Fibonacci Number using Recursion

What is a Fibonacci Number?

The Fibonacci sequence is a famous mathematical series in which each number is the sum of the two preceding ones. It's defined by the recurrence relation:

```
 F(0) = 0
F(1) = 1
F(n) = F(n-1) + F(n-2)  for n > 1
```

This generates a series like:

```
 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
```

Each number is the sum of the two before it.

This sequence appears frequently in nature (e.g., flower petals, pine cones, and spiral shells), in algorithms (like dynamic programming), and even in computer science problems related to recursion, time complexity, and optimization.

Approach: Recursion

Recursion is a technique where a function solves a problem by calling itself on smaller sub-problems.

In the context of Fibonacci:

To compute fib(n), we:
Ask: "What is fib(n-1)?"
Ask: "What is fib(n-2)?"
Return the sum of the two: fib(n) = fib(n-1) + fib(n-2)

This continues until we reach the base cases:

If n == 0, return 0
If n == 1, return 1

Time & Space Complexity

Time Complexity: O(2^n)
This is because each function call makes 2 recursive calls, leading to a binary tree of calls.

For large n, this becomes very inefficient, as many subproblems are solved repeatedly.

Space Complexity: O(n)

Although the number of calls is exponential, the maximum call depth is n.

So, the space used on the call stack is linear in the worst case.

Sample Outputs

Input n	Output fib(n)
0	0
1	1
2	1
3	2
5	5
6	8
7	13
10	55

JavaScript C C++ Java Python C#

```
var fib = function(n) {
    if (n <= 1)
        return n;
    return fib(n - 1) + fib(n - 2);
};
```

Video Course Discuss doubts Contribute Certificate

Fibonacci Number using Recursion - DSA Notes

Fibonacci Number using Recursion - DSA Notes