Exercise6

1. Understanding Search Algorithms

Linear Search:

- **Algorithm:** Traverse through each element in the list until the target element is found or the end of the list is reached.
- **Time Complexity:** O(n) where `n` is the number of elements in the list.
- **When to Use:** Suitable for small or unsorted datasets where the overhead of sorting is not justified.

Binary Search:

- **Algorithm:** Works on a sorted list by repeatedly dividing the search interval in half. If the target element is less than the middle element, the search continues in the lower half; otherwise, it continues in the upper half.
- **Time Complexity:** O(log n) where `n` is the number of elements in the list.
- **When to Use:** Suitable for large datasets where the list is already sorted or can be sorted.

4. Analysis

Time Complexity Comparison:

- **Linear Search:**
  - **Best Case:** O(1) if the book is the first in the list.
  - **Worst Case:** O(n) where `n` is the number of books, as it may require checking each book.
- **Binary Search:**
  - **Best Case:** O(1) if the book is located at the middle of the sorted array.
  - **Worst Case:** O(log n) where `n` is the number of books, due to the halving of the search space each step.

When to Use Each Algorithm:

- **Linear Search:** Useful when the dataset is small or unsorted, where the overhead of sorting is not justified. It is also simple to implement and does not require the list to be sorted.
- **Binary Search:** Ideal for large datasets where the list is sorted. It provides much faster search times compared to linear search due to its logarithmic time complexity.