**Financial Forecasting with Recursive Algorithms**

**Understanding Recursion**

**Recursion** is a programming technique where a function calls itself to solve smaller instances of the same problem. In financial forecasting, recursion can elegantly model compound growth patterns where each period's value depends on previous periods.

Advantages for Financial Forecasting:

* Naturally models compound growth patterns
* Simplifies implementation of multi-period calculations
* Makes formulas more readable by breaking them into smaller steps
* Easily handles variable growth rates across periods

**Analysis and Optimization**

Time Complexity Analysis

Version Time Complexity Space Complexity

Basic Recursive O(n) O(n) (call stack)

Variable Rates O(n) O(n) (call stack)

Memoized O(n) O(n) (memo array)

**Optimization Techniques**

1. **Memoization (Top-Down Dynamic Programming)**
   * Stores previously computed results to avoid redundant calculations
   * Shown in the calculateFutureValueMemoized implementation
   * Reduces time complexity from O(2^n) to O(n) for certain recursive patterns
2. **Iterative Approach (Bottom-Up)**
   * Eliminates call stack overhead
   * Constant O(1) space complexity
3. **Tail Recursion Optimization**
   * Some compilers can optimize tail-recursive functions to use constant stack space
   * Our basic implementation is already tail-recursive

**When to Use Recursion vs. Iteration**

**Use recursion when:**

* The problem naturally fits a recursive model (like compound growth)
* The code is more readable and maintainable
* You need to handle variable growth rates elegantly

**Use iteration when:**

* Dealing with very large periods (to avoid stack overflow)
* Performance is absolutely critical
* Implementing in languages without tail-call optimization