**Explanation:**

When analyzing an algorithm, we often consider three scenarios:

1. Best Case (Ω - Omega):

* The most favorable situation.
* For Linear Search, it’s when the element is found at the first index → O(1).

2. Average Case (Θ - Theta):

* The expected performance over all possible inputs.
* For Linear Search, it's when the item is somewhere in the middle → O(n/2) ≈ O(n).

3. Worst Case (O - Big O):

* The most unfavorable scenario.
* For Linear Search, the element is not present or is the last item → O(n).
* For Binary Search, it's O(log n) in all scenarios, but worst case means maximum depth is reached.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** |
| Linear Search | O(1) | O(n) | O(n) |
| Binary Search | O(1) | O(log n) | O(log n) |

**Analysis:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** | **Requires Sorted Data** |
| **Linear Search** | O(1) – first match | O(n/2) → O(n) | O(n) – last/no match | No |
| **Binary Search** | O(1) – middle match | O(log n) | O(log n) | Yes |

**Which Algorithm is More Suitable for an E-commerce Platform and Why?**

Binary Search is generally more suitable for large-scale e-commerce platforms if the product list is sorted.

Why Binary Search?

* Performance: It’s significantly faster with large datasets (O(log n) vs O(n)).
* User Experience: Faster response improves search responsiveness and customer satisfaction.
* Scalability: It performs consistently well even as the number of products grows.

When to use Linear Search?

* For small datasets or temporary in-memory searches.
* When products aren't sorted and sorting is too expensive or unnecessary.
* For flexible or fuzzy matching, where exact sorting isn’t feasible.