**Linear Search (Sequential Search)**

**What it is:**  
Linear search is the simplest searching algorithm. You go through each item in the list **one by one** from start to end, comparing it with the target value.

**How it works:**  
Imagine you have a stack of books and you’re looking for a book with a specific title. You start at the top, check the first book, then the second, and keep going until you find the book or reach the end.

**When it's useful:**

* When the list is **unsorted**
* For **small datasets**
* When you **don’t want or can’t sort the data**

**Example:**  
Let’s say you're looking for "Harry Potter" in a list of 5 books:

1. Pride and Prejudice

2. The Alchemist

3. Harry Potter ← Found here (3rd position)

4. Moby Dick

5. 1984

It checks the first two, then finds the book at the 3rd position.

**Time Complexity:**

* Best Case: O(1) → If the book is at the start
* Worst Case: O(n) → If the book is at the end or not found
* Average Case: O(n)

**⚡ Binary Search**

**What it is:**  
Binary search is a much faster method, but it only works on a **sorted list**. It divides the search space in half with each step.

**How it works:**  
Let’s say the books are sorted alphabetically. Instead of checking every book, you check the **middle book** first.

* If the title matches → Done.
* If your title comes before it alphabetically → Search in the left half.
* If it comes after → Search in the right half.

This keeps repeating until the book is found or the search space is empty.

**When it's useful:**

* For **large datasets**
* When the list is **already sorted** or sorting is manageable

**Example:**  
Searching for “Moby Dick” in this sorted list:

1. 1984

2. Moby Dick ← Found here (2nd attempt)

3. Pride and Prejudice

4. The Alchemist

5. To Kill a Mockingbird

First, it checks the middle → "Pride and Prejudice"  
Since “Moby Dick” comes before that alphabetically, it checks the left side next and finds it.

**Time Complexity:**

* Best Case: O(1) → If the target is in the middle
* Worst & Average Case: O(log n)