**Analysis**

**Time Complexity:**  
• The recursive function makes one call per year → T(n) = O(n)  
• Therefore, time complexity is O(n), where n is the number of years.

**Space Complexity:**  
• Each recursive call adds a frame to the call stack → O(n)  
• More years = more memory used → risk of stack overflow for large n

**Optimization Strategies**

**• Use iteration instead of recursion**  
→ Convert the logic into a loop to use constant space  
→ Reduces space complexity to O(1)

• **Avoid repeated calculations**  
→ Not needed in this case, but useful in recursive problems like Fibonacci  
→ Can be done using memoization or dynamic programming

• **Tail recursion (if supported)**  
→ Some compilers optimize tail-recursive functions  
→ Java does not support tail call optimization natively