**Exercise 2: E-commerce Platform Search Function**

1. **UNDERSTANDING ASYMPTOTIC NOTATIONS**
2. Explain Big O notation and how it helps in analyzing algorithms?

Answer:

**Introduction**

Asymptotic notation is a mathematical tool that evaluates the performance of an algorithm in terms of input size (**order of growth** of time taken or space occupied by an algorithm in terms of input size). One of the notations is Big O.

**Big O Definition**

Given two functions f(n) and g(n), we say that f(n) is O(g(n)) if there exist constants c > 0 and n0 >= 0 such that

f(n) <= c\*g(n) for all n >= n0.

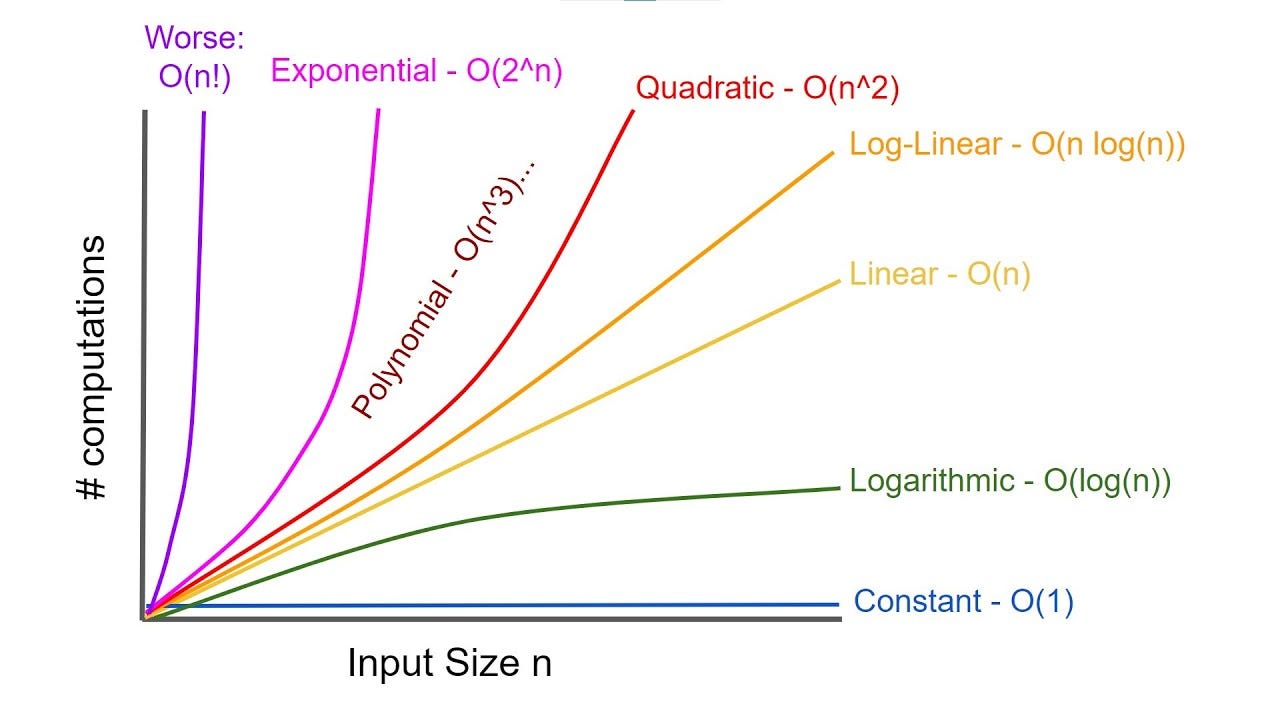
Big-O is a way to express the **upper bound**of an algorithm’s time or space complexity.

**Big O Complexities**

O(1) → O(log n) → O(n) → O(n log n) → O(n²) → O(n³) → O(2^n) → O(n!) – Increasing order

**Usefulness in Analyzing Algos**

* Helps to choose algo and data structures that allows most efficient use of of CPU cycles and memory requirements.
* Allows programmers to compare different algorithms and choose the most efficient one for a specific problem.
* It helps us understand how well an algorithm scales and anticipates its behavior as input size increases.
* It allows developers to optimise their code, enhance efficiency, and improve overall performance.



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

Answer:

1. **Linear Search –** No sorting required.

Best Case: O(1), Target is at index 0.

Average Case: O(n/2) →O(n), Target is at n/2 (in the middle)

Worst Case: O(n), Target is at last index or is not present.

1. **Binary Search –** Sorting is mandatory.

Best Case: O(1), Target is at the middle (first comparison for middle element)

Average Case: O(log n), Target is found after several halving steps.

Worst Case: O(log n), Target is not present or found at the last comparison.

1. **ANALYSIS**
2. Compare the time complexity of linear and binary search algorithms.

Answer:

**Linear search** checks each element one-by-one, time grows proportionally with the number of elements.

Best Case: O(1), Target is at index 0.

Average Case: O(n/2) →O(n), Target is at n/2 (in the middle)

Worst Case: O(n), Target is at last index or is not present.

**Binary search** divides the search space in half each time ,hence time grows logarithmically.

Best Case: O(1), Target is at the middle (first comparison for middle element)

Average Case: O(log n), Target is found after several halving steps.

Worst Case: O(log n), Target is not present or found at the last comparison.

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

Answer:

For search operation functionality in an e-commerce platform, **Binary Search** overpowersand performs better than **Linear Search.**

Justification:

1. In e-commerce platforms, where there are products in huge numbers and of diverse categories, search operation by category or product name or product id becomes optimal using binary search (O(log n)) over linear search (O(n)).
2. The operation of search is frequent, hence overhead of O(n) is not performance-friendly.

Overhead of Sorting in Binary Search:

Time complexity of O(n logn) is the most optimal for Sorting operation in Average Case.

Hence linear search is preferred when there are less items to search upon.

But, the product ids are sorted by default in most systems, hence this avoids the overhead of sorting and Binary Search is easily applied.