**Financial Forecasting**

**Explain the concept of recursion and how it can simplify certain problems.**

**Recursion** is a programming technique where a function calls itself to solve a problem. It breaks down a problem into smaller, similar subproblems. A base case stops the recursion. Using a recursive algorithm, certain problems can be solved easily. Examples of such problems are Towers of Hanoi, Factorial, DFS of Graph etc.

For example:

**Factorial of a number** is a classic example of how recursion can simplify problem solving.

The factorial of a non-negative integer n is the product of all positive integers less than or equal to n.

If n is 0, the factorial is 1 (base case).

If n is greater than 0, the factorial of n is [n \* factorial (n-1)].

The function keeps calling itself with smaller values until it reaches the base case, then the results are multiplied upwards to compute the final factorial.

**Time Complexity Analysis:**

* 1. **Discuss the time complexity of your recursive algorithm.**

The recursive algorithm for financial forecasting has a time complexity of O(n) in worst case, where n is the no. of years for prediction. Explaining:

* The base case (yr == 0) takes constant time O(1).
* In recursive case, the function calls itself with yr - 1, leading to recursion until reaches the base case.
* However, the repeated calculations for the same (yr) cause redundant computations.

For e.g, to calculate future value for year 5: it needs the future value for yr 4, which requires the value for yr 3, and so on. This repetition happens for every yr in the prediction.

* 1. **Explain how to optimize the recursive solution to avoid excessive computation.**

The code already implements a technique called **memoization** to optimize the time complexity.

**Memoization:** This optimization technique improves performance by avoiding repeated calculations and storing previously computed values in a HashMap.

* **Without memoization,** the recursive algorithm has an exponential time complexity of O(n^n) due to redundant calculations.
* However, **by using memoization** to store intermediate results, the time complexity is reduced to O(n).