**1. Understanding the Problem**

Recursion is a technique where a function calls itself in order to solve a problem. The recursive approach typically involves breaking down a problem into smaller, more manageable sub-problems of the same type.

**How Recursion Simplifies Problems:**

* **Divides Problems:** Recursion breaks complex problems into simpler sub-problems.
* **Base Case:** Provides a straightforward solution for small instances, known as the base case.
* **Reduction:** Each recursive call works on a smaller subset of the problem, eventually reaching the base case.

**2. Analysis**

**Time Complexity:**

* The time complexity of the recursive algorithm is O(n), where n is the number of periods. This is because each recursive call reduces the problem size by 1 until it reaches the base case. In each call, a constant amount of work is done (multiplying by (1 + rate)).

**Optimizing the Recursive Solution:** Recursive algorithms can be inefficient if they involve redundant computations or deep recursion, which can lead to performance issues and stack overflow errors.

To optimize the recursive solution:

1. **Memoization:** Store results of recursive calls in a data structure to avoid redundant calculations.
2. **Iterative Approach:** For problems with simple linear recursion, consider using an iterative approach instead.