**Understand Asymptotic Notation:**

1.Explain Big O notation and how it helps in analyzing algorithms.

Ans : Big O notation describes the **upper bound** of an algorithm's runtime or space requirements in terms of input size n. It's a way to measure **how well an algorithm scales**.

| **Complexity** | **Description** |
| --- | --- |
| O(1) | Constant time |
| O(log n) | Logarithmic time |
| O(n) | Linear time |
| O(n log n) | Log-linear time |
| O(n²) | Quadratic time |

2. Describe the best, average, and worst-case scenarios for search operations.

Ans:

| **Type** | **Best Case** | **Average Case** | **Worst Case** |
| --- | --- | --- | --- |
| **Linear Search** | O(1) | O(n/2) ≈ O(n) | O(n) |
| **Binary Search** | O(1) | O(log n) | O(log n) |

* **Best case:** Item is found at the first attempt.
* **Average case:** Item is somewhere in the middle.
* **Worst case:** Item is not in the list (full scan for linear, max depth for binary).

**Analysis**

3. Compare the time complexity of linear and binary search algorithms.

Ans:

| **Algorithm** | **Time Complexity** | **Suitable For** |
| --- | --- | --- |
| Linear Search | O(n) | Small datasets or unsorted data |
| Binary Search | O(log n) | Large datasets **with sorted data** |

4. Discuss which algorithm is more suitable for your platform and why.

Ans: Use **linear search** for:

* Small number of products.
* Unsorted data (no sorting overhead).

Use **binary search** for:

* Large product lists.
* Optimized performance with sorted product names.