**Search Algorithms**

**Linear Search**

* **Description:** Linear search sequentially checks each element of the list until a match is found or the whole list has been searched.
* **Time Complexity:**
  + Best Case: O(1)) (when the element is found at the beginning of the list)
  + Average Case: O(n) (where n is the number of elements in the list)
  + Worst Case: O(n) (when the element is found at the end of the list or not at all)

**Binary Search**

* **Description:** Binary search works on sorted lists by repeatedly dividing the search interval in half. It compares the target value to the middle element of the array; if they are not equal, it halves the array and continues the search.
* **Time Complexity:**
  + Best Case: O(1) (when the middle element is the target value)
  + Average Case: O(logn)
  + Worst Case: O(logn)

**Analysis**

**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(logn)
  + Worst Case: O(logn)

**When to Use Each Algorithm**

* **Linear Search:**
  + Use when the list is unsorted or the list is small.
  + Suitable for small datasets where the overhead of sorting isn't justified**.**
* **Binary Search:**
  + Use when the list is sorted.
  + Suitable for large datasets where the list is already sorted or can be sorted efficiently.