**Step 1: Understand Asymptotic Notation**

**Big O Notation:**

Big O notation is a mathematical representation used to describe the time complexity of algorithms. It provides an upper bound on the running time, helping to understand how the algorithm's performance scales with input size. Key points include:

* **O(1)**: Constant time – performance is unaffected by the size of the input.
* **O(n)**: Linear time – performance scales directly with the size of the input.
* **O(log n)**: Logarithmic time – performance grows logarithmically with input size.
* **O(n^2)**: Quadratic time – performance scales with the square of the input size.

**Search Operation Scenarios:**

1. **Linear Search:**
   * **Best Case**: O(1) – The target element is at the first position.
   * **Average Case**: O(n) – The target element is somewhere in the middle.
   * **Worst Case**: O(n) – The target element is at the last position or not present.
2. **Binary Search:**
   * **Best Case**: O(1) – The target element is in the middle of the array.
   * **Average Case**: O(log n) – The array is repeatedly divided in half.
   * **Worst Case**: O(log n) – The search space is halved each time.

**Step 4: Analysis**

**Time Complexity Comparison:**

1. **Linear Search:**
   * **Time Complexity**: O(n)
   * **Pros**: Simple to implement, no need for sorted data.
   * **Cons**: Less efficient for large datasets; performance degrades linearly with the size of the dataset.
2. **Binary Search:**
   * **Time Complexity**: O(log n) after sorting
   * **Pros**: Very efficient for large datasets; performance improves significantly with larger datasets.
   * **Cons**: Requires sorted data; sorting introduces an additional time complexity of O(n log n) if sorting is needed.

**Which Algorithm is More Suitable?**

* **For Smaller Datasets**: Linear search might be sufficient due to its simplicity.
* **For Larger Datasets**: Binary search is more suitable due to its logarithmic time complexity. However, ensure the dataset is sorted, which might require additional overhead for sorting.