**Understand Recursive Algorithms:**

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

Ans: Recursion is a technique where a method **calls itself** to solve a smaller instance of the same problem. It simplifies problems that can be broken down into similar subproblems.

**Real-world Analogy:**

To calculate compound growth over n years, you can compute the value for n-1 years and multiply by (1 + growth rate).

**Analysis**

2.Discuss the time complexity of your recursive algorithm.

Ans: **Time Complexity**

* **Recursive calls:** O(n)
* Each call reduces years by 1 until 0.
* For n years, there are n+1 recursive calls.

3. Explain how to optimize the recursive solution to avoid excessive computation.

Ans: Problems with Deep Recursion

* Can lead to stack overflow for large n.
* Performance is not ideal if values are recomputed many times.

Use **Memoization** or **Iterative** Approach

Optimized code snippet:

*public static double calculateFutureValueIterative(double initialAmount, double growthRate, int years) {*

*double result = initialAmount;*

*for (int i = 0; i < years; i++) {*

*result \*= (1 + growthRate);*

*}*

*return result;*

*}*

**Time Complexity: O(n)**

Faster and avoids call stack buildup.