**1. Understanding Search Algorithm**

**Linear Search**

Linear search, also known as sequential search, is a simple search algorithm that checks each element in a list or array one by one until the target element is found or the end of the list is reached.

**Example:** Suppose you have an array [3, 5, 7, 9, 11] and want to find the value 9.

* Start at the beginning: check 3 (not a match), then 5 (not a match), then 7 (not a match), then 9 (match found).

**Time Complexity:**

* **Best Case:** O(1) - The target is at the first position.
* **Average Case:** O(n) - On average, you might need to check half the elements.
* **Worst Case:** O(n) - The target is not present or is at the last position.

**Binary Search**

Binary search is an efficient search algorithm that works on sorted arrays or lists. It repeatedly divides the search interval in half, checking if the target is in the left or right half of the current interval.

**Example:** Suppose you have a sorted array [3, 5, 7, 9, 11] and want to find the value 9.

* Check the middle element (7). Since 9 > 7, search the right half [9, 11].
* Check the middle element of the right half (9). Match found.

**Time Complexity:**

* **Best Case:** O(1) - The target is at the middle of the array.
* **Average Case:** O(log n) - Each comparison halves the search space.
* **Worst Case:** O(log n) - The search space is halved each time until the target is found or the space is exhausted.

**Comparison of Time Complexity**

* **Linear Search:**
  + **Best Case:** O(1)
  + **Average Case:** O(n)
  + **Worst Case:** O(n)
* **Binary Search:**
  + **Best Case:** O(1)
  + **Average Case:** O(log n)
  + **Worst Case:** O(log n)

**When to Use Each Algorithm**

* **Linear Search:**
  + Use when the list is unsorted or when the data set is small.
  + Use when simplicity is preferred and sorting is not feasible.
* **Binary Search:**
  + Use when the list is sorted and the data set is large.
  + Preferred for efficient searching in large data sets due to its logarithmic time complexity.