**EXERCISE 2: E-COMMERCE PLATFORM SEARCH FUNCTION**

**Explain Big O notation and how it helps in analysing algorithms. Describe the best, average, and worst-case scenarios for search operations.**

Big-O, commonly referred to as Order of, is used to express the upper bound of an algorithm’s time complexity. It analyses the worst case of an algorithm.

Big O notation only describes the asymptotic behavior of a function, not its exact value.

Denoted as **O(f(n)),** where f(n) is a function that represents the number of operations (steps) that an algorithm performs to solve a problem of size n.

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.

**How Big O notation is used,**

* analyze the efficiency of algorithms.
* describe how the runtime or space requirements of an algorithm grow as the input size increases.
* compare different algorithms and choose the most efficient one for a specific problem.
* to understand the scalability of algorithms
* predicting how they will perform as the input size grows.
* to optimize code and improve overall performance.

The various complexities,

* Constant time – O(1)
* Linear – O(n)
* Logarithmic – O(log n)
* Super Linear – O(n log n)
* Polynomial – O(n^2)
* Exponential – O(2^n)

**For search operation,**

* **Best Case:** The scenario where the search operation completes in the minimum possible time. Example would be the search element being the first element of the array.
* **Average Case:** The scenario that represents the average time taken by the search operation over all possible inputs. Example would be the search element being somewhere as the middle element of the array.
* **Worst Case:** The scenario where the search operation takes the maximum possible time. Example would be the search element being the last element of the array.

**Compare the time complexity of linear and binary search algorithms. Discuss which algorithm is more suitable for your platform and why.**

The time complexity for linear search and binary search are,

**Linear Search:**

Suitable for small datasets or when the data is unsorted. Simple to implement but not efficient for large datasets.

* **Best Case: O(1)** – The product is found at the first position.
* **Average Case: O(n)** – The product is somewhere in the middle.
* **Worst Case: O(n)** – The product is at the last position or not found.

**Binary Search (sorted array):**

Suitable for large datasets where the data is sorted. More efficient than linear search for large datasets due to logarithmic time complexity. Requires additional overhead for maintaining sorted data.

* **Best Case: O(1)** – The product is at the middle.
* **Average Case: O(log n)** – The search space is halved each time.
* **Worst Case: O(log n)** – The product is at the beginning or the end of the search space.

For an e-commerce platform with potentially large datasets, binary search is more suitable due to its efficient O(log n) time complexity. However, it requires the product data to be sorted, which might add an overhead for maintaining the sorted order