**Time Complexity of the Recursive Algorithm**

Recursion is a programming technique where a function calls itself to solve smaller parts of a bigger problem. It’s like breaking a tough task into smaller, more manageable steps—one at a time—until you reach a simple base case you can directly solve.

In the context of financial forecasting (like computing future value over n years), recursion is used to repeatedly apply a growth formula for each year until we hit year 0.

**Time Complexity: O(n)**

Let’s break it down:

For each **year**, the function calls itself once, decreasing the year count by 1.

This means, for n years, the function makes **n recursive calls**.

Each call performs one multiplication operation and one call, making the total work proportional to n.

**Key Points:**

The time complexity is **linear (O(n))**, because the function does the same amount of work for each year.

**No overlapping subproblems**: Each recursive call is unique—there's no reuse of previously computed values.

Because of this, we **don’t need extra memory** for memoization or dynamic programming.