

Recursion – Fibonacci Using Recursion

1. Introduction

The **Fibonacci series** is a sequence of numbers where each number is the **sum of the previous two numbers**.

It is one of the most common problems used to **explain recursion**, because the definition of Fibonacci naturally follows a recursive pattern.

2. What is the Fibonacci Series?

The Fibonacci sequence starts as:

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

Each number is calculated using:

$$F(n) = F(n-1) + F(n-2)$$

3. Why Fibonacci is Suitable for Recursion

Fibonacci is ideal for recursion because:

- The problem is defined in terms of itself
 - Each Fibonacci number depends on two smaller Fibonacci values
 - Clear base cases exist
 - Helps understand multiple recursive calls
-

4. Recursive Definition of Fibonacci

The Fibonacci number at position n is defined as:

$$F(n) = F(n-1) + F(n-2)$$

This definition includes:

- **Recursive case:** Sum of two previous Fibonacci numbers
 - **Base cases:** Values for $n = 0$ and $n = 1$
-

5. Base Case for Fibonacci

The **base case** stops recursion.

Base Case Conditions:

- If $n == 0$, return 0
- If $n == 1$, return 1

These values are already known and do not require further recursion.

6. Recursive Case for Fibonacci

The **recursive case** breaks the problem into smaller parts.

Recursive Step:

- Call the same function for $(n-1)$ and $(n-2)$
- Add the results

Each call reduces the value of n , moving closer to the base case.

7. Logic for Fibonacci Using Recursion (Plain English)

1. If the number is 0, return 0
2. If the number is 1, return 1
3. Otherwise, return Fibonacci of $(n-1)$ plus Fibonacci of $(n-2)$
4. Repeat until base cases are reached
5. Combine results while returning back

8. Step-by-Step Execution Example

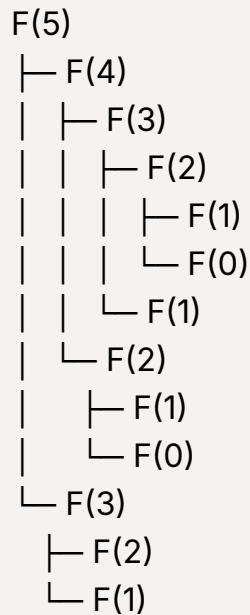
Find Fibonacci of 5:

$$\begin{aligned} F(5) &= F(4) + F(3) \\ &= (F(3) + F(2)) + (F(2) + F(1)) \\ &= ((F(2) + F(1)) + (F(1) + F(0))) + (F(2) + F(1)) \\ &= 5 \end{aligned}$$

Final result:

$$F(5) = 5$$

9. Call Stack Visualization



This shows **repeated function calls**, which affects performance.

10. Time and Space Complexity

Aspect	Complexity
Time Complexity	$O(2^n)$
Space Complexity	$O(n)$

The large time complexity is due to **repeated calculations**.

11. Advantages of Recursive Fibonacci

- Simple and clear logic
 - Closely matches mathematical definition
 - Good for understanding recursion trees
 - Useful for learning purposes
-

12. Limitations of Recursive Fibonacci

- Very slow for large values of n
 - Repeats same calculations multiple times
 - High time complexity
 - Risk of stack overflow
-

13. Recursive vs Iterative Fibonacci

Aspect	Recursive	Iterative
Readability	High	Medium
Performance	Poor	Good
Memory Usage	High	Low
Learning Purpose	Excellent	Moderate

14. Real-World Relevance

- Teaching recursion concepts

- Understanding recursion trees
 - Academic demonstrations
 - Basis for dynamic programming optimization
-

15. Summary

- Fibonacci series is defined recursively
 - Uses two recursive calls
 - Requires base cases for termination
 - Time complexity is exponential
 - Best used for learning recursion
-