### ****Analysis: Recursive Forecasting and Its Optimization****

While building this financial forecasting tool, I wanted to explore how recursion can be used in real-world problem-solving. Predicting future value based on annual growth felt like a suitable scenario to apply recursion, as the value for each year naturally depends on the result of the previous one.

I used a **recursive function with memoization** to avoid redundant calculations. Memoization helped improve performance by storing already computed values for each unique year, making future lookups instant. This was a good learning experience in applying optimization techniques to recursive logic.

In terms of **time complexity**, the recursive solution with memoization runs in **O(n)**, where n is the number of years. Each value is computed once and then reused. The **space complexity** is also **O(n)**, because of the recursion stack and the memo dictionary.

That said, I also realized that although recursion works and feels elegant, it's not always the most efficient choice, especially if performance or memory usage becomes a concern. An **iterative version** of the same logic performs just as well in terms of time but is more space-efficient, using only **O(1)** space.

Even better, I found that this kind of forecasting — since it's essentially compound interest — can actually be solved using a **mathematical formula**, which brings the complexity down to **O(1)**. It’s the fastest and cleanest method when intermediate yearly values aren’t needed.

So while recursion was a great way to understand the problem and implement a working solution, I’d probably choose the iterative or formula-based method in a real-world project where performance and scalability matter more.