Exercise 7 – Financial Forecasting

# 1. Objective

To develop a recursive algorithm that forecasts future financial values based on past growth data.

# 2. Problem Statement / Scenario

You are developing a financial forecasting tool that predicts future values based on past data. A recursive approach will be used to model the compounding effect of growth over time.

# 3. Approach / Steps

## 3.1 Understand Recursive Algorithms

Recursion is a technique where a method calls itself to solve smaller instances of a problem. It can simplify problems that have repetitive patterns or can be broken into similar sub-problems.

## 3.2 Setup

Create a method that recursively computes the future value of an investment over a number of years.

## 3.3 Implementation

Implement both a simple recursive and an optimized memoized version of the forecasting function.

# 4. Code

## FinancialForecaster.java

public class FinancialForecaster {  
 public double forecast(double initialAmount, double growthRate, int years) {  
 if (years == 0) return initialAmount;  
 return forecast(initialAmount, growthRate, years - 1) \* (1 + growthRate);  
 }  
  
 public double forecastMemo(double initialAmount, double growthRate, int years, double[] memo) {  
 if (years == 0) return initialAmount;  
 if (memo[years] != 0) return memo[years];  
 memo[years] = forecastMemo(initialAmount, growthRate, years - 1, memo) \* (1 + growthRate);  
 return memo[years];  
 }  
}

## Main.java

public class Main {  
 public static void main(String[] args) {  
 FinancialForecaster forecaster = new FinancialForecaster();  
 double initialAmount = 10000.0;  
 double growthRate = 0.08;  
 int years = 5;  
  
 double futureValue = forecaster.forecast(initialAmount, growthRate, years);  
 System.out.printf("Future Value (Recursive): %.2f\n", futureValue);  
  
 double[] memo = new double[years + 1];  
 double futureMemo = forecaster.forecastMemo(initialAmount, growthRate, years, memo);  
 System.out.printf("Future Value (Memoized): %.2f\n", futureMemo);  
 }  
}

# 5. Analysis

Time Complexity:

• Basic Recursive: O(2^n) in the worst case without memoization.

• Memoized Recursive: O(n) due to storing intermediate results.

Recursive solutions can be elegant but may cause excessive computation. Memoization optimizes the recursive calls by storing already computed results.

# 6. Conclusion

Recursive algorithms provide a clean way to implement financial forecasting models. However, to avoid inefficiency, it’s essential to optimize with memoization where applicable.

# 7.Screenshot(output)

