8.2 Analyse des EUKLIDischen Algorithmus

Ein kleines Problem hat sich im vorigen Abschnitt eingeschlichen: Zwar sind die Quotienten und Divisionsreste sicher durch die Eingabeparameter beschränkt; aber die Koeffizienten x_i und y_i sind auf den ersten Blick nicht kontrollierbar. Wie kann man garantieren, dass es hier nicht zu einem Überlauf bei der üblichen Ganzzahl-Arithmetik mit beschränkter Stellenzahl kommt? Nun, das Wachstum wird durch die folgende Überlegung kontrolliert:

Hilfssatz 2 *Für die Koeffizienten x_i und y_i im erweiterten Euklidischen Algorithmus gilt:*

(i) $x_i > 0$, *wenn i gerade,* $x_i \leq 0$, *wenn i ungerade, und* $|x_{i+1}| \geq |x_i|$ *für* $i = 1, \ldots, n$.

(ii) $y_i \leq 0$, *wenn i gerade,* $y_i > 0$, *wenn i ungerade, und* $|y_{i+1}| \geq |y_i|$ *für* $i = 2, \ldots, n$.

(iii) $x_{i+1}y_i - x_i y_{i+1} = (-1)^{i+1}$ *für* $i = 0, \ldots, n$; *insbesondere sind x_i und y_i stets teilerfremd für* $i = 0, \ldots, n+1$.

(iv) $|x_i| \leq |b|, |y_i| \leq |a|$ *für* $i = 0, \ldots, n+1$, *falls* $b \neq 0$ *bzw.* $a \neq 0$.

Beweis. (Nur angedeutet.) (i), (ii) und (iii) zeigt man durch Induktion. Aus $0 = r_{n+1} = |a|x_{n+1} + |b|y_{n+1}$ folgt dann $x_{n+1}|b$ und $y_{n+1}|a$. $\diamond$

Von besonderem Interesse ist, dass der Euklidische Algorithmus sehr effizient ist – die Zahl der Iterationsschritte wächst nur linear mit der *Stellenzahl* der Eingabeparameter, die gesamte Rechenzeit quadratisch. Es ist eine ziemlich genaue Analyse möglich, die wie folgt aussieht.

Die Divisionskette habe die Länge n (o. B. d. A. $b \neq 0$). Wie groß muss b dann mindestens sein? Es ist $r_n \geq 1, r_{n-1} \geq 2$ und $r_{i-1} \geq r_i + r_{i+1}$. Die FIBONACCI-Zahlen F_n sind rekursiv definiert durch

$$F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2} \quad \text{für } n \geq 2.$$

Durch Induktion erhält man also $r_i \geq F_{n+2-i}$, wobei der Induktionsanfang heißt: $r_n \geq 1 = F_2$, $r_{n-1} \geq 2 = F_3$; insbesondere folgt $|b| \geq F_{n+1}$. Anders formuliert:

Satz 2 (BINET 1841) *Für $a, b \in \mathbb{Z}$ mit $0 < b < F_{n+1}$ ergibt der Euklidische Algorithmus den größten gemeinsamen Teiler in höchstens $n-1$ Iterationsschritten.*

Zusatz. *Das gilt auch für $b = F_{n+1}$, außer wenn $a \equiv F_{n+2} \equiv F_n \pmod{b}$.*

Damit haben wir eine elegante mathematische Formulierung, aber noch keine Lösung. Jedoch ist das Wachstum der FIBONACCI-Zahlen sehr genau bekannt. Man kann es durch den goldenen Schnitt $\varphi = \frac{1+\sqrt{5}}{2}$ ausdrücken; es ist $\varphi^2 - \varphi - 1 = 0$.

Hilfssatz 3 *Für eine reelle Zahl $c \in \mathbb{R}$ und einen Index $k \in \mathbb{N}$ sei $F_k > c \cdot \varphi^k$ und $F_{k+1} > c \cdot \varphi^{k+1}$. Dann gilt $F_n > c \cdot \varphi^n$ für alle $n \geq k$.*

Beweis. (Durch Induktion.)

$$F_n = F_{n-1} + F_{n-2} > c\varphi^{n-1} + c\varphi^{n-2} = c\varphi^{n-2}(\varphi + 1) = c\varphi^n$$

für $n \geq k + 2$. $\diamond$

Korollar 1 $F_{n+1} > 0.43769 \cdot \varphi^{n+1}$ *für* $n \geq 2$.

