13. Die Fibonacci-Zahlen

Die Fibonacci-Zahlen sind rekursiv definiert durch

$$f_0 = 0, \quad f_1 = 1, \quad f_{n+1} = f_n + f_{n-1}, \ (n \geq 1).$$

Der Anfang der Folge der Fibonacci-Zahlen ist daher

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

In Matrizen-Schreibweise lässt sich die Rekursionsformel für die Fibonacci-Zahlen so ausdrücken:

$$\begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix}.$$

Mit der Abkürzung $Q := \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ wird daher

$$\begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = Q \begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix} = Q^2 \begin{pmatrix} f_{n-1} \\ f_{n-2} \end{pmatrix} = \ldots = Q^n \begin{pmatrix} f_1 \\ f_0 \end{pmatrix},$$

also

$$\begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = Q^n \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Zur Berechnung von $Q^n \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ ist es nützlich, die Eigenwerte und Eigenvektoren von Q zu bestimmen. Dazu berechnen wir zunächst das charakteristische Polynom von Q.

$$\chi_Q(\lambda) = \det(Q - \lambda E) = \det \begin{pmatrix} 1-\lambda & 1 \\ 1 & -\lambda \end{pmatrix} = \lambda^2 - \lambda - 1.$$

Die Nullstellen dieses Polynoms sind $\lambda_{1/2} = \frac{1}{2} \pm \frac{1}{2}\sqrt{5}$. Wir führen folgende Bezeichnungen ein:

$$\begin{aligned} g &:= \lambda_1 = \tfrac{1}{2} + \tfrac{1}{2}\sqrt{5}, \\ \tilde{g} &:= \lambda_2 = \tfrac{1}{2} - \tfrac{1}{2}\sqrt{5}. \end{aligned}$$

Es gilt

$$g + \tilde{g} = 1 \quad \text{und} \quad \tilde{g} = -\frac{1}{g}.$$

Das Verhältnis $g : 1 = 1 : (g-1) \approx 1.618..$ ist der berühmte *goldene Schnitt*. Die Eigenvektoren zum Eigenwert λ sind Lösungen der Gleichung $(Q - \lambda E)v = 0$. Da

$$\begin{pmatrix} 1-g & 1 \\ 1 & -g \end{pmatrix} \begin{pmatrix} g \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \text{und} \quad \begin{pmatrix} 1-\tilde{g} & 1 \\ 1 & -\tilde{g} \end{pmatrix} \begin{pmatrix} \tilde{g} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

sind $\begin{pmatrix} g \\ 1 \end{pmatrix}$ und $\begin{pmatrix} \tilde{g} \\ 1 \end{pmatrix}$ Eigenvektoren von Q zu den Eigenwerten g und $\tilde{g}$. Wir stellen nun den Vektor $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ als Linearkombination der Eigenvektoren dar:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} = c \begin{pmatrix} g \\ 1 \end{pmatrix} + \tilde{c} \begin{pmatrix} \tilde{g} \\ 1 \end{pmatrix}.$$

Die zweite Komponente liefert $\tilde{c} = -c$. Eingesetzt in die erste Komponente ergibt sich

$$1 = cg - c\tilde{g} = c(g - \tilde{g}) = c\sqrt{5} \quad \Longrightarrow \quad c = \frac{1}{\sqrt{5}},$$

also

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{5}} \begin{pmatrix} g \\ 1 \end{pmatrix} - \frac{1}{\sqrt{5}} \begin{pmatrix} \tilde{g} \\ 1 \end{pmatrix}.$$

Damit erhält man

$$\begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = Q^n \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{5}} (Q^n \begin{pmatrix} g \\ 1 \end{pmatrix} - Q^n \begin{pmatrix} \tilde{g} \\ 1 \end{pmatrix}) = \frac{1}{\sqrt{5}} (g^n \begin{pmatrix} g \\ 1 \end{pmatrix} - \tilde{g}^n \begin{pmatrix} \tilde{g} \\ 1 \end{pmatrix}).$$

Durch Betrachtung der zweiten Komponente dieser Gleichung ergibt sich folgende explizite Formel für die n-te Fibonacci-Zahl:

$$f_n = \frac{1}{\sqrt{5}} (g^n - \tilde{g}^n) = \frac{1}{\sqrt{5}} (g^n - (-1)^n \frac{1}{g^n}).$$

Da $g \approx 1.61..$ und $\sqrt{5} \approx 2.23..$, gilt für alle $n \geq 0$

$$\left| \frac{1}{\sqrt{5}} \cdot \frac{1}{g^n} \right| < \frac{1}{2}.$$

Da die Fibonacci-Zahlen ganz sind, können wir also auch schreiben

$$(*) \qquad \boxed{f_n = \operatorname{round}(\frac{g^n}{\sqrt{5}})}$$

Dabei bedeutet $\mathrm{round}(x)$ für eine reelle Zahl x die nächstgelegene ganze Zahl. (Liegt x genau in der Mitte zwischen zwei ganzen Zahlen, so sei die Betrags-größere gewählt.)

Beispielsweise ist

$$\frac{g^{10}}{\sqrt{5}} = 55.003636.., \quad \frac{g^{11}}{\sqrt{5}} = 88.997752..,$$

also $f_{10} = 55$, $f_{11} = 89$.

