**Step 1: Understand Search Algorithms:**

Explain linear search and binary search algorithms.

### **1. Linear Search (Sequential Search)**

#### **How It Works:**

Linear search goes through each element in a list **one by one** from the start until it either:

* Finds the matching item, or
* Reaches the end without finding it.

There is **no assumption** about the data being sorted.

#### **Use Cases:**

* Ideal for **small datasets**
* When the list is **unsorted**
* When one **don't want to sort** the data

#### **Time Complexity:**

* **Best Case**: O(1) — if the item is the first element
* **Average Case**: O(n/2) → O(n)
* **Worst Case**: O(n) — if the item is at the end or not present

#### **Advantages:**

* Simple to implement
* No sorting needed

#### **Disadvantages:**

* Slow on large lists
* Time grows linearly with input size

### **2. Binary Search**

#### **How It Works:**

Binary search works by **dividing** the sorted list into halves:

* Compare the target with the **middle** element.
* If it matches — return it
* If the target is **less than** middle, search the **left half**
* If it’s **greater**, search the **right half**
* Repeat until the item is found or range becomes empty

This algorithm is **logarithmic** — it cuts the search space in half at every step.

**Use Cases:**

* Works **only** on sorted lists
* Ideal for **large, frequently searched datasets**
* Great for repeated searches on the same data

#### **Time Complexity:**

* **Best Case**: O(1) — if the middle element is the target
* **Average Case**: O(log n)
* **Worst Case**: O(log n)

#### **Advantages:**

* Extremely fast on large sorted data
* Reduces comparisons dramatically

#### **Disadvantages:**

* Requires **sorted** data
* More complex to implement than linear search

**Step 4: Analysis:**

Compare the time complexity of linear and binary search.

| Search Type | Best Case | Average Case | Worst Case | Sorted Required? |
| --- | --- | --- | --- | --- |
| Linear Search | O(1) | O(n) | O(n) | No |
| Binary Search | O(1) | O(log n) | O(log n) | Yes |

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

### When to Use Which:

| Situation | Use Linear Search | Use Binary Search |
| --- | --- | --- |
| List is **unsorted** | Yes | No |
| List is **small** (few items) | Yes | Optional |
| List is **sorted and large** | Too slow | Yes |
| One-time search, no need to sort | Yes | Not worth it |
| Repeated search on stable sorted data | Inefficient | Best choice |

**Linear Search** is simple and works on any list, but it becomes slow as the list grows.

**Binary Search** is far more efficient, but only applicable if the list is already sorted.

In a real-world **library system**, binary search is preferred **if** the book list is sorted (e.g., by title or author).

For  **small or unsorted data**, linear search is often enough.