**Linear and Binary Search Algorithms:**

**Linear Search** is the simplest searching technique where we go through each element of the array one by one until we find the target value or reach the end. It doesn’t require the data to be sorted and works on any list. For example, if we are looking for a number in an unsorted array, we start from the first element and keep checking each one until we find it.

**Binary Search**, on the other hand, is a much faster algorithm but only works on **sorted data**. It uses a divide-and-conquer approach. We first check the middle element; if it’s not the target, we eliminate half of the array and continue searching in the remaining half. This process continues until the element is found or the search space becomes empty.

**Time Complexity Comparison:**

* **Linear Search** has a time complexity of **O(n)** in the worst and average cases, where *n* is the number of elements. This is because we may have to check every element one by one.
* **Binary Search** is more efficient with a time complexity of **O(log n)**, as it divides the search space in half with each step.

In simple terms, binary search is much faster than linear search for large datasets, but only if the data is already sorted.

**When to Use Which Algorithm:**

Use **linear search** when:

* The data is **unsorted**.
* The dataset is **small**, so the performance difference doesn’t matter much.
* You want a simple and quick implementation.

Use **binary search** when:

* The data is **sorted**.
* The dataset is **large**, and performance is important.
* Fast searching is needed repeatedly.

For example, if you're searching through a short list of student roll numbers entered randomly, linear search is fine. But if you have a large, sorted list like a database of user IDs, binary search will give much better performance.