**COGNIZANT DIGITAL NURTURE – 3.0**

**JAVA FSE**

**WEEK – 1 EXERCISES**

**DATA STRUCTURES AND ALGORITHMS**

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

**Step 1: Understand Asymptotic Notation:**

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

An algorithm's complexity or performance can be expressed using Big O notation. It concentrates on the worst-case situation as the input size increases. It aids in the analysis of how an algorithm's runtime or space needs increase with input size.

* O (1): Constant time: Regardless of the size of the input, the operation takes the same amount of time.
* O(n): Linear time, in which the amount of the input determines how long the operation takes.
* O (log n): Logarithmic time: As the size of the input increases, the operation's duration increases logarithmically.
* O (n log n): Linearithmic time: The time it takes to complete an operation increases in a way that combines logarithmic and linear growth.
* O(n^2): Quadratic time: As the size of the input increases, the operation's time increases quadratically.

**Best, Average, and Worst-Case Scenarios**

* Best Case: The situation in which the algorithm runs in the fewest possible steps. This would be locating the target element on the first attempt for search operations.
* Average Case: Assuming a random distribution of inputs, this is the anticipated situation. This would be the typical quantity of search operations needed to locate an element.
* Worst Case: The situation in which the algorithm executes as many actions as possible. This would mean, in terms of search operations, that the element would either not be found at all or would only be found at the very end.

**Step 2: Setup**

***Refer Program Files***

**Step 3: Implementation**

***Refer Program Files***

**Step 4: Analysis:**

**I.** **Compare the time complexity of linear and binary search algorithms.**

**Time Complexity:**

**Linear Search:**

* **Best Case: O (1)** – When the target product is the first element.
* **Average Case: O(n**) – On average, it will take n/2 comparisons.
* **Worst Case: O(n)** – When the target product is the last element or not present at all.

**Binary Search**:

* **Best Case**: O (1) – When the target product is the middle element.
* **Average Case**: O (log n) – Because the array is divided in half each time.
* **Worst Case**: O (log n) – Even in the worst case, the array is still divided in half each time.

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

The platform is better suited for binary search if the following criteria are satisfied:

* **Data Sorting:** Sorting the data is necessary for binary search. If the data is changed often, sorting could need extra overhead.
* **Regular Searches**: The effectiveness of binary search will offset the sorting overhead if search operations occur more frequently than update operations.
* **Big Dataset:** Compared to linear search's linear complexity, binary search's logarithmic complexity offers appreciable speed gains for huge datasets.