**Exercise 6: Library Management System**

**Linear Search**

**Concept:**  
Linear search is a straightforward search algorithm that checks each element in a list one by one until it finds the target or reaches the end of the list. It is applicable for both sorted and unsorted data.

**Steps:**

1. Start at the beginning of the list.
2. Compare each element’s title with the target title.
3. If they match, return the book.
4. If not, move to the next element.
5. Continue until the end of the list.
6. If no match is found, return "Book not found".

**Time Complexity:** O(n)  
Where n is the number of books in the list. In the worst case, all elements may need to be checked.

**Example:**  
Suppose we have:

Books: ["Data Structures", "Operating Systems", "Algorithms", "Networking"]

Search for: "Algorithms"

Linear search checks:

* "Data Structures" → no match
* "Operating Systems" → no match
* "Algorithms" → match found at index 2.

### ****Binary Search****

**Concept:**  
Binary search is a more efficient algorithm used on **sorted** lists. It works by comparing the middle element to the target and then narrowing the search to the appropriate half.

**Steps:**

1. Ensure the list is sorted by title.
2. Identify the mid element: (low + high) / 2.
3. Compare target title with the mid-title.
   * If they match, return the book.
   * If target < mid, search the left half.
   * If target > mid, search the right half.
4. Repeat until the book is found or the range is empty.

**Time Complexity:** O(log n)  
At each step, the list is halved, which results in much fewer comparisons than linear search.

**Example:**  
Sorted List: ["Algorithms", "Data Structures", "Networking", "Operating Systems"]  
Search for: "Networking"

1. Mid = "Data Structures" → "Networking" > mid → search right
2. Mid = "Networking" → match found

### ****Time Complexity Comparison****

| **Search Type** | **Best Case** | **Average Case** | **Worst Case** |
| --- | --- | --- | --- |
| Linear Search | O(1) | O(n) | O(n) |
| Binary Search | O(1) | O(log n) | O(log n) |

### ****When to Use Each Algorithm****

#### **Linear Search**

**Use Case:**

* When data is **unsorted**.
* When working with **small datasets**.
* When **searches are infrequent**.

**Pros:**

* Simple and easy to implement.
* Works on any type of list—sorted or unsorted.

**Cons:**

* Inefficient for large datasets.
* Search time grows linearly as the list grows.

#### **Binary Search**

**Use Case:**

* When the data is **already sorted**.
* When working with **large datasets**.
* When **frequent searches** are performed.

**Pros:**

* Much faster for large datasets.
* Minimizes the number of comparisons significantly.

**Cons:**

* Only works on sorted data.
* Slightly more complex to implement.