**FINANCIAL FORECASTING**

**RECURSION:**

- A recursive algorithm is a function that calls itself to solve smaller instances of a problem.

- Useful for problems that can be broken into similar subproblems (e.g., Fibonacci, factorial, tree traversal).

**ADVANTAGES:**

- Simplifies code for divide-and-conquer problems.

- Easier to read and reason for certain logical structures**.**

**DISADVANTAGES:**

- May lead to high time complexity or stack overflow if not optimized.

- Requires base case to terminate recursion.

**PROGRAM:**

public class FinancialForecast {

public static double forecastValue(double initialValue, double[] growthRates, int years) {

if (years == 0) return initialValue;

return forecastValue(initialValue, growthRates, years - 1) \* (1 + growthRates[years - 1]);

}

public static void main(String[] args) {

double initialValue = 1000.0;

double[] growthRates = {0.05, 0.04, 0.06, 0.03, 0.05};

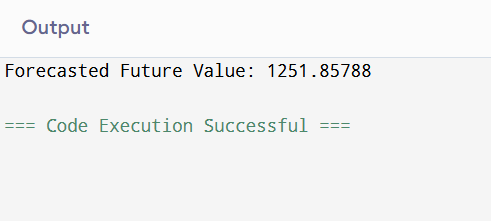
int years = growthRates.length;

double futureValue = forecastValue(initialValue, growthRates, years);

System.out.println("Forecasted Future Value: " + futureValue);

}

}



**TIME COMPLEXITY:**

- The time complexity of this recursive function is O(n), where n is the number of years.

- It makes one recursive call per year, so it's linear.

**OPTIMIZATION:**

- This recursion is already efficient (linear) with no repeated subproblems.

- However, for more complex recursive problems (e.g., Fibonacci), you can use:

- Memoization (caching results of subproblems).

- Iterative approaches (bottom-up loop).