**E-commerce Platform Search Function**

**Big O Notation**

Big O notation is used to describe the performance characteristics of an algorithm, specifically its time and space complexity. It provides an upper bound on the growth rate of an algorithm’s runtime or memory usage as the input size increases. This helps in analyzing and comparing algorithms, especially for large inputs.

* **O(1)**: Constant time – the runtime does not change with the size of the input.
* **O(log n)**: Logarithmic time – the runtime increases logarithmically as the input size increases.
* **O(n)**: Linear time – the runtime increases linearly with the size of the input.
* **O(n log n)**: Linearithmic time – the runtime increases proportionally to nlog⁡nn \log nnlogn.
* **O(n^2)**: Quadratic time – the runtime increases quadratically with the size of the input.

**Search Operation Scenarios**

1. **Linear Search**:
   * **Best Case**: O(1) – The element is found at the first position.
   * **Average Case**: O(n) – The element could be anywhere in the array.
   * **Worst Case**: O(n) – The element is at the last position or not present.
2. **Binary Search**:
   * **Best Case**: O(1) – The middle element is the target.
   * **Average Case**: O(log n) – The array is divided in half each iteration.
   * **Worst Case**: O(log n) – The element is not found after logarithmic iterations.

**Analysis**

**Time Complexity Comparison**

1. **Linear Search**:
   * **Time Complexity**: O(n) – Each element is checked sequentially.
2. **Binary Search**:
   * **Time Complexity**: O(log n) – Efficient for large datasets when the array is sorted.

**Suitability for the Platform**

* **Linear Search**: Simple to implement and does not require a sorted array. Suitable for small datasets or unsorted data.
* **Binary Search**: Much faster for large datasets but requires the data to be sorted. Sorting the data adds an additional O(n log n) complexity.