Analysis

1. Compare the time complexity of linear and binary search algorithms.

Ans.

**Linear Search:** Linear search scans through each element in a list one by one until it finds the target value or reaches the end of the list.

Time Complexity:

Best Case: 𝑂(1) — This occurs if the target value is the first element in the list.

Average Case: 𝑂(𝑛) — On average, the search will need to examine half of the elements in the list.

Worst Case: 𝑂(𝑛) — This occurs if the target value is at the end of the list or not present at all, requiring the search to check every element.

**Binary Search:** Binary search works on a sorted list by repeatedly dividing the search interval in half. It compares the target value with the middle element of the interval and eliminates half of the search space based on the comparison.

Time Complexity:

Best Case: 𝑂(1) — This occurs if the target value is the middle element of the list.

Average Case: 𝑂(log𝑛) — On average, binary search will require log𝑛 comparisons to find the target value or determine that it is not in the list.

Worst Case: 𝑂(log𝑛) — This occurs when the target value is either present in the list or not, requiring log𝑛 comparisons.

Explanation: Binary search has a time complexity of 𝑂(log𝑛) because it repeatedly halves the search space, making the number of comparisons proportional to the logarithm of the number of elements in the list.

1. Discuss which algorithm is more suitable for your platform and why.

Ans. In case of our program we are inserting all the data manually, so they are unsorted by default. If we sort it the minimum time complexity would be O(nlogn) and then sorting it using binary search would be O(logn) which is not feasible. In such a case when we have unsorted array by default, it is better to use linear search. So in this problem linear search is preferrable as it takes less time.

In problems where the list would be sorted initially, binary search is preferrable.