Fibonacci y Markov: aproximaciones computacionales en Rust.

Sergio García Macías

Resumen— Análisis y comparación de diferentes algoritmos para computar términos de la sucesión de Fibonacci.

Palabras clave— Fibonacci, computación, algoritmo, Markov, eficiencia, coste.

Sergio García Macías, Ingeniería Informática, Universidad de Huelva, sergio.garcia1@alu.uhu.es.

1. INTRODUCCIÓN

La sucesión de Fibonacci es una secuencia de números enteros en la que cada número es la suma de los dos anteriores. Formalmente, se define como:

$$\begin{aligned} f_0 = 0; f_1 = 1 \\ f_{n+1} = f_{n-1} + f_n \quad \forall n > 0 \end{aligned} \tag{1}$$

Con esta definición podemos obtener que los primeros términos de la sucesión son:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89...$$

La sucesión de Fibonacci aparece en numerosos patrones en la naturaleza, desde la disposición de hojas de un tallo, hasta la proporción del tamaño en un nautilus[1].

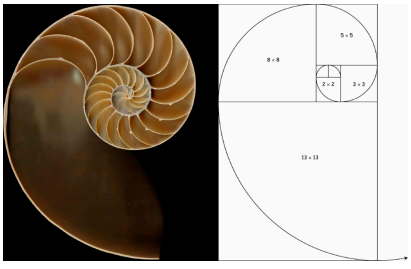

Fig. 1. Proporción de la concha del Nautilus.

Fig. 2. Disposición de las semillas de las margaritas.

Las semillas de las margaritas se disponen en forma de 21 y 34 espirales [2].

Además, los números de Fibonacci están estrechamente relacionados con el número áureo $\varphi = \frac{1+\sqrt{5}}{2} \approx 1,618033988...$.

Al calcular la relación entre el enésimo y el anterior número de Fibonacci, obtendremos un valor considerablemente próximo al valor real del número áureo:

$$\frac{55}{34} = 1.61764, \tag{2}$$

por lo que si usáramos números infinitamente grandes obtendríamos el valor exacto:

$$\lim_{n\to\infty} \frac{F_{n+1}}{F_n} = \varphi. \quad [3] \tag{3}$$

El número áureo ha sido considerado durante siglos un símbolo de belleza y proporción. Su presencia se ha observado en la arquitectura clásica, como en el Partenón, así como en el cuerpo humano, donde ciertas proporciones naturales se aproximan a este valor [2].

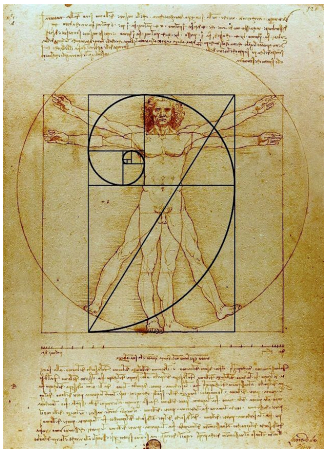

Fig. 3. Presencia del número aureo en el Hombre de Vitruvio de da Vinci.

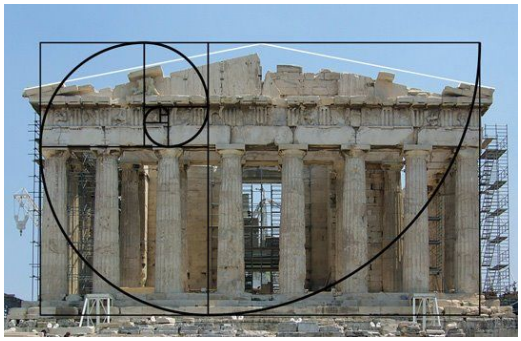

Fig. 4. Presencia del número aureo en el Partenón.

2. APROXIMACIONES

Para analizar la eficiencia computacional de diferentes métodos a la hora de calcular números de Fibonacci, se han realizado tres algoritmos diferentes en Rust [4]: recursivo, iterativo y mediante cadenas de Markov. Cada algoritmo correrá en la máquina durante 10 ms y al final del tiempo se comprobará cuál fue el último número que calculó.

