KAPITEL 3

Grundeigenschaften der Ringe $\mathbb{Z}$ und $\mathbb{Z}/n\mathbb{Z}$

1. Die Landau-Symbole

Hat man einen Algorithmus oder ein Programm, so will man oft die Laufzeit bzw. Schrittzahl abschätzen. Dazu ist die Landausche O-Notation sinnvoll:

DEFINITION. *Sind $f : \mathbb{N} \to \mathbb{R}$, $g : \mathbb{N} \to \mathbb{R}$ Funktionen, so schreibt man*

$$f = O(g) \quad \text{oder} \quad f(n) = O(g(n)),$$

falls Zahlen $M \in \mathbb{R}_{>0}$ und $n_0 \in \mathbb{N}$ existieren, sodass gilt

$$|f(n)| \leq Mg(n) \text{ für alle } n \geq n_0.$$

(Man sagt: „f ist Groß-O von g“.)

Beispiele:

(1) Es gilt
$$123n^2 - 33n + 7 + \ln n = O(n^2).$$
(2) Hat man einen Algorithmus, ein Programm, das von einer natürlichen Zahl n abhängt, so nimmt man $f(n)$ als Laufzeit oder Schrittzahl und versucht $f(n)$ durch eine einfache Funktion $g(n)$ abzuschätzen.

Achtung: Obwohl man $f = O(g)$ schreibt, hat hier das Gleichheitszeichen nicht die übliche Bedeutung.

In verschiedenen Teilen der Mathematik, beispielsweise in der Analytischen Zahlentheorie, ist auch folgende Schreibweise nützlich:

DEFINITION. *Sind $f, g, h : \mathbb{N} \to \mathbb{R}$ Funktionen, so schