**Understanding Recursive Algorithms in Financial Forecasting**

Recursion is a method in programming where a function calls itself with smaller inputs to solve a larger problem. Each recursive call breaks the problem into simpler subproblems, continuing until a **base case** is reached, which stops further calls.

In financial forecasting, especially when modeling compound growth over time, recursion simplifies the logic. For example, to calculate the future value of an investment with a fixed annual growth rate, we can define the problem recursively:

**FutureValue(years) = (1 + rate) × FutureValue(years - 1)**

This models how each year builds upon the previous year’s value. Recursion reflects the natural structure of time-based growth, making it easier to understand and implement forecasting scenarios such as interest accumulation or economic projections.

**Setup and Implementation**

To implement this in code, we define a recursive method that:

* Multiplies the current amount by the growth factor for that year.
* Reduces the remaining number of years by one.
* Stops when the number of years becomes zero (base case).

This recursive structure allows a clean, intuitive representation of how values grow annually based on historical growth rates.

**Time Complexity Analysis**

In the recursive approach:

* Each recursive call processes one year.
* The number of calls is directly proportional to the number of years.

**Time Complexity: O(n)**  
Where n is the number of years. The function makes one recursive call per year, leading to a **linear growth** in time.

**Optimization Strategies**

While recursion is elegant, it may not always be efficient for large inputs. In general:

* **Memoization** is used to cache and reuse results of repeated subproblems.
* However, in this forecasting scenario, each year produces a unique value, so memoization has limited impact.

**Potential Issues:**

* With very large n, the recursive depth may lead to a **stack overflow** due to too many nested function calls.

**Alternative:**

To improve performance and avoid memory issues, we can convert the recursive function to an **iterative version** using a simple loop. This eliminates function call overhead and works better for large-scale forecasts.