Al final de este artículo se mostrarán las gráficas con los resultados obtenidos por cada algoritmo.

2.1. RECURSIVO

La implementación recursiva comúnmente usada sigue la definición matemática original, pero su coste computacional crece exponencialmente, tiene un coste $O(n^2)$ debido a la duplicación de llamadas [5]. Es adecuada solo para propósitos didácticos o valores pequeños de n. Además, si el lenguaje usado carece de TCO [5], la eficiencia de este algoritmo sería aún peor.

```
fn fib_rec(n: u64) -> u64 {
    if n <= 1 {
        n
    } else {
        fib_rec(n - 1) + fib_rec(n - 2)
    }
}
```

Código **1**. Código algoritmo recursivo.

2.2. ITERATIVO

El pseudocódigo correspondiente al algoritmo iterativo es el siguiente:

$$
\begin{aligned}
&a \to 0 \\
&b \to 1 \\
&\text{para } i = 2 \text{ hasta } n: \\
&\quad (a, b) \to (b, a + b)
\end{aligned}
\tag{4}
$$

Resultado: $F_n = b$ si $n > 0$ o a si $n = 0$

El algoritmo iterativo es una versión optimizada del cálculo recursivo. El cálculo de la secuencia se optimiza debido a la ausencia de llamadas recursivas cuyo número crece exponencialmente. El rendimiento de este nuevo algoritmo es $O(n)$.

```
fn fib_iterative(n: u64) -> u64 {
    let (mut a, mut b) = (0, 1);
    for _ in 0..n {
        let tmp = a;
        a = b;
        b = tmp + b;
    }
    a
}
```

Código **2**. Código algoritmo iterativo.

2.3. CADENAS DE MARKOV

Una cadena de Markov es una sucesión $\boldsymbol{u_0}, \boldsymbol{u_1}, \boldsymbol{u_2}, \ldots$ de vectores de probabilidad (sus componentes son no negativas y suman 1), junto con una **matriz estocástica A** (matriz cuadrada donde cada columna representa un vector de probabilidad) tal que:

$$\boldsymbol{u_1} = A\boldsymbol{u_0}, \boldsymbol{u_2} = A\boldsymbol{u_1}, \boldsymbol{u_3} = A\boldsymbol{u_2}, \ldots$$

Aunque estas matrices se usen en procesos estadísticos también pueden resultar útiles en procesos deterministas en los que podemos pasar de un estado a otro aplicando la misma transformación matricial. La sucesión de Fibonacci cumple con estos requisitos por lo que es ideal para modelarse con cadenas de Markov.

Tenemos que

$$\begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} * \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix} \quad \forall n > 0, \tag{5}$$

por tanto,

$$\begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n * \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \forall n > 0. \tag{6}$$

Aquí la matriz de transición describe cómo han de transformarse los elementos de los estados anteriores para obtener el nuevo estado. Para este caso usamos el siguiente código:

```
fn fib_matrix(n: u64) -> u64 {
    if n == 0 {
        0
    } else {
        let m = matrix_pow(
          [[1, 1], [1, 0]],
          n - 1
        );
        m[0][0]
    }
}
```

Código **3**. Código algoritmo Markov.

3. RESULTADOS

Los resultados al ejecutar los diferentes métodos para computar números de la sucesión de Fibonacci son los esperados. Para estas pruebas no se ha tenido en cuenta el overflow de números enteros para tamaños mayores a $2^{64} - 1$ por lo que el rendimiento es considerablemente más alto que el real esperado.

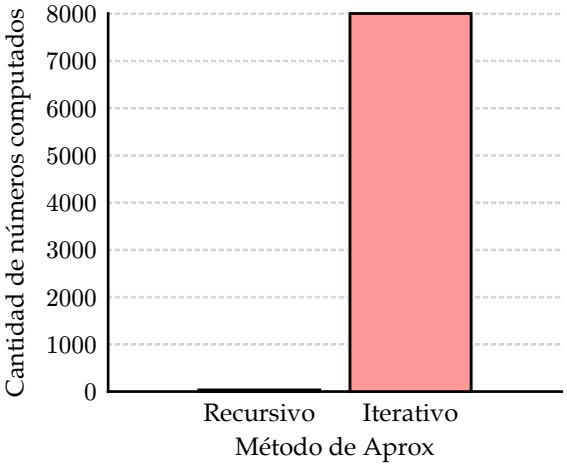

Fig. 5. Comparativa Recursivo vs Iterativo.

