Financial Forecasting

1. Understand Recursive Algorithms

- **Concept of Recursion**:
  - **Definition**: Recursion is a method where the solution to a problem depends on solutions to smaller instances of the same problem. A recursive function calls itself to solve smaller sub-problems.
  - **Benefits**: Simplifies complex problems by breaking them down into simpler sub-problems. Makes code more readable and easier to manage for certain problems.
  - **Drawbacks**: Can lead to excessive memory usage and stack overflow if not implemented correctly. Inefficient for problems with overlapping sub-problems unless optimized.

Analysis

- **Time Complexity**:
  - **Recursive Algorithm**: The time complexity is O(n) because each recursive call reduces the problem size by one, leading to n recursive calls.
  - **Optimized Recursive Algorithm (Memoization)**: The time complexity remains O(n) because each value is computed only once and stored for future use.
- **Optimization to Avoid Excessive Computation**:
  - **Memoization**: Store results of sub-problems in a memoization array to avoid redundant calculations. This reduces the number of recursive calls, saving time and preventing stack overflow for large input sizes.

Explanation:

- **Recursive Algorithm**:
  - Starts with the base case (n = 0) where the initial value is returned.
  - For each year, the future value is calculated by multiplying the previous year's value by (1 + growth rate).
  - This approach is simple but may lead to excessive computation for large n due to repeated calculations.
- **Optimized Recursive Algorithm**:
  - Uses a memoization array to store the results of each year's computation.
  - Before computing a year's value, it checks if the value is already computed and stored in the array.
  - This approach significantly reduces redundant calculations and improves efficiency.