Analisis y Diseño de Algoritmos

Marco Antonio Bastida Flores

06 Junio 2023

En el presente reporte se muestra el desarrollo del algoritmo para obtener el n-esimo numero de Fibonacci utilizando un algoritmo logaritmico.

1 Fibonacci

La serie de fibonacci se puede obtener de manera matricial utilizando la siguinete formula:

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n = \begin{bmatrix} f_n - 1 & f_n \\ f_n & f_n + 1 \end{bmatrix}$$

Se puede observar que al multiplicar $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ por si misma n veces, se obtiene f_n que representa el n-esimo numero de la serie de Fibonacci. Y $f_n + 1$ que representa el siguiente número en la serie de Fibonacci.
El código para la multiplicación de dos matrices cuadradas se muestra a continuación.

```python
def mul_square_matrix(A, B):
    C = [ [0,0], [0,0] ]
    C[0][0] = A[0][0]*B[0][0] + A[0][1]*B[1][0]
    C[0][1] = A[0][0]*B[0][1] + A[0][1]*B[1][1]
    C[1][0] = A[1][0]*B[0][0] + A[1][1]*B[1][0]
    C[1][1] = A[1][0]*B[0][1] + A[1][1]*B[1][1]
    return C
```

Dado que sabemos de antemano que nuestro algoritmo ocupará únicamente matrices cuadradas de $2 * 2$, podemos hacer uso de las equaciones de Strassen para resolver la multiplicación de una manera sencilla. La función mul_square_matrix(A, B) Toma dos matrices cuadradas A y B y retorna el resultado de la multiplicación A * B.

2 Exponenciación

Para multiplicar n veces la matriz base, utilizaremos el algoritmo de "squaring by exponentiation" que se repesenta de la siguiente manera:

$$x^n = \begin{cases} x(x^2)^{(n-1)/2} & Si\ n\ es\ impar \\ (x^2)^{n/2} & Si\ n\ es\ par \end{cases}$$

El codigo para obtener la potencia de un número usando exponenciación se muestra a continuación:

```
def exp_squaring(base, exp):
    if exp == 0:
        return 1
    while exp > 0:
        if exp % 2 == 1:
            return base * exp_squaring( base * base, (exp - 1) / 2 )
        else:
            return exp_squaring( base * base, exp / 2 )
```

Utilizando las condiciones de la definición se parte el problema por la mitad, utilizando un algoritmo recursivo donde el caso base es cuando el exponente es 0 y el resultado es 1.

3 Integración de algoritmos para calcular en n-esimo numero de Fibonacci

Ahora, se integran ambos modulos para obener la n-ésima potencia de la matriz base $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ para obtener la serie de Fibonacci.
El código se muestra a continuación.

```
base_matrix = [ [0,1], [1,1] ]
identity_matrix = [ [1,0], [0,1] ]

def mul_square_matrix(A, B):
    C = [ [0,0], [0,0] ]
    C[0][0] = A[0][0]*B[0][0] + A[0][1]*B[1][0]
    C[0][1] = A[0][0]*B[0][1] + A[0][1]*B[1][1]
    C[1][0] = A[1][0]*B[0][0] + A[1][1]*B[1][0]
    C[1][1] = A[1][0]*B[0][1] + A[1][1]*B[1][1]
    return C

def exp_square(base, n):
    if n == 0 :
        return identity_matrix
    while n > 0 :
        if n % 2 == 1:
            return mul_square_matrix(base, exp_square(\
                mul_square_matrix(base, base), (n - 1)/2 ) )
        else:
            return exp_square( mul_square_matrix(base, base), n / 2 )

def fibonacci(n):
    return exp_square(base_matrix, n)[1][0]
```

Se sustituye en el algoritmo de exponenciación la multiplicación de enteros por la multiplicacion de matrices usando la función mul_square_matrix(A, B) . El caso base será cuando el exponente n sea igual a 0 y en este caso se hace uso de la matriz identidad $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ para sustituir el caso base de multiplicar por 1, para ser consistentes con el algoritmo que exponencia numeros enteros y que se toma como base.

3.1 Resultados

Para validar que se obtiene la serie de Fibonacci con el algoritmo se hace uso del siguiente snippet:

for n in range(0, 30):
print(n, ": ", fibonacci(n))

En la Figura 1 se puede observar que la secuencia de los números de la serie de Fibonnacci se siguen correctamente.