En la **Fig. 5** se comparan la cantidad de términos calculados entre los algoritmos **recursivo** e **iterativo**, obteniendo en 10ms 32 y 8006 términos respectivamente. El algoritmo **iterativo** logra obtener una mejora en el cálculo de x250. Esto se debe a que la complejidad es menor:

$$O(n) < O(n^2).$$

El algoritmo que usa cadenas de Markov para el cómputo de términos de la sucesión de Fibonacci, resulta ser mucho más eficiente que el iterativo, ya que el coste de este algoritmo puede reducirse a

$$O(\log n).$$ [3]

Aun así, debido a limitaciones de hardware, no es posible obtener tal eficiencia, aunque para los códigos propuestos estas limitaciones no se aplican, ya que no se está teniendo en cuenta el desbordamiento de los enteros usados (unsigned 64 bits).

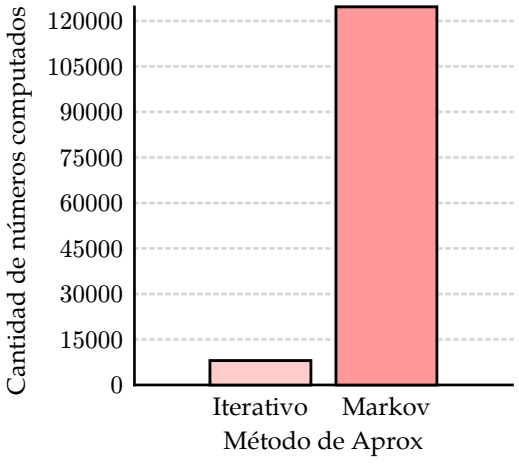

Fig. 6. Comparativa Iterativo vs Markov.

En la **Fig. 6** se muestra el número de elementos de la secuencia calculados por los métodos **iterativo** y **Markov**, 8006 y 124609 respectivamente.

4. CONCLUSIONES

A la hora de calcular aproximaciones de secuencias numéricas en escenarios deterministas, como puede ser la sucesión de Fibonacci, la recursión presenta una complejidad temporal $O(n^2)$ debido a la ramificación resultante de la recursión. Por otro lado, el método iterativo logra rebajar la complejidad hasta $O(n)$, ya que cada término ha de calcularse una única vez. Sin embargo, métodos de cálculo numérico fundamentados en álgebra lineal, como el basado en cadenas de Markov, permiten bajar el coste hasta $O(\log n)$ en el caso ideal. Esta comparación resalta la importancia de elegir el algoritmo adecuado según el contexto, especialmente donde la eficiencia es esencial.

REFERENCIAS

[1] Wikipedia contributors, «Golden ratio — Wikipedia, The Free Encyclopedia». [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Golden_ratio

[2] R. Knott, «Fibonacci Numbers and Nature». 2016.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, y C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: MIT Press, 2009.

[4] S. Klabnik y C. Nichols, *The Rust Programming Language*. San Francisco: No Starch Press, 2019.

[5] A. Sweigart, *The Recursive Book of Recursion: Ace the Coding Interview with Python and Recursion*. San Francisco: No Starch Press, 2022.

[6] D. R. Cox y H. D. Miller, *The Theory of Stochastic Processes*. London: Methuen, 1970.

[7] A. T. Bharucha-Reid, *Elements of the Theory of Markov Processes and Their Applications*. en McGraw-Hill Series in Probability and Statistics. New York: McGraw-Hill, 1960.

[8] R. Sedgewick y K. Wayne, *Algorithms*, 4th ed. Boston: Addison-Wesley, 2011.