1. Recursion is a programming technique where a function calls itself to solve a problem. It breaks down a complex problem into smaller, more manageable sub-problems. Each recursive call handles a smaller portion of the problem until a base case is reached, which stops the recursion.

**How Recursion Simplifies Problems:**

1. **Divide and Conquer**: Recursion divides a problem into smaller sub-problems that are easier to solve.
2. **Code Simplification**: Recursive solutions can be more concise and easier to understand than iterative ones, especially for problems like tree traversal, factorial calculation, and Fibonacci sequence generation.
3. **Natural Fit**: Some problems, like those involving hierarchical data structures (e.g., trees), naturally lend themselves to recursive solutions.
4. **Time Complexity of the Financial Forecast Problem**

The time complexity of the recursive function is O(n), where ‘n’ is the number of periods. This is because the function makes a single recursive call for each period until it reaches the base case.

**Optimizing the Recursive Solution**

To optimize the recursive solution and avoid excessive computation, we can store the results of expensive function calls and reuse them when the same inputs occur again, reducing the number of computations.