**Understand Asymptotic Notation**

**Big O Notation:**

* **Definition:** Big O notation describes the upper bound of an algorithm's time complexity, giving an estimate of the worst-case scenario in terms of the size of the input data. It provides a high-level understanding of the algorithm's performance and helps compare the efficiency of different algorithms.

**Best, Average, and Worst-Case Scenarios:**

* **Best Case:** The minimum time an algorithm takes to complete, usually when the input is optimally sorted or the target element is at the beginning of the data structure.
* **Average Case:** The expected time an algorithm takes to complete, averaged over all possible inputs of a given size.
* **Worst Case:** The maximum time an algorithm takes to complete, usually when the input is in the worst possible configuration for the algorithm.

**Analysis**

**Time Complexity Analysis:**

1. **Linear Search:**
   * **Best Case:** O(1) (The element is found at the beginning of the array)
   * **Average Case:** O(n) (The element is found somewhere in the middle of the array)
   * **Worst Case:** O(n) (The element is found at the end of the array or not found at all)
2. **Binary Search:**
   * **Best Case:** O(1) (The element is found at the middle of the array)
   * **Average Case:** O(log n) (The element is found somewhere in the array)
   * **Worst Case:** O(log n) (The element is not found in the array)

**Which Algorithm is More Suitable?**

* **Linear Search:** Suitable for small datasets or unsorted arrays. It does not require sorting but is inefficient for large datasets due to its O(n) time complexity.
* **Binary Search:** Suitable for large datasets and requires the array to be sorted. It is much more efficient with O(log n) time complexity.

For an e-commerce platform with a large inventory, **binary search** is more suitable due to its significantly better performance in terms of time complexity when searching through a sorted dataset.