**Solution 7: Financial Forecasting**

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

**>> Explain the concept of recursion and how it can simplify certain problems.**

Concept of Recursion

* Definition: Recursion is a programming technique where a method calls itself to solve smaller instances of the same problem. The solution to the problem depends on solutions to smaller sub-problems.
* Base Case: Every recursive function must have a base case that terminates the recursion to prevent infinite loops.
* Recursive Case: The part of the function that includes the recursive call.

Simplification Using Recursion

* Problem Decomposition: Recursion simplifies problems by breaking them down into smaller, more manageable sub-problems.
* Examples: Common examples include computing factorials, Fibonacci sequences, and solving problems involving tree structures.

**Analysis**

**Financial Forecasting with Recursion**

For forecasting, let’s use a simple model where future value is calculated based on past growth rates. We’ll implement a recursive method to forecast future values.

* **Future Value Formula**: FV=PV×(1+r)^n
  + FV = Future Value
  + PV = Present Value
  + r = Growth Rate
  + n = Number of Periods

In the recursive approach, the future value can be calculated by recursively applying the growth rate until reaching the desired number of periods.

**>> Discuss the time complexity of your recursive algorithm.**

The recursive algorithm implemented in the calculateFutureValue method has a specific time complexity:

1. **Time Complexity**:
   * **Algorithm**: The recursive function calculateFutureValue performs a recursive call for each period (n). Each call involves a constant amount of work (multiplying the present value by (1 + rate)).
   * **Time Complexity**: O(n)
     + Here, n represents the number of periods. The time complexity is linear because the function performs n recursive calls, and each call takes constant time.
2. **Space Complexity**:
   * **Space Complexity**: O(n) due to the call stack.
     + Each recursive call adds a new frame to the call stack. In the worst case, the maximum depth of the recursion is equal to the number of periods n, leading to O(n) space complexity.

**>> Explain how to optimize the recursive solution to avoid excessive computation.**

* + **Iterative Approach**: Avoids recursion overhead and is generally preferred for problems with linear recursion depth.
  + **Memoization and Tail Recursion**: Useful in other recursive problems but not directly applicable in this simple case.

Using these optimizations and understanding the complexity helps in designing efficient algorithms and avoiding pitfalls associated with deep recursion.

**How to Run the code :**

* Run FutureForeCasting\_Main.java file