**Analysis :**

**Time Complexity of the Recursive Algorithm**

In this financial forecasting application, the future value is calculated using a recursive approach where each year’s value depends on the value of the previous year with a fixed growth rate. The method repeatedly calls itself to perform this calculation for the number of years specified.

**Time Complexity of the Recursive Algorithm:**

The recursive method forecast(price, year) makes exactly **one recursive call per year** until the base case (year == 0) is reached. Each call performs a constant amount of work — a simple arithmetic operation — before making the next call.

Therefore, the time complexity can be analyzed as follows:

* **Best Case:** O(n) – when the number of years n is small, and the recursion reaches the base case quickly.
* **Average Case:** O(n) – in normal cases, the function makes n recursive calls, one for each year.
* **Worst Case:** O(n) – even in the worst case, the recursion depth is equal to the number of years.

The function performs **linear recursive calls** in terms of the number of years, and hence the **overall time complexity is O(n)**.

This means the performance of the algorithm increases linearly with the number of years entered by the user.

**Optimizing recursive algorithms:**

* Recursive algorithms can bee optimized using Dynamic Programming but it is effective only when we have to recompute the same sub problem again and again.
* Using Bottom up Approach (using loop instead of recursively calling function) will reduce space complexity because recursion via memorization requires stack space.