**Exercise 7: Financial Forecasting**

**Scenario:**

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

**1. Understand Recursive Algorithms**

**Recursion**:

* **Description**: Recursion involves a method calling itself with modified parameters until a base condition is met. It simplifies problems that can be divided into smaller, similar problems.
* **Advantages**: Simplifies code for problems like factorials, Fibonacci sequences, and divide-and-conquer algorithms.
* **Disadvantages**: Can lead to excessive computation if not optimized, and may cause stack overflow for deep recursion.

**2. Setup**

public class FinancialForecasting {

// Recursive method to calculate future value

public static double calculateFutureValue(double initialValue, double growthRate, int years) {

if (years <= 0) {

return initialValue;

}

return calculateFutureValue(initialValue \* (1 + growthRate), growthRate, years - 1);

}

public static void main(String[] args) {

double initialValue = 1000; // Initial investment

double growthRate = 0.05; // 5% annual growth rate

int years = 10; // Number of years

double futureValue = calculateFutureValue(initialValue, growthRate, years);

System.out.println("Future Value after " + years + " years: $" + futureValue);

}

}

**3. Implementation**

**Implementing Recursive Financial Forecasting**:

* The method calculateFutureValue uses recursion to compute the value of an investment after a number of years, applying a growth rate each year.

**4. Analysis**

**Time Complexity**:

* **Recursive Method**: O(n), where n is the number of years. Each recursive call processes one year.

**Optimizing Recursive Solution**:

* **Memoization**: Store previously computed values to avoid redundant calculations.
* **Iterative Approach**: For simple problems like this, an iterative approach may be more efficient and avoids potential stack overflow issues.