Concept of Recursion:

Definition: Recursion is a method of solving problems where a function calls itself as a subroutine. This allows the function to be repeated several times as it can call itself during its execution.

Base Case: Every recursive function must have a base case to stop the recursion; otherwise, it will result in infinite recursion.

Simplifying Assumption: Each recursive call should simplify the problem, bringing it closer to the base case.

Benefits:

Simplicity: Recursion can simplify the code, making it easier to read and maintain.

Divide and Conquer: It breaks down complex problems into simpler sub-problems.

Analysis

Time Complexity:

Recursive Algorithm: The time complexity of this recursive algorithm is O(n), where n is the number of years. This is because the function calls itself n times, reducing the number of years by one each time.

Optimization:

While the current algorithm is straightforward, it can be optimized to avoid excessive computation in some scenarios. However, in this case, the recursive approach is already optimal with O(n) time complexity. For more complex recursive problems, techniques like memoization or dynamic programming can be used to optimize performance.

Memoization

Memoization involves storing the results of expensive function calls and reusing the cached result when the same inputs occur again.

Recursion can simplify problems by breaking them down into simpler sub-problems and calling the function itself.

The recursive algorithm provided has a time complexity of O(n), which is efficient for this specific problem.

Memoization can be used to optimize recursive solutions, especially for problems with overlapping sub-problems, to avoid redundant computations.

In the case of financial forecasting with a simple growth rate, the recursive approach is effective and straightforward to implement.