**Data Structure and Algorithms:**

Exercise 2: E-commerce Platform Search Function

Understand Asymptotic Notation:

Big O Notation:

* Big O notation tells the upper bound of an algorithm's running time as a function of input size n.
* It helps developers analyse howalgorithms scale and choose the most efficient one for a given problem.

Best, Average, and Worst Case (For searching):

* **Best Case**: The ideal scenario where the item is found early.
* **Average Case**: The typical case considering random input.
* **Worst Case**: The most time-consuming scenario. For example, the item is last or not present.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Best Case | Average Case | Worst Case |
| Linear Search | O(1) | O(n) | O(n) |
| Binary Search | O(1) | O(logn) | O(logn) |

Analysis:

Time Complexity Comparison:

|  |  |  |
| --- | --- | --- |
| Algorithm | Time Complexity | What does the algo do |
| Linear Search | O(n) | Scans each element one by one |
| Binary Search | O(logn) | Divides the search space in half each step. |

Which Is More Suitable:

* Binary Search is more efficient for large data sets.
* But it needs sorting, which adds O(n log n) extra time for preprocessing.
* If data is frequently updated, linear search may be more useful

Exercise 7: Financial Forecasting

**Understand Recursive Algorithms:**

Recursion is a technique where a function calls itself to solve a smaller part of the problem. It breaks the main problem into sub problems.

Like Factorial and Fibonacci can be solved by recursion. As it breaks the bigger problem into sub problems.

Example:   
 For Factorial, we can write

factorial(n) = n \* factorial(n-1);

So the factorial(n) depends on factorial(n-1).

**Analysis**

Time Complexity**:**

* **Time Complexity:** O(n) - one recursive call per year.
* **Space Complexity:** O(n) - due to the recursive call stack.

**How to optimize:**

There are two ways to optimize.

First one is to avoid recursion and use an iterative approach.

Second one will be Dynamic Programming. If there is overlapping subproblems in the algorithm, then we can use dynamic programming as it stores intermediate results for an algorithm.

There are two ways of applying Dynamic Programming. First one is Top down approach and second one is Bottom up approach.