**Data structures and Algorithms**

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

**Solutions :**

**Step 1 :** **Understand Asymptotic Notation:**

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

**Big O notation :**

Big O Notation is a mathematical way to describe the efficiency of an algorithm in terms of:

* Time Complexity : how long an algorithm takes to run
* Space Complexity : how much memory an algorithm uses

Big O describes how the runtime or space requirement grows relative to the size of the input (commonly denoted as n), especially as n becomes very large.

**Helps in analyzing algorithms :**

* Helps understand how fast or slow an algorithm will be for large datasets.
* Allows comparison between multiple algorithms to pick the one with the best performance.
* Helps determine which algorithm will perform better in real-world, large-scale applications.

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

When we use a search algorithm to find something in a list, the time it takes can change depending on where the item is. So, we look at three cases: best case, average case, and worst case.

* The **best case** is the fastest situation. This happens when the item we are looking for is at the beginning of the list. For example, in linear search, if the item is in the first position, we find it right away. This is very quick and takes only one step.
* The **average case** is what usually happens in normal situations. It means the item could be anywhere in the list — maybe in the middle or randomly placed. So, the search will take some time but not too long. For example, in linear search, we might have to check half of the list.
* The **worst case** is the slowest situation. This happens when the item is at the end of the list or not in the list at all. In this case, we have to check every item one by one, which takes the most time.

From all the three cases, we can understand how fast or slow a search algorithm can be. This helps us choose the best method for searching, especially when the list is big.

**Step 2 : Setup (in code folder)**

**Step 3: Implementation (in code folder)**

**Step 4 : Analysis**

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

**Linear Search :**

* **Definition**: Searches each element in the list one by one.
* **Best Case**:  
  🔸 O(1) – If the target element is at the beginning.
* **Average Case**:  
  🔸 O(n) – On average, it checks half the elements.
* **Worst Case**:  
  🔸 O(n) – If the target is at the end or not found.
* **When to Use**:  
  🔹 When the list is **unsorted**.

**Binary Search :**

* **Definition**: Repeatedly divides the sorted list in half to find the element.
* **Best Case**:  
  🔸 O(1) – If the target is at the middle.
* **Average Case**:  
  🔸 O(log n) – Cuts the list size in half each time.
* **Worst Case**:  
  🔸 O(log n) – Even if the element is not present, it takes log steps.
* **When to Use**:  
  🔹 When the list is **sorted**.

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

| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** | **Time Complexity** |
| --- | --- | --- | --- | --- |
| **Linear Search** | O(1) | O(n) | O(n) | Slower for large datasets |
| **Binary Search** | O(1) | O(log n) | O(log n) | Much faster, but needs sorted data |
|  |  |  |  |  |

* For small or unsorted data, linear search is simple and works fine.
* For large datasets (common in e-commerce), binary search is better due to logarithmic performance, but the array must be sorted.
* Thus, binary search is more suitable for an e-commerce platform with a pre-sorted product list or when search speed is critical.