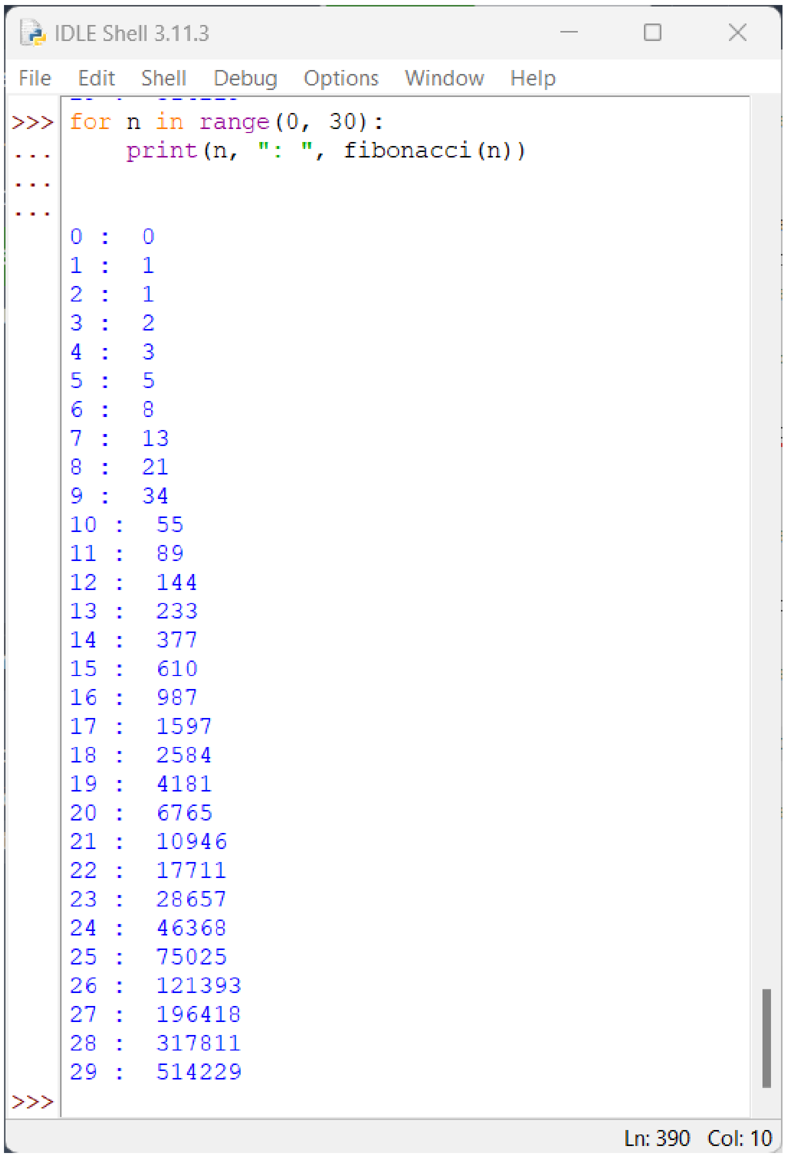

Figure 1: Validación del algoritmo de potencia por exponenciación

En la figura 2 se hace una prueba con un número de Fibonacci igual a 10,000 y se puede observar que el resultado se pinta en consola casi inmediatamente.

