**Exercise 7: Financial Forecasting**

**1. Understand Recursive Algorithms**

**Concept of Recursion**

* **Definition**: Recursion is a programming technique where a function calls itself in order to solve a problem. The problem is divided into smaller instances of the same problem.
* **Base Case**: The condition under which the recursion stops.
* **Recursive Case**: The part where the function calls itself with a modified argument to move closer to the base case.

**How Recursion Simplifies Problems**

* **Problem Decomposition**: Recursion allows breaking down complex problems into simpler subproblems.
* **Code Readability**: For some problems, recursion can make the code more concise and easier to understand compared to iterative solutions.

**4. Analysis**

**Time Complexity of the Recursive Algorithm**

* **Time Complexity**: O(n) - The recursive function makes nnn recursive calls (where nnn is the number of years), each performing a constant amount of work.
* **Space Complexity**: O(n) - Each recursive call adds a new frame to the call stack, so the space complexity is proportional to the number of recursive calls.

**Optimizing the Recursive Solution**

* **Memoization**: Store intermediate results to avoid redundant calculations. This can transform the algorithm into a dynamic programming approach.
* **Tail Recursion**: If the language/compiler supports tail call optimization, rewriting the function to be tail-recursive can help avoid stack overflow issues.