**Understanding Asymptotic Notation**

**Big O notation** describes the **upper bound** of an algorithm's **running time or space requirement** as the input size grows. It gives a high-level understanding of the algorithm's **efficiency.**

O (1): Constant- Indicates that the operation doesn’t depend on the input size.

O (log n): Logarithmic- Indicates that the operation is proportional to the logarithm of input size.

O (n): Linear- Indicates that the operation is proportional to the input size.

O (n log n): Linear Logarithmic- Indicates that the operation is proportional to input size times the logarithm of input size.

O (n2): Quadratic- Indicates that the operation is proportional to square of the input size.

Other: O(2n), O(nn), O(nx), O(n!)

For search operations,

Best case occurs when the element is found in first try.

Time complexity :- Linear Search: O(1) & Binary Search: O(1)

Average case occurs when the element is somewhere in the middle.

Time complexity :- Linear Search: O(n/2) ≈ O(n) & Binary Search: O(log n)

Worst case occurs when the element is not found or is the last element.

Time complexity :- Linear Search: O(n) & Binary Search: O(log n)

**Which Algorithm is better?**

We use Linear Search for Small and Unsorted datasets, Partial/prefix matching (.contains()) whereas Binary Search is used for large, sorted datasets for exact match lookups. ***Binary search is preferred for fast exact lookups for an ecommerce platform.***