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

Solution by: Mahika Thakur, CSE 7TH Semester, MCKV Institute of Engineering

**Q. i) Explain Explain Big O notation and how it helps in analyzing algorithms.**

Big O notation is a mathematical notation used to describe the upper bound of an algorithm's time or space complexity. It provides an abstract measurement of the efficiency of an algorithm by categorizing how the runtime or space requirements grow as the input size increases. It is essential for comparing algorithms and understanding their performance, especially for large inputs.

O(1): Constant time - The operation's time complexity is independent of the input size.

O(n): Linear time - The operation's time complexity grows linearly with the input size.

O(log n): Logarithmic time - The operation's time complexity grows logarithmically with the input size.

O(n log n): Linearithmic time - The operation's time complexity grows linearly with a logarithmic factor.

O(n^2): Quadratic time - The operation's time complexity grows quadratically with the input size.

**ii) Describe the best, average, and worst-case scenarios for search operations.**

Best Case: The scenario where the algorithm performs the minimum number of operations. For example, in a search operation, the best case occurs when the target element is found at the beginning of the array.

Average Case: The scenario that represents the expected behavior of the algorithm for typical inputs. It considers all possible inputs and their probabilities.

Worst Case: The scenario where the algorithm performs the maximum number of operations. For example, in a search operation, the worst case occurs when the target element is not found or is found at the end of the array.

**Q. i) Compare the time complexity of linear and binary search algorithms.**

Time Complexity Comparison

Linear Search:

Best Case: O(1) - The target element is at the beginning of the array.

Average Case: O(n) - The target element is somewhere in the middle of the array.

Worst Case: O(n) - The target element is at the end of the array or not present.

Binary Search:

Best Case: O(1) - The target element is at the middle of the array.

Average Case: O(log n) - The array is divided in half repeatedly.

Worst Case: O(log n) - The target element is at one of the ends or not present.

**ii)Discuss which algorithm is more suitable for your platform and why**

Binary search is generally more efficient than linear search for large datasets because of its O(log n) time complexity compared to O(n) for linear search. However, binary search requires the array to be sorted, which may incur additional overhead if the array is frequently updated.

We should use binary search if the product list can be maintained in a sorted order and the search operations are frequent. For small or unsorted datasets, or if the product list is frequently updated, linear search might be simpler and more practical.