Financial Forecasting

**Concept of Recursion:** Recursion is a technique in programming where a function calls itself directly or indirectly in order to solve a problem. A recursive function typically has two main components:

* **Base Case(s):** These are the conditions under which the recursion ends. Without a base case, a recursive function would call itself indefinitely.
* **Recursive Case(s):** These define how the problem can be broken down into simpler instances of the same problem.

**Simplifying Problems with Recursion:** Recursion can simplify the solution to certain problems by breaking them down into smaller, more manageable subproblems. For example, problems involving hierarchical data structures (like trees) or problems that can be divided into similar subproblems (like the Fibonacci sequence) are often naturally suited to recursive approaches.

**Time Complexity:** The time complexity of the recursive algorithm is O(n), where n is the number of periods. This is because the function makes a single recursive call for each decrement of n.

**Optimizing the Recursive Solution:** To avoid excessive computation, particularly the overhead of recursive calls, we can optimize the solution using **memoization**. Memoization stores the results of expensive function calls and returns the cached result when the same inputs occur again.

By using memoization, we reduce the number of recursive calls, significantly improving the efficiency of the algorithm, especially for large values of n. The time complexity remains O(n), but the space complexity increases due to the storage required for memoization.