**Exercise 3: E-commerce Platform Search Function**

1. **Explain Big O notation and how it helps in analysing algorithms.**  
   Big O notation: It describes how an algorithm's performance scales as the size of the input grows.  
   It gives an upper bound on the number of operations an algorithm will take, helping us understand the worst-case scenario in terms of:  
   Time complexity: how the execution time increases with input size.   
   Space complexity: how memory consumption increases with input size.  
   Common Big O notations:

|  |  |  |
| --- | --- | --- |
| Notation | Name | Use Case |
| O(1) | Constant Time | Accessing an array element |
| O(log n) | Logarithmic Time | Binary Search |
| O(n) | Linear Time | Traversing an array |
| O(n log n) | Linearithmic Time | Merge Sort, Quick Sort (avg) |
| O(n²) | Quadratic Time | Bubble Sort |
| O(2ⁿ) | Exponential Time | Recursive Fibonacci |
| O(n!) | Factorial Time | Brute force |

Big O helps in:

1. Predict Performance for Large Inputs

Even if an algorithm is fast on small inputs, Big O reveals if it'll scale well.

2. Compare Algorithms Objectively

Let’s say you’re choosing between:

Linear Search: O(n)

Binary Search: O(log n)

Even if linear search seems faster at small sizes, binary search will outperform it significantly for large datasets.

3. Avoid Bottlenecks

Knowing the time complexity helps identify where to optimize. For example:

Nested loops → O(n²) → candidate for improvement

Repeated recursive calls → O(2ⁿ) → likely inefficient

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

When analysing the efficiency of search algorithms, we examine their behaviour in three different input situations:  
Best Case: The scenario where the algorithm performs the least amount of work.  
Average Case: The scenario reflecting a typical or expected input.  
Worst Case: The scenario requiring the most operations  
Example in Linear Search :

Target element is first in the list → O(1) – this is the Best Case Scenario

Target is somewhere in the middle → O(n) —this is the Average Case Scenario

Target is last or not present → O(n) –this is the Worst-Case scenario.  
 Knowing the best, average, and worst cases helps in:  
 System Design: Plan for worst-case performance (avoid crashes)  
 Algorithm Comparison: Decide which search algorithm suits your needs  
 Optimization & Scaling: Handle large data with predictable performance

|  |
| --- |
|  |