**LIBRARY MANAGEMENT SYSTEM**

**Linear Search**

* **Description**: A search algorithm that sequentially checks each element of a list until the desired element is found or the list is exhausted.
* **Time Complexity**:
  + Best case: O(1) (element found at the first position)
  + Average case: O(n)
  + Worst case: O(n) (element not found or at the last position)
* **Advantages**:
  + Simple to implement
  + Does not require the list to be sorted

**Binary Search**

* **Description**: A search algorithm that finds the position of a target value within a sorted array by repeatedly dividing the search interval in half.
* **Time Complexity**:
  + Best case: O(1) (element found at the middle position)
  + Average case: O(log n)
  + Worst case: O(log n) (element not found)
* **Advantages**:
  + Much faster than linear search for large datasets
  + Requires the list to be sorted

**Analysis**

**Time Complexity Comparison**

* **Linear Search**:
  + Best case: O(1) (element found at the first position)
  + Average case: O(n)
  + Worst case: O(n) (element not found or at the last position)
* **Binary Search**:
  + Best case: O(1) (element found at the middle position)
  + Average case: O(log n)
  + Worst case: O(log n) (element not found)

**When to Use Each Algorithm**

* **Linear Search**:
  + Suitable for small datasets where the overhead of sorting is not justified.
  + Useful when the dataset is unsorted or dynamically changing frequently.
* **Binary Search**:
  + Preferred for large datasets due to its logarithmic time complexity.
  + Requires the dataset to be sorted, so it may involve an initial sorting step (O(n log n)).
  + More efficient for datasets where search operations are more frequent than insertions or deletions.