**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**Steps:**

1. **Understand Recursive Algorithms:**
   * Explain the concept of recursion and how it can simplify certain problems.
2. **Setup:**
   * Create a method to calculate the future value using a recursive approach.
3. **Implementation:**
   * Implement a recursive algorithm to predict future values based on past growth rates.
4. **Analysis:**
   * Discuss the time complexity of your recursive algorithm.
   * Explain how to optimize the recursive solution to avoid excessive computation.

**Solution:**

**Understand Recursive Algorithm**

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

Recursion is a programming technique where a method or function calls itself to solve smaller instances of the same problem.

Recursion simplifies programming by solving complex problems through repetition of smaller subproblems that follow the same structure. Instead of using loops or managing your own stack of operations, recursion lets the function call itself, letting the call stack manage the flow naturally.

**Analysis**

* Discuss the time complexity of your recursive algorithm.

The recursive algorithm has a time complexity of O(n), where n is the number of years, because it makes one call for each year until it reaches zero. It also has a space complexity of O(n) because of the recursive call stack. Although this method is simple and clear, it can be inefficient for large inputs. An iterative method or using Math.pow() is better, reducing time to O(1) and avoiding stack use.

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

To improve the recursive solution and reduce unnecessary computation, you can switch from recursion to iteration or use built-in math functions. For financial forecasting, recursion makes one call per year, leading to deep call stacks and slower performance. Instead, you can use an iterative loop that performs the same calculation without the extra burden of recursive calls.

**Output**

