is empty.

**Time Complexity:**

* **Best Case:** O(1) (When the target element is at the middle of the list on the first check)
* **Worst Case:** O(log n) (When the list is repeatedly divided until the target element is found or the sublist is empty)
* **Average Case:** O(log n) (On average, the target element will be found after log n comparisons)

**Comparison of Time Complexity**

* **Linear Search:**
  + **Best Case:** O(1)
  + **Worst Case:** O(n)
  + **Average Case:** O(n)
* **Binary Search:**
  + **Best Case:** O(1)
  + **Worst Case:** O(log n)
  + **Average Case:** O(log n)

**When to Use Each Algorithm**

**Linear Search:**

* **Data Set Size:** Suitable for small data sets where the time complexity of O(n) is acceptable.
* **Order:** Can be used on unsorted lists since it does not require any specific order.
* **Use Case:** Ideal for scenarios where the list is small, unsorted, or constantly changing, and sorting the list is not practical.

**Binary Search:**

* **Data Set Size:** Suitable for large data sets where the time complexity of O(log n) provides significant efficiency gains.
* **Order:** Requires the list to be sorted in order to function correctly.
* **Use Case:** Ideal for scenarios where the list is large and sorted or can be sorted. It is also useful when multiple searches need to be performed on the same list, justifying the initial sorting overhead.

**Conclusion**

Linear search is simple and effective for small or unsorted lists, while binary search is highly efficient for large, sorted lists.