**2\_Financial Forecasting**

**a. Understand Recursive Algorithms**

Recursion is when a method calls itself to solve smaller subproblems.  
It’s useful when a problem can be broken into identical subproblems, like:

* Factorial, Fibonacci etc.
* In this case, financial forecasting.

**b. Time Complexity**

* The recursive function has a time complexity of **O(n)**, where *n* is the number of years we're forecasting. That’s because with each recursive call, we’re moving one year closer to the base case. Simple, linear steps.
* If we’re dealing with a **huge number of years**, like 1,000,000, recursion can be risky. Each recursive call adds a new layer to the call stack, and too many of those can crash our program with a **stack overflow**.
* Also, in other problems like **Fibonacci**, recursion ends up doing the same work repeatedly, recalculating the same values over and over.

**c. How to Optimize**

* **Iteration** - When your logic is simple and linear (like forecasting year by year), just use a loop. It avoids the overhead of function calls and won’t overflow the stack.
* **Memoization** - Useful in problems with **repeating calculations** (like Fibonacci). You store already computed values so you don’t recalculate them.
* **Tail Recursion** - Some languages optimize tail-recursive functions automatically (like Python or Scala). Sadly, **Java doesn’t fully support** this optimization, so for safety, iteration is usually the better choice here.