Project Euler - Exercise 25

The problem

The Fibonacci Sequence is given by $f_n = f_{n-1} + f_{n-2}$ if $n \geq 3$, and $f_1 = f_2 = 1$.

What is the index of the first term in the Fibonacci Sequence to contain d digits?

Iterative solution

The algorithm with iterations it's straightforward:

FibonacciIndex(d):

1. $n \leftarrow 3$
   $f_{n-2} \leftarrow 1$
   $f_{n-1} \leftarrow 1$
   $f_n \leftarrow f_{n-2} + f_{n-1}$
2. While $f_n < 10^{d-1}$,
   $f_{n-2} \leftarrow f_{n-1}$,
   $f_{n-2} \leftarrow f_{n-1}$,
   $f_n \leftarrow f_{n-2} + f_{n-1}$,
   $n \leftarrow n + 1$
3. Return n

It's not so easy to tell with precision what is the complexity of this algorithm, once the growth of the Fibonacci Sequence is exponential. But since 10^{d-1} also grows exponentially, the complexity might be close to $O(d)$, i.e, a linear algorithm.

Solution with Linear Algebra

Mathematical explanation

If we define the linear transformation above

$$T = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

We notice that, if we apply it to the vector $\begin{bmatrix} f_{n-2} \\ f_{n-1} \end{bmatrix}$, we have

$$\Rightarrow T \begin{bmatrix} f_{n-2} \\ f_{n-1} \end{bmatrix} = \begin{bmatrix} f_{n-1} \\ f_{n-1} + f_{n-2} \end{bmatrix} = \begin{bmatrix} f_{n-1} \\ f_n \end{bmatrix}$$

Starting with the vector $v_0 = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and successively applying the transformation to the new result, we have:

$$\begin{aligned} Tv_0 = T \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} f_2 \\ f_3 \end{bmatrix} = v_1 \\ \Rightarrow Tv_1 = T(Tv_0) = T^2 v_0 = \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} f_3 \\ f_4 \end{bmatrix} = v_2 \\ \Rightarrow Tv_2 = T(T^2 v_0) = T^3 v_0 = \begin{bmatrix} 3 \\ 5 \end{bmatrix} = \begin{bmatrix} f_4 \\ f_5 \end{bmatrix} = v_3 \end{aligned}$$

If we continue the iterations, it's easy to notice that the pattern holds. Then, we have $T^n v_0 = v_n = \begin{bmatrix} f_{n+1} \\ f_{n+2} \end{bmatrix}$. This relation means that one way to calculate the n'th term of the Fibonacci Sequence is computing T^{n-2} and apply to v_0.

Since multiplication of matrices is a operation with high computational cost, we want to find a way to compute T^n with more efficiency.

Let's calculate the eigenvalues of T, i.e, the scalars $\lambda \in \mathbb{C}$ and $v \in \mathbb{R}^2$ such that:

$$\begin{aligned} Tv = \lambda v \qquad (1) \\ \Rightarrow Tx - \lambda v = 0 \\ \Rightarrow (T - \lambda I)v = 0 \end{aligned}$$

Since we want non trivial solutions, we want that the matrix $T - \lambda I$ be singular (not invertible). Then:

$$\begin{aligned} \det(T - \lambda I) = 0 \\ \det(\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}) \\ = \det(\begin{bmatrix} -\lambda & 1 \\ 1 & 1 - \lambda \end{bmatrix}) \\ = \lambda^2 - \lambda - 1 = 0 \end{aligned}$$

Then, the eigenvalues of T are $\lambda_1 = \dfrac{1+\sqrt{5}}{2}$ and $\lambda_2 = \dfrac{1-\sqrt{5}}{2}$. Since the eigenvalues are distinct, the eigenvectors v_1, v_2 associated are linearly independent.

Let $V = [v_1, v_2]$ the matrix where the eigenvectors are the columns and $\Lambda = \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix}$ the diagonal matrix of eigenvalues.

We can write the equation (1) as:

$$\begin{aligned} TV &= V\Lambda \\ \Rightarrow T &= V\Lambda V^{-1} \end{aligned}$$

since V is invertible (linearly independent eigenvectors).

Notice now, that if we want to calculate powers of T, we have:

$$\begin{aligned} T^2 &= (V\Lambda V^{-1})(V\Lambda V^{-1}) \\ &= V\Lambda(VV^{-1})\Lambda V^{-1} \\ &= V\Lambda I\Lambda V^{-1} \\ &= V\Lambda^2 V^{-1} \end{aligned}$$

This means, that if we want to compute T^n, we just have to compute $\Lambda^n = \begin{bmatrix} \lambda_1^n & \\ & \lambda_2^n \end{bmatrix}$.

Then, the n'th term of the Fibonacci Sequence is given by

$$\begin{bmatrix} f_{n-1} \\ f_n \end{bmatrix} = V \begin{bmatrix} \lambda_1^{n-2} & \\ & \lambda_2^{n-2} \end{bmatrix} V^{-1} v_0$$

A important observation, is since T is a symmetric matrix, we know that $V^{-1} = V^T$.

For this particular transformation, it can be shown that the following inequality holds:

$$f_n \leq 1.5\lambda_1^{n-2} \tag{2}$$

To demonstrate that, involves some messy calculations using the coordinates of its eigenvectors, so we are not going to explain that.

Then, one heuristic to find the first element of the Fibonacci Sequence with $d \in \mathbb{N}$ digits is using (2).

Let's calculate the n'th iteration which most approximates f_n to 10^{d-1}:

$$\begin{aligned} f_n \leq 1.5\lambda_1^{n-2} &\approx 10^{d-1} \\ \Rightarrow \log_{10}(1.5\lambda_1^{n-2}) &\approx \log_{10}(10^{d-1}) \\ \Rightarrow \log_{10}(1.5) + (n-2)\log_{10}(\lambda_1) &\approx d-1 \\ \Rightarrow n-2 &\approx \frac{d-1-\log_{10}(1.5)}{\log_{10}(\lambda_1)} \\ \Rightarrow n &\approx \frac{d-1-\log_{10}(1.5)}{\log_{10}(\lambda_1)} + 2 \end{aligned}$$

Then, if we want to calculate the first element of the Fibonacci Sequence which has d digits, a good initial value to start the iterations is given by

$$n_0 = \left\lfloor \frac{d-1-\log_{10}(1.5)}{\log_{10}(\lambda_1)} + 2 \right\rfloor$$

where $\lfloor . \rfloor$ is the floor function.

This choose of n guarantees that $f_n \leq 1.5\lambda_1^{n-2} \leq 10^{d-1}$.

The algorithm

Therefore, the algorithm to solve the problem, is given by:

FibonacciIndexLA(d):

1. Calculate the Spectral Decomposition of $T = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$, i.e, calculating the eigenvalues and eigenvector matrices, represented by Λ and V respectively;
2. Compute $n_0 = \left\lfloor \dfrac{d - 1 - \log_{10}(1.5)}{\log_{10}(\lambda_1)} + 2 \right\rfloor$, where λ_1 is the greatest eigenvalue of Λ;
3. Let $n \leftarrow n_0$ and compute:

$$\begin{bmatrix} f_{n-1} \\ f_n \end{bmatrix} \leftarrow T^{n-2} v_0 = V \Lambda^{n-2} V^{-1} v_0$$

   where $v_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$;
4. While $f_n < 10^{d-1}$, $\begin{bmatrix} f_{n-1} \\ f_n \end{bmatrix} \leftarrow \begin{bmatrix} f_n \\ f_{n+1} \end{bmatrix}$ and $n \leftarrow n + 1$;
5. Return n.

This should give the desired result.

Since the matrix T has dimension 2×2, which is fixed, then the operation of multiplication of matrices may have constant time, just as $\log_{10}$. And since the only step with a loop is (4.), which is really short because we estimated the initial value really close to the result, the algorithm may have a complexity approaching to computing power of numbers, which is much faster than the iterative solution.