Exercise 7: Financial Forecasting

1. **Understand Recursive Algorithms :** Recursion is a technique in which a function calls itself to solve smaller instances of a problem. Recursive solutions are especially useful for problems that can be broken down into smaller sub-problems of the same kind. In financial forecasting, recursion can be used to calculate future values when growth depends on previously computed values (like compound interest or exponential growth). Recursion is a programming technique where a function calls itself directly or indirectly to solve a problem. It is used to solve problems that can be broken down into smaller, similar subproblems.

2. **Key concept**:
   1. **Base Case:**
      The simplest case in recursion that ends the recursive calls. Without a base case, recursion would continue indefinitely and cause a stack overflow.
   2. **Recursive Case:**
      The part of the function where it calls itself to break down the problem into smaller instances.
   3. **Memory usage** :
      Using recursion can cost high memory usage because a stack of new iterations build one after other.

3. Analysis :

   Time Complexity

   The recursive function forecastFutureValue(currentValue, growthRate, years) performs one recursive call for each year until it reaches the base case (when years == 0).

   therefore, the time complexity is O(n). Where n is the number of years.

   Each recursive call performs a constant-time operation (multiplication and subtraction), and calls the function again with years – 1.

   Optimization

   Although this problem does not involve overlapping subproblems (so memoization is not necessary), recursion can still be inefficient for large values of n due to:

   High memory usage (stack frames for each call) Risk of stack overflow (for large n).

   Optimizations you can apply:

   1. Replace recursion with a simple loop.

   2. In languages that support tail call optimization (not Java), you can write the function in a tail-recursive form