**ALGORITHM DATASTRUCTURES**

**Exercise 7: Financial Forecasting**

**Scenario:**

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

**Steps:**

1. **Understand Recursive Algorithms:**
   * Explain the concept of recursion and how it can simplify certain problems.
2. **Setup:**
   * Create a method to calculate the future value using a recursive approach.
3. **Implementation:**
   * Implement a recursive algorithm to predict future values based on past growth rates.
4. **Analysis:**
   * Discuss the time complexity of your recursive algorithm.
   * Explain how to optimize the recursive solution to avoid excessive computation.

**DESCRIPTION:**

This Java program forecasts the **future financial value** based on the **current value**, **annual growth rate**, and **number of years** using a **recursive approach**. The user inputs the required values, and the program calculates compound growth recursively.

**RECURSION:**

Recursion is a technique in programming where a method calls itself to tackle a problem by dividing it into smaller, more manageable parts.

* Enables writing cleaner and shorter code for repetitive operations like factorials, Fibonacci series, or tree traversals.
* Reduces the need for loops in problems suited to a recursive approach.
* Improves readability when the problem naturally fits a recursive structure, such as compound interest calculations or hierarchical data processing.

**PROGRAM:**

import java.util.Scanner;

public class FinancialForecast {

public static double predictFutureValue(double currentValue, double growthRate, int years) {

if (years == 0) {

return currentValue;

}

return predictFutureValue(currentValue \* (1 + growthRate), growthRate, years - 1);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter current value (₹): ");

double currentValue = scanner.nextDouble();

System.out.print("Enter annual growth rate (%): ");

double ratePercent = scanner.nextDouble();

double growthRate = ratePercent / 100;

System.out.print("Enter number of years to forecast: ");

int years = scanner.nextInt();

double futureValue = predictFutureValue(currentValue, growthRate, years);

System.out.printf(" Future value after %d years: ₹%.2f\n", years, futureValue);

scanner.close();

}

}

**OUTPUT:**

****

**ANALYSIS:**

* **Time Complexity: O(n)**The function is invoked once per year, resulting in a total of n recursive calls when forecasting over n years.
* **Space Complexity: O(n)**Since each recursive call adds a new frame to the call stack, the memory usage increases linearly with the number of years.

**WAYS TO OPTIMIZE:**

**Use iteration:**  
Replace recursion with a for loop to save memory and prevent stack overflow.

**Avoid deep recursion:**  
Recursion is risky for large inputs; loops are safer and faster.