## **Financial Forecasting**

## 

## **Understanding Recursive Algorithms:**

**Recursion** is a method where a function calls itself to solve smaller subproblems of the original problem.

**Why recursion?**

* It simplifies problems that have a repetitive, self-similar structure (like growth over time).
* For financial forecasting, if we assume a fixed growth rate, we can model the future value as:  
   FV(n)=FV(n−1)×(1+r)

where n is the number of years, and r is the growth rate.

## **Analysis:**

### **➤ Time Complexity**

* **Naive recursion**:  
  + Time: **O(n) (**This is a linear recursion, not a branching one)
  + Space: **O(n)** (due to call stack)
* **Memoized recursion**:  
  + Time: **O(n)**
  + Space: **O(n)** (for memoization array)

To **optimize a recursive solution** and avoid excessive computation (especially in problems with **overlapping subproblems**), we use **Dynamic Programming** techniques like **Memoization** and **Tabulation**.

### **Memoization (Top-Down Approach)**

**Idea:** Store results of **already solved subproblems** in a lookup table (usually an array or map). So if the same input comes again, return the stored result instead of recomputing.

**How it avoids excessive computation:**

* Prevents recomputation by caching.
* Reduces time complexity from exponential (e.g., O(2^n)) to linear (e.g., O(n)).

### **Tabulation (Bottom-Up Approach)**

**Idea:** Instead of recursion, fill a table iteratively from base cases up to the desired result.

**How it avoids excessive computation:**

* No recursive calls.
* Directly builds solution from base case up.