**Exercise 7: Financial Forecasting**

**1. Understand Recursive Algorithms**

* **Recursion**: Recursion involves a function calling itself to solve smaller instances of the same problem. It simplifies complex problems by breaking them down into simpler, manageable subproblems. Recursive algorithms are often elegant and reduce code complexity but can lead to high memory usage and stack overflow if not handled properly.

**2. Setup**

* **Recursive Method**: Create a method that calculates the future value of an investment based on past growth rates. The method will use a recursive approach, where each call computes the value for a smaller subset of the problem until it reaches a base case.

**3. Implementation**

* **Recursive Algorithm**: Implement the recursive method to predict future values. For instance, if forecasting based on a constant growth rate, the method would call itself to compute the value for each time period until the desired future date is reached.

**4. Analysis**

* **Time Complexity**:
  + Recursive algorithms can have exponential time complexity if they make multiple recursive calls per step and recompute results for overlapping subproblems. In such cases, the complexity could be O(2^n) for straightforward recursion.
* **Optimization**:
  + **Memoization**: Store the results of expensive function calls and reuse them when the same inputs occur again. This avoids redundant computations and can significantly reduce time complexity.
  + **Dynamic Programming**: Use dynamic programming to build a solution iteratively, storing intermediate results to avoid recomputation. This transforms exponential time complexity into polynomial time complexity, such as O(n).