**Recursive Algorithms**

**Concept of Recursion**

* **Definition:** Recursion is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem. A recursive function calls itself with modified parameters until it reaches a base case.

Recursion can simplify the coding of complex problems by breaking them down into simpler subproblems. For example, calculating the factorial of a number or the Fibonacci sequence are classic examples of problems that are naturally recursive.

**Analysis**

**Time Complexity of the Recursive Algorithm**

* Time Complexity: The time complexity of this recursive algorithm is O(n), where n is the number of years. This is because the function makes one recursive call per year until it reaches the base case.
* Space Complexity: The space complexity is also O(n) due to the call stack storing each recursive call until the base case is reached.

**Optimizing the Recursive Solution**

* Memoization: One way to optimize the recursive solution is to use memoization, which stores the results of expensive function calls and returns the cached result when the same inputs occur again. However, in this specific problem, memoization is not necessary because each year's prediction depends directly on the previous year's value and is only computed once.
* Iterative Approach: Another optimization is to convert the recursive solution into an iterative one, which can reduce the space complexity to O(1) by eliminating the call stack overhead.