Analysis

Time Complexity

* T(n) = T(n - 1) + O(1)
* Total function calls = n
* Therefore, Time Complexity: O(n)

Where n is number of years considered.

* Each recursive call adds a stack frame.
* Maximum depth of recursion = n (years)

Space Complexity: O(n)

This is due to the call stack space used by the recursion.

For large values of years (e.g., 10,000+), this could cause a StackOverflowError.

For optimizing the recursive solution, we can do either of the following:

1. Use iterative method (Space complexity will be O(1))
2. Use Tail Recursion

public double calculateFV(double pv, double rate, int years) {

return tailRecursiveHelper(pv, rate, years, pv);

}

private double tailRecursiveHelper(double pv, double rate, int years, double acc) {

if (years == 0) return acc;

return tailRecursiveHelper(pv, rate, years - 1, acc \* (1 + rate));

}