# **Analysis**

## Compare the time complexity of linear and binary search.

The time complexity of linear search and binary search:

Time Complexity Comparison

1. Linear Search:
   * Best Case: O(1)
     + The target is the first element in the list.
   * Average Case: O(n)
     + On average, half of the elements need to be checked.
   * Worst Case: O(n)
     + The target is not in the list, or it is the last element.
2. Binary Search:
   * Best Case: O(1)
     + The target is the middle element of the list.
   * Average Case: O(log n)
     + The list is divided in half with each comparison.
   * Worst Case: O(log n)
     + The search space is reduced to 1 element after log(n) comparisons.

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

The guide on when to use linear search and binary search based on dataset size and order:

When to Use Linear Search

* Unsorted Data: Use linear search when the dataset is unsorted, as it does not require any specific order.
* Small Datasets: Effective for small datasets (e.g., less than 20-30 elements) where the overhead of sorting or managing a sorted list may not be justified.
* Simple Implementation: Ideal for quick implementations or when searching needs to be done infrequently.

When to Use Binary Search

* Sorted Data: Use binary search only when the dataset is sorted. It takes advantage of the sorted order to significantly reduce search time.
* Large Datasets: More efficient for larger datasets (e.g., thousands of elements or more) due to its O(log n) time complexity.
* Frequent Searches: Beneficial in scenarios where searches are performed frequently, as the upfront cost of sorting (if needed) pays off with faster search times later.