**1.Understand Search Algorithms:**

***Q1.Explain linear search and binary search algorithms***.

Linear Search:

Description: It is a basic simple search algorithm that checks each element of the list linearly until the required element has been found or the end of the list is reached.

Time Complexity:

Best Case: O(1) when the item to be searched is the first one.

Average Case: O(n)

Worst Case: O(n) when the item to be searched is the last one or not found.

Binary Search:

Description: Binary Search is a very efficient search algorithm that requires the array to be sorted. It involves repeatedly dividing the search interval in half and comparing the target value to the middle element. In case of nonequivalence, it halves the interval and repeats the process in the appropriate half.

Time Complexity:

Best Case: O(1), when the middle element is the target.

Average Case: O(log n)

Worst Case: O(log n)

2.**Analysis:**

***Q1.Compare the time complexity of linear and binary search.***

Linear Search:

Best Case: O(1) when it is the first element

Average Case: O(n)

Worst Case: O(n) when target element is the last or not present

Binary Search:

Best Case: O(1) when middle element is the target

Average Case: O(log n)

Worst Case: O(log n)

***Q2.Discuss when to use each algorithm based on the data set size and order.***

Comparison and Usage:

Linear Search:

For the small-sized array or for unsorted array

It is very simple, no need to sort the array.

Preferable when the cost for sorting the data is more than the benefit of faster search times.

Binary Search:

Preferred for large datasets with sorted data.

Much more effective in great, sorted datasets because of its logarithmic operation time.

Needs a pre-sorted order of data before searching, which would require an extra O(n log n) operation time for the sorting.