**Financial Forecasting**

**Understand Recursive Algorithms:**

**Q1. Explain the concept of recursion and how it can simplify certain problems.**

Recursion is a programming technique where a function calls itself to solve problems by breaking them into smaller, identical subproblems. It requires a base case to stop recursion and a recursive case that moves toward the base. This approach simplifies complex problems like tree traversal, mathematical calculations, and sorting algorithms through elegant, self-referential solutions.

**Analysis:**

**Q1. Discuss the time complexity of your recursive algorithm.**

In this recursive algorithm the time complexity is O(n).

n = number of periods.

Each recursive call reduces the periods by 1, making n recursive calls.

The recursive depth equals the number of periods.

Recursive Relation:  
T(n) = T(n-1) + O(1)

Base case: T(0) = O(1)

Recursive case: One recursive call plus constant work.

**Q2. Explain how to optimize the recursive solution to avoid excessive computation.**

Techniques:

1. Memoization (Top-Down Dynamic Programming):

It stores previously computed results in a cache to avoid recalculating the same values. Reduction in time complexity from O(2^n) to O(n) for overlapping subproblems. Use this when subproblems overlap.

1. Bottom-Up Approach:

Convert recursive solutions to iterative ones to eliminate call stack overhead. This is iterative approach. Eliminates call stack overhead entirely. Use it when depth is a concern.

1. Limiting Recursion Depth:

Add safeguards to prevent stack overflow for large inputs.