**MONICA B (SUPERSET ID - 5008627)**

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

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

Big O notation explains how an algorithm's efficiency improves as the amount of data it analyzes rises. It focuses on the most important element impacting an algorithm's runtime, neglecting constant factors and lower-order terms. This makes it an effective tool for comparing and analyzing algorithms without becoming mired down in implementation specifics.

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

**Best Case:** If the element is in the first place, linear search takes (O(1)). If the element is at the midway of the sorted array, the time complexity for binary search is O(1).

**Average Case:** Linear search takes (O(n)) time because it may need to check half of the elements. The temporal complexity of binary search is O(\log n), assuming a balanced search.

**Worst Case:** Linear search may take (O(n)) if the element is at the end or not present. Binary search remains (O(\log n)), as it can traverse the entire height of the search tree.

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

1.**Linear Search**: O(n) in both average and worst cases

O(1) In the best case.

2.**Binary Search**: O(log⁡n) in both average and worst cases

O(1) In the best case.

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

For an e-commerce platform, binary search is more suitable due to its (O(log n)) time complexity, which greatly accelerates search operations compared to linear search's (O(n)). Although binary search requires a sorted array, its ability to handle big datasets makes it perfect for quick and responsive search capabilities. Linear search, while simpler, is less efficient for big inventories and preferable for smaller or unsorted data.