# time complexity of your recursive algorithm

The recursive future value prediction algorithm has:

* **Time Complexity**: **O(n)** (linear)
  + Each recursive call processes one period until reaching the base case (periods == 0).
  + For n periods, the function calls itself n times.
  + Example: Predicting 5 periods requires 5 recursive calls.
* **Space Complexity**: **O(n)** (due to call stack)
  + Each recursive call adds a stack frame until unwinding.

# Optimization techniques

To avoid excessive computation:

**A. Memoization (Caching Results)**

* **Problem**: The naive recursive approach recalculates the same values repeatedly.
* **Solution**: Cache intermediate results (e.g., store futureValue(t) at each time t).
* **Impact**: Reduces time complexity to **O(n)** with **O(n)** space.

**B. Iterative Approach**

* **Problem**: Recursion has stack overhead.
* **Solution**: Replace recursion with a loop.
* **Impact**: Retains **O(n)** time but uses **O(1)** space.

**C. Tail Recursion (Theoretical Optimization)**

* **Problem**: Java doesn’t optimize tail recursion (unlike Scala or Kotlin).
* **Hypothetical Solution**: If supported, tail recursion would use **O(1)** stack space.