**Understand Asymptotic Notation:**

**Big O Notation**

Big O notation describes how an algorithm's runtime or space requirements grow as the input size grows. It gives us an upper bound on complexity.

* O(1): Constant time (independent of input size)
* O(log n): Logarithmic time (very efficient)
* O(n): Linear time (proportional to input size)
* O(n²): Quadratic time (grows rapidly with input size)

1. **Best Case**: Minimum possible running time
   * Linear search: O(1) (item is first element)
   * Binary search: O(1) (item is middle element)
2. **Average Case**: Expected running time for random inputs
   * Linear search: O(n)
   * Binary search: O(log n)
3. **Worst Case**: Maximum possible running time
   * Linear search: O(n) (item is last or not present)
   * Binary search: O(log n)

**Analysis:**

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

Performance Comparison(Output):

* Linear search time: 1400 ns
* Binary search time: 6500 ns

**Which Algorithm is More Suitable?**

For an e-commerce platform:

**Binary Search is better when:**

* Products can be maintained in sorted order (by ID)
* The product catalog is large (thousands+ items)
* Search by ID is the primary operation

**Linear Search might be better when:**

* The product catalog is very small
* Products are frequently added/removed (maintaining sorted order is costly)