Financial Forecasting

# Understand Recursive Algorithms

Recursion is a technique where a function calls itself to solve smaller sub-problems of the original problem. It simplifies problems like mathematical computations, tree traversals, and repetitive calculations.

In financial forecasting, recursion can calculate the future value of an investment by breaking it down year-by-year.

**Analysis**

Time Complexity:

- Basic Recursive: O(n)

- Memoized: O(n)

Space Complexity: O(n) due to recursion stack or memo array.

Optimization: Use memoization or iterative solutions to reduce redundant calls and avoid stack overflow in deep recursions.

**Implementation**

**Forecast.java**

public class Forecast {

public static double futureValue(int years, double initialAmount, double growthRate) {

if (years == 0) {

return initialAmount;

}

return futureValue(years - 1, initialAmount, growthRate) \* (1 + growthRate);

}

public static double futureValueMemo(int years, double initialAmount, double growthRate, double[] memo) {

if (years == 0) return initialAmount;

if (memo[years] != 0.0) return memo[years];

memo[years] = futureValueMemo(years - 1, initialAmount, growthRate, memo) \* (1 + growthRate);

return memo[years];

}

}

**ForecastTest.java (Main Class)**

public class ForecastTest {

public static void main(String[] args) {

int years = 5;

double initialAmount = 10000; // Starting investment

double growthRate = 0.10; // 10% annual growth

double forecast = Forecast.futureValue(years, initialAmount, growthRate);

System.out.printf("Recursive Forecast Value after %d years: ₹%.2f%n", years, forecast);

double[] memo = new double[years + 1];

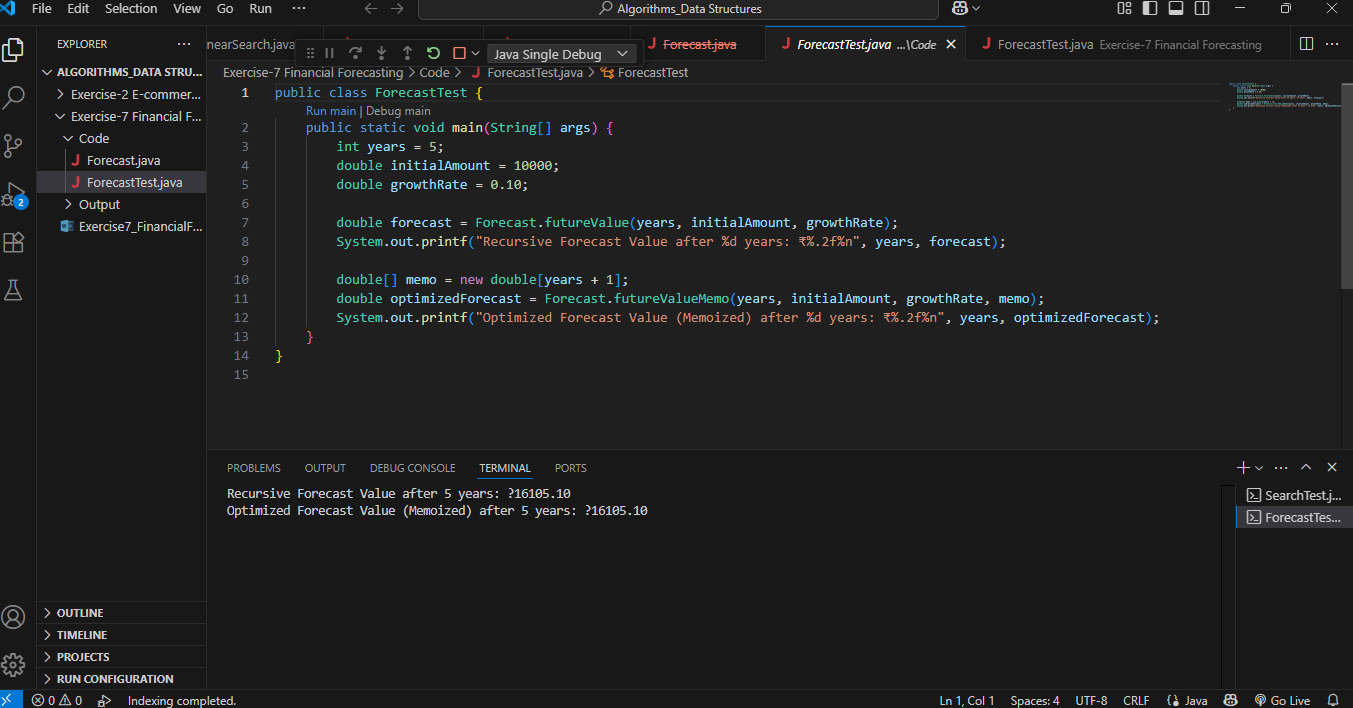
double optimizedForecast = Forecast.futureValueMemo(years, initialAmount, growthRate, memo);

System.out.printf("Optimized Forecast Value (Memoized) after %d years: ₹%.2f%n", years, optimizedForecast);

}

}

**Output**

****