**EXERCISE 2: E-commerce Platform Search Function**

**Understanding Asymptotic Notation**

**Big O Notation:**

Big O notation is a mathematical notation used to describe the upper bound of an algorithm's time or space complexity, providing an estimate of the worst-case scenario in terms of performance as the input size grows. It helps in comparing the efficiency of different algorithms and understanding how they scale with increasing input sizes.

**Best, Average, and Worst-Case Scenarios for Search Operations:**

* **Best Case:**
  + The scenario where the search operation takes the least amount of time. For linear search, this happens when the element is at the first position (O(1)).
  + For binary search, the best case is also when the element is found in the middle of the array on the first comparison (O(1)).
* **Average Case:**
  + The expected time complexity for the search operation given a uniform distribution of possible positions of the target element.
  + For linear search, the average case is O(n/2), which simplifies to O(n).
  + For binary search, the average case is O(log n), as the search space is halved with each step.
* **Worst Case:**
  + The scenario where the search operation takes the most amount of time.
  + For linear search, the worst case is when the element is at the last position or not present at all (O(n)).
  + For binary search, the worst case is when the element is not present, and all levels of the search tree are traversed (O(log n)).

**Analysis**

**Time Complexity Comparison:**

* **Linear Search:**
  + **Best Case:** O(1) (when the target is the first element)
  + **Average Case:** O(n)
  + **Worst Case:** O(n)
* **Binary Search:**
  + **Best Case:** O(1) (when the target is the middle element)
  + **Average Case:** O(log n)
  + **Worst Case:** O(log n)

**Which Algorithm is More Suitable and Why:**

* **Linear Search:**
  + **Pros:** Simple to implement, no need for a sorted array.
  + **Cons:** Inefficient for large datasets, O(n) time complexity in average and worst cases.
  + **Suitable for:** Small datasets where simplicity is preferred over efficiency.
* **Binary Search:**
  + **Pros:** Much more efficient for large datasets, O(log n) time complexity.
  + **Cons:** Requires a sorted array, additional overhead for sorting if the array is not already sorted.
  + **Suitable for:** Large datasets where search efficiency is critical, and the data can be maintained in sorted order.