Financial Forecasting – Recursive Algorithm Analysis

# 1. Time Complexity Analysis

The recursive method for forecasting future value is defined as:

public static double forecastRecursive(double currentValue, double growthRate, int years) {  
 if (years == 0) {  
 return currentValue;  
 }  
 return forecastRecursive(currentValue, growthRate, years - 1) \* (1 + growthRate);  
}

- The function makes one recursive call per year.

- Each call performs a constant-time multiplication.

🔹 Therefore, the time complexity is:  
  O(n), where n is the number of years.

🔹 Space complexity:  
  Each recursive call adds a new stack frame, so the space complexity is:  
  O(n)

# 2. Optimization Strategy

To avoid potential issues like stack overflow and improve performance, an iterative version can be used:

public static double forecastIterative(double currentValue, double growthRate, int years) {  
 for (int i = 0; i < years; i++) {  
 currentValue \*= (1 + growthRate);  
 }  
 return currentValue;  
}

✅ Benefits of the Iterative Approach:

- Eliminates recursion overhead

- Uses constant memory → O(1) space

- Faster for large values of n

# 3. Summary Comparison

|  |  |  |  |
| --- | --- | --- | --- |
| Approach | Time Complexity | Space Complexity | Best Use Case |
| Recursive | O(n) | O(n) | Simple cases, small n |
| Iterative | O(n) | O(1) | Larger n, better performance |

📌 For advanced scenarios (e.g., changing growth rates), consider dynamic programming or memoization to further optimize calculations.