Proof of Time Complexity for the Recursive Fibonacci Algorithm

Mohammad Hayeri

10/14/24

Fibonacci Recursive Function

The naive recursive Fibonacci function can be defined as follows:

```
def fibonacci(n):
    if n <= 1:
return n
return fibonacci(n - 1) + fibonacci(n - 2)
```

Recurrence Relation

The time complexity $T(n)$ of the Fibonacci function can be expressed as a recurrence relation:

$$\begin{aligned} \text{For } n = 0 \text{ or } n = 1: \quad & T(n) = O(1) \\ \text{For } n > 1: \quad & T(n) = T(n-1) + T(n-2) + O(1) \end{aligned}$$

The $O(1)$ term accounts for the constant time taken to perform the addition and the function call overhead.

Simplifying the Recurrence Relation

Ignoring the constant term for simplicity, we can write:

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

This recurrence relation is similar to the Fibonacci sequence itself. To solve this, we can use the characteristic equation method.

Characteristic Equation

The characteristic equation for the recurrence relation $T(n) = T(n-1)+T(n-2)$ can be derived as follows:

1. Assume a solution of the form $T(n) = r^n$. 2. Substitute into the recurrence relation:

$$r^n = r^{n-1} + r^{n-2}$$

3. Dividing through by r^{n-2} (assuming $r \neq 0$):

$$r^2 = r + 1$$

4. Rearranging gives us the characteristic equation:

$$r^2 - r - 1 = 0$$

Solving the Characteristic Equation

Using the quadratic formula $r = \frac{-b\pm\sqrt{b^2-4ac}}{2a}$:

Here, $a = 1, b = -1, c = -1$:

$$r = \frac{1 \pm \sqrt{5}}{2}$$

The two roots are:

$$r_1 = \frac{1+\sqrt{5}}{2} \quad \text{(the golden ratio, approximately 1.618)}$$

$$r_2 = \frac{1-\sqrt{5}}{2} \quad \text{(approximately } -0.618\text{)}$$

General Solution

The general solution to the recurrence relation can be expressed as:

$$T(n) = Ar_1^n + Br_2^n$$

where A and B are constants determined by the initial conditions.

Asymptotic Behavior

As n grows large, the term involving r_2 (which is negative and less than 1 in absolute value) becomes negligible. Therefore, the dominant term is:

$$T(n) \approx Ar_1^n$$

Since r_1 is approximately 1.618, we can conclude that:

$$T(n) = O(r_1^n) = O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$$

Conclusion

The time complexity of the naive recursive Fibonacci algorithm can be expressed as:

$$\text{Time Complexity: } O(2^n) \quad (\text{since } r_1 \approx 1.618 < 2, \text{ but grows exponentially})$$

This analysis shows that the naive recursive Fibonacci algorithm has exponential time complexity due to the nature of the recursive calls, which can be modeled using a recurrence relation and solved using characteristic equations.