**Exercise 7: Financial Forecasting**

**Concept of Recursion:**

* **Recursion** is a programming technique where a function calls itself to solve smaller instances of the same problem.
* Recursive algorithms typically have two parts:
  1. **Base Case**: A condition under which the recursion stops.
  2. **Recursive Case**: The part where the function calls itself with a modified argument.

**Advantages:**

* Simplifies the code for problems that can be divided into similar sub-problems.
* Often used in problems involving sequences or structures like trees and graphs.

**Example**: Calculating factorials, Fibonacci sequences, or mathematical problems that can be expressed in terms of similar sub-problems.

**Implementation:**

Please refer the code.

**Analysis**

**Time Complexity:**

* The time complexity of the recursive algorithm is O(n)O(n)O(n), where nnn is the number of years. This is because each recursive call reduces the number of years by 1, resulting in nnn calls.

**Space Complexity:**

* The space complexity is also O(n)O(n)O(n) due to the call stack used in recursion. Each recursive call consumes stack space, so for nnn recursive calls, the stack space grows linearly.

**Optimization:**

* **Memoization**: Store results of previously computed values to avoid redundant calculations. This technique converts the recursive solution into a more efficient dynamic programming solution.
* **Iterative Approach**: For this specific problem, an iterative approach may be more efficient. An iterative solution avoids the overhead of recursive function calls and stack space.
* **Recursive Approach**: Simplifies the code by breaking the problem into smaller instances, but may have high space complexity due to recursion depth.
* **Iterative Approach**: Often more efficient in terms of space and performance, especially for simple problems like financial forecasting.