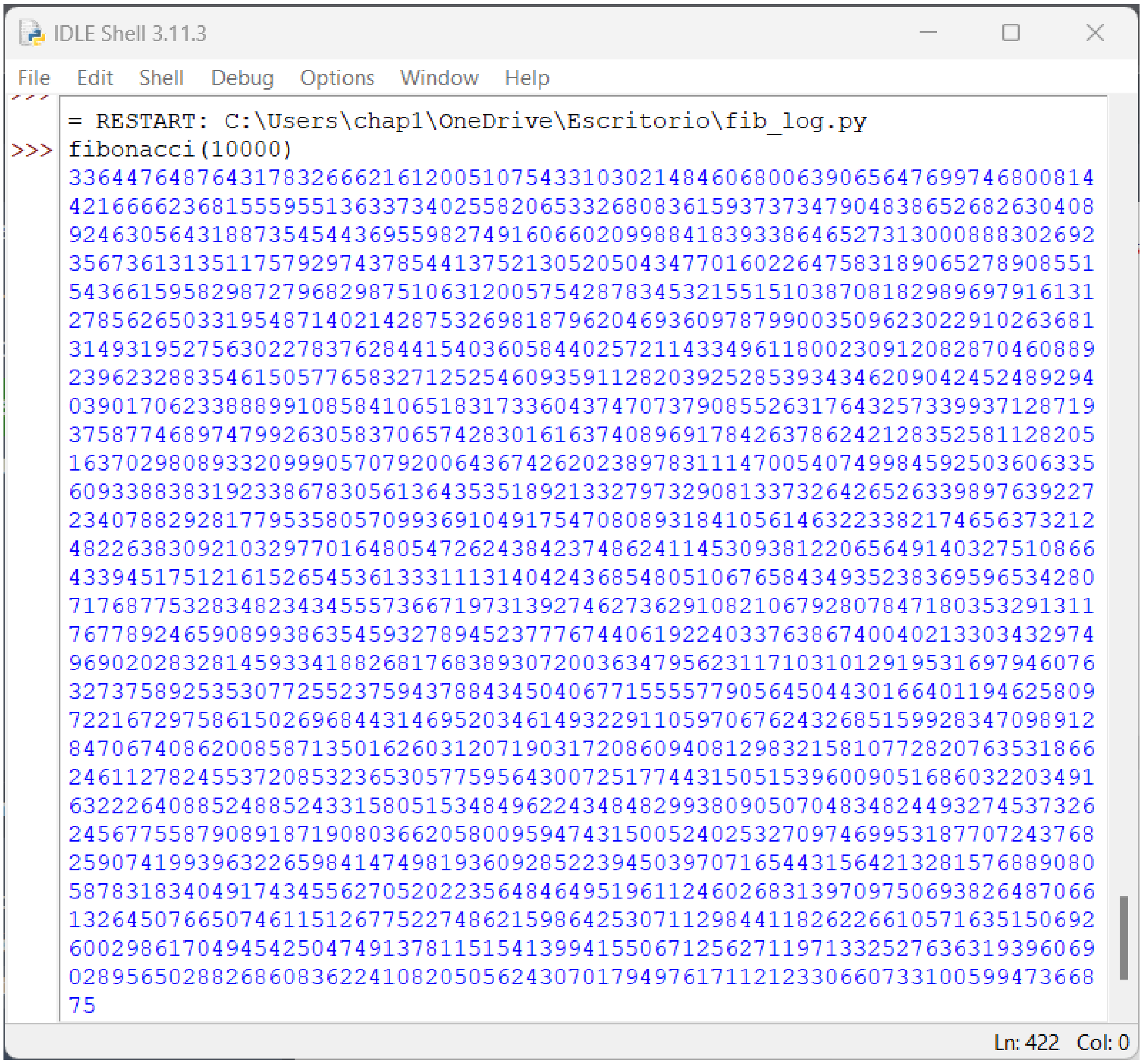

Figure 2: Algoritmo exponencial para obtener el 10,000 número de Fibonacci

Por ultimo se muestra en la figura 3 el calculo de el millonesimo numero de Fibonacci pudiendo observar que el tiempo que tomo para hacer el calculo fue de unos cuantos segundos.

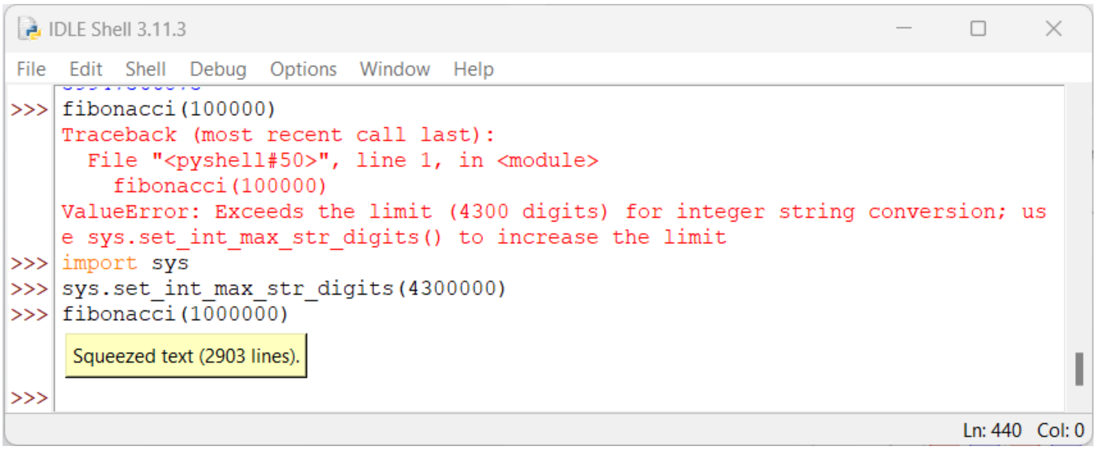

Figure 3: Algoritmo exponencial para obtener el 1,000,000 número de Fibonacci

View the Code on GitHub
Click to browse repository