**Linear Search:**

* **Description:** Linear search sequentially checks each element of the list until the target element is found or the list ends.
* **Time Complexity:**
  + Best Case: O(1)
  + Average Case: O(n)
  + Worst Case: O(n)
* **Use Case:** Suitable for small or unsorted datasets.

**Binary Search:**

* **Description:** Binary search divides the sorted list into two halves and repeatedly narrows down the search interval by comparing the target value to the middle element.
* **Time Complexity:**
  + Best Case: O(1)
  + Average Case: O(log n)
  + Worst Case: O(log n)
* **Use Case:** Suitable for large and sorted datasets.

**Time Complexity Comparison:**

* **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:**
  + **Unsorted Data:** Linear search is ideal for unsorted datasets since it doesn't require any preconditions about the order of elements.
  + **Small Data Sets:** For small datasets, the difference in performance between linear and binary search is negligible, making linear search a simpler choice.
* **Binary Search:**
  + **Sorted Data:** Binary search requires the dataset to be sorted. It's much more efficient for large datasets due to its logarithmic time complexity.
  + **Large Data Sets:** For large datasets, binary search drastically reduces the number of comparisons needed to find the target element, making it more suitable.