Exercise7

1. Understanding Recursive Algorithms

Recursion Concept:

- **Definition:** Recursion is a method where a function calls itself in order to solve a problem. It simplifies complex problems by breaking them down into smaller, more manageable sub-problems.
- **Base Case:** The condition under which the recursion ends.
- **Recursive Case:** The condition under which the function calls itself to solve a smaller sub-problem.

Example Use Case:

- **Predicting Future Value:** Recursive algorithms can simplify the calculation of future values based on past data by iterating over the growth rate through recursive calls.

4. Analysis

Time Complexity:

- **Time Complexity:** O(n) where `n` is the number of years. Each recursive call performs a constant amount of work and there are `n` recursive calls.
- **Space Complexity:** O(n) due to the call stack depth. Each recursive call adds a new frame to the call stack.

Optimization:

- **Memoization:** To avoid redundant calculations, store intermediate results of sub-problems in a data structure (e.g., an array or map). This technique reduces the number of recursive calls by reusing previously computed results.