Beweis.

$$\begin{aligned} \varphi^2 &= \varphi + 1 = \frac{3 + \sqrt{5}}{2}, \\ \varphi^3 &= \varphi^2 + \varphi = 2 + \sqrt{5}, \\ \varphi^4 &= \varphi^3 + \varphi^2 = \frac{7 + 3\sqrt{5}}{2}. \end{aligned}$$

Daraus folgt

$$\begin{aligned} \frac{F_3}{\varphi^3} &= \frac{2}{2 + \sqrt{5}} = \frac{2(\sqrt{5} - 2)}{1} = 2\sqrt{5} - 4 > 0.47, \\ \frac{F_4}{\varphi^4} &= \frac{3 \cdot 2}{7 + 3\sqrt{5}} = \frac{6(7 - 3\sqrt{5})}{49 - 45} = \frac{21 - 9\sqrt{5}}{2} > 0.43769 \end{aligned}$$

und daraus die Behauptung. $\diamond$

Korollar 2 *Seien $a, b \in \mathbb{Z}$ mit $b \geq 2$. Dann ist die Anzahl der Iterationsschritte im Euklidischen Algorithmus für* $\mathrm{ggT}(a, b)$ *kleiner als* $0.718 + 4.785 \cdot {}^{10}\log(b)$.

Beweis. Wenn die Divisionskette die Länge n hat, ist $b \geq F_{n+1}$,

$$b \geq F_{n+1} > 0.43769 \cdot \varphi^{n+1},$$

$${}^{10}\log(b) > {}^{10}\log(0.43769) + (n+1) \cdot {}^{10}\log(\varphi) > -0.35884 + 0.20898 \cdot (n+1),$$

also $n < 0.718 + 4.785 \cdot {}^{10}\log(b)$. $\diamond$

Etwas gröber, aber einfacher zu merken, ist die folgende Version:

Korollar 3 *Seien $a, b \in \mathbb{Z}$ mit $b \geq 2$. Dann ist die Anzahl der Iterationsschritte im Euklidischen Algorithmus für* $\mathrm{ggT}(a, b)$ *kleiner als fünfmal die Zahl der Dezimalstellen von b außer für $b = 8, a \equiv 5 \pmod 8$, wo man 5 Iterationsschritte braucht.*

Berücksichtigt man noch die Stellenzahl der vorkommenden Zahlen und den Aufwand für die Multiplikation und Division langer Zahlen, kommt man auf eine Rechenzeit, die quadratisch mit der Stellenzahl wächst, wie im folgenden gezeigt wird.

Hat a (bezüglich der Basis B) die Stellenzahl m und b die Stellenzahl p, so ist der Aufwand für die erste Division alleine schon $\leq c \cdot (m-p) \cdot p$; dabei ist c eine Konstante, die höchstens zweimal so groß ist wie die, die den Aufwand für die „Rückmultiplikation Quotient $\times$ Divisor“ beschreibt. Für B wird man bei heutigen Rechnerarchitekturen in der Regel 2^{32} annehmen, und als primitive Operationen werden die Grundrechenschritte Addition, Subtraktion, Multiplikation, Division mit Rest und Vergleich von einstelligen Zahlen (zur Basis B) gezählt. Zum Glück nehmen im Verlauf der euklidischen Divisionskette die zu dividierenden Zahlen exponentiell ab. Der Divisionsschritt

$$r_{i-1} = q_i r_i + r_{i+1}$$

benötigt noch $\leq c \cdot {}^B\log(q_i) {}^B\log(r_i)$ primitive Operationen, die gesamte Divisionskette also

$$\begin{aligned} A(a,b) &\leq c \cdot \sum_{i=1}^{n} {}^B\log(q_i) {}^B\log(r_i) \leq c \cdot {}^B\log|b| \cdot \sum_{i=1}^{n} {}^B\log(q_i) \\ &= c \cdot {}^B\log|b| \cdot {}^B\log(q_1 \cdots q_n). \end{aligned}$$

Das Produkt der q_i lässt sich weiter abschätzen:

$$|a| = r_0 = q_1 r_1 + r_2 = q_1(q_2 r_2 + r_3) + r_2 = \ldots = q_1 \cdots q_n r_n + \cdots \geq q_1 \cdots q_n;$$

also haben wir die grobe Abschätzung

$$A(a,b) \leq c \cdot {}^B\log|b| \cdot {}^B\log|a|.$$

Satz 3 *Die Anzahl der primitiven Operationen im Euklidischen Algorithmus für ganze Zahlen a und b der Stellenzahlen $\leq m$ ist $\leq c \cdot m^2$.*

Der Aufwand für den Euklidischen Algorithmus mit Input a und b ist also nicht wesentlich größer als der für die Multiplikation von a und b. Eine feinere Abschätzung soll hier nicht durchgeführt werden; ebensowenig werden mögliche Verbesserungen diskutiert. Erwähnt soll aber werden, dass ein Verfahren von LEHMER erlaubt, einen großen Anteil der Langzahl-Divisionen im Euklidischen Algortithmus durch primitive Operationen zu ersetzen.