Wachstum der Fibonacci-Zahlen

Mit der Formel $(*)$ lässt sich das Wachstum der Fibonacci-Zahlen leicht abschätzen. Logarithmieren zur Basis 10 ergibt

$$\log_{10}(f_n) \approx n \log_{10}(g) - \tfrac{1}{2}\log_{10} 5 \approx 0.209n - 0.349\,.$$

Die Anzahl der Dezimalstellen der n-ten Fibonacci-Zahl wächst also etwa linear mit n. Dies sei durch die folgenden Werte illustriert.

$$\begin{aligned}
f_{25} &= 75025, \\
f_{50} &= 1\,25862\,69025, \\
f_{100} &= 3\,54224\,84817\,92619\,15075, \\
f_{200} &= 28\,05711\,72992\,51014\,00376\,11932\,41303\,86771\,89525.
\end{aligned}$$

Die Kenntnis des Wachstums der Fibonacci-Zahlen erlaubt eine Anwendung auf den Euklidischen Algorithmus.

Komplexität des Euklidischen Algorithmus

Um den größten gemeinsamen Teiler zweier ganzer Zahlen x, y zu bestimmen, (o.B.d.A. sei $x > y > 0$), geht man wie folgt vor (vgl. §11): Man setze $a_0 := x$, $a_1 := y$ und führe sukzessive Divisionen mit Rest durch.

$$\begin{aligned}
a_0 &= q_1 a_1 + a_2, & 0 < a_2 < a_1, \\
a_1 &= q_2 a_2 + a_3, & 0 < a_3 < a_2, \\
&\vdots \\
a_{n-2} &= q_{n-1} a_{n-1} + a_n, & 0 < a_n < a_{n-1}, \\
a_{n-1} &= q_n a_n + 0.
\end{aligned}$$

Dann ist $\gcd(x, y) = a_n$.

Wir wählen jetzt speziell $x := f_{n+1}$ und $y := f_n$. Der Euklidische Algorithmus für dieses Zahlenpaar hat besonders einfache Gestalt:

$$\begin{aligned}
f_{n+1} &= 1 \cdot f_n + f_{n-1},\\
f_n &= 1 \cdot f_{n-1} + f_{n-2},\\
&\vdots\\
f_3 &= 1 \cdot f_2 + f_1,\\
f_2 &= 1 \cdot f_1 + 0.
\end{aligned}$$

Daraus ergibt zum einen, dass $\gcd(f_{n+1}, f_n) = 1$, d.h. f_{n+1} und f_n sind teilerfremd. Andrerseits erhält man aber auch durch Vergleich mit dem allgemeinen Fall (da alle $q_i \geq 1$), dass $x \geq f_{n+1}$, falls der Euklidische Algorithmus für $x > y > 0$ nach n Schritten endet. Somit folgt

$$\frac{g^n}{\sqrt{5}} < x \quad \Longrightarrow \quad n < (\log x + \tfrac{1}{2}\log 5)/\log(g) \;\approx\; 2.08 \log x,$$

also $n = \mathrm{O}(\log x)$, d.h. die Anzahl der Schritte, die beim Euklidischen Algorithmus nötig sind, wächst höchstens linear mit dem Logarithmus der Eingabezahlen.

Schneller Algorithmus zur Berechnung von Potenzen

Will man die Fibonacci-Zahlen durch Integer-Arithmetik mit der Formel $\begin{pmatrix} f_{n+1} \\ f_n \end{pmatrix} = Q^n \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ berechnen, so erfordert die naive Methode $n-1$ Matrizen-Multiplikationen. Es gibt aber eine wesentlich schnellere Methode zur Berechnung von Potenzen. Dabei ist es gleichgültig, ob es sich um Potenzen von Matrizen oder von Zahlen handelt. Wir setzen also voraus, dass x ein Element eines beliebigen Ringes ist und x^n berechnet werden soll. Wir unterscheiden zwei Fälle:

$$\begin{array}{llll}
\text{a)} & n \text{ gerade}, & n = 2m & \Longrightarrow \quad x^n = x^{2m} = (x^m)^2,\\
\text{b)} & n \text{ ungerade}, & n = 2m+1 & \Longrightarrow \quad x^n = x^{2m+1} = (x^m)^2 x.
\end{array}$$

Damit wird die Berechnung von x^n auf die Berechnung von x^m mit $m \leq n/2$ zurückgeführt. Durch rekursive Anwendung dieses Tricks erkennt man, dass man die Potenz x^n mit $\mathrm{O}(\log n)$ Multiplikationen berechnen kann.

Beispiel. Berechnung von x^{100}.

$$\begin{aligned}
&x^{100} = (x^{50})^2,\; x^{50} = (x^{25})^2,\; x^{25} = (x^{12})^2 x,\\
&x^{12} = (x^6)^2,\; x^6 = (x^3)^2,\; x^3 = x^2 x.
\end{aligned}$$

Es sind also nur 8 statt 99 Multiplikationen nötig.

Bemerkung. Zur Implementation des Algorithmus ist die Binär-Entwicklung des Exponenten n nützlich. Sei also

$$n = \sum_{i=0}^{k} b_i \cdot 2^{k-i} \quad \text{mit } b_i \in \{0,1\},\ b_0 = 1.$$

Mit $n_j := \sum_0^j b_i \cdot 2^{j-i}$ gilt dann $n_0 = 1$, $n_k = n$ und

$$n_{j+1} = 2n_j + b_{j+1} \quad \text{für } j = 0, \ldots, k-1.$$

Setzt man $P_0 := x$ und

$$P_{i+1} := \begin{cases} P_i^2 & \text{falls } b_{i+1} = 0, \\ P_i^2 x & \text{falls } b_{i+1} = 1 \end{cases}$$

für $i = 0, \ldots, k-1$, so gilt $P_i = x^{n_i}$ für alle i, insbesondere $P_k = x^n$.

Anwendung auf die Fibonacci-Zahlen ergibt, dass man f_n mit $\mathrm{O}(\log n)$ Multipliktionen berechnen kann. (Allerdings werden die zu multiplizierenden Zahlen immer größer.)