**Exercise 7: Financial Forecasting**

**1. Understand Recursive Algorithms:**

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

The process in which a function calls itself directly or indirectly is called recursion and the  
corresponding function is called a recursive function. Using a recursive algorithm, certain  
problems can be solved quite easily. Examples of such problems are Towers of Hanoi (TOH),  
Inorder/Preorder/Postorder Tree Traversals, DFS of Graph, etc. A recursive function solves a  
particular problem by calling a copy of itself and solving smaller subproblems of the original  
problems.  
Recursion simplifies problems by breaking them down into smaller, more manageable sub-  
problems that follow the same structure as the original problem. This allows for elegant and  
concise solutions, particularly for problems involving hierarchical or nested structures.  
a. The Towers of Hanoi is a classic problem where recursion provides a straightforward  
solution. The problem involves moving a set of disks from one peg to another,  
following specific rules.  
b. Tree traversal is a common problem in computer science where recursion can  
significantly simplify the implementation. Traversing a tree involves visiting all the  
nodes in a specific order (pre-order, in-order, or post-order).

**4. Analysis:**

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

The overall time complexity of the recursive algorithm is O(n), where n is the number of periods. This linear time complexity indicates that the algorithm's execution time increases linearly with the number of periods.

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

To optimize a recursive solution and avoid excessive computation, one can use  
memoization. Memoization stores the results of function calls in a cache. When the same inputs  
are encountered again, the cached results are used instead of recalculating them. This process  
reduces redundant calculations and improves performance by transforming the time complexity  
from potentially exponential to linear. Essentially, memoization ensures that each unique  
computation is performed only once, making the algorithm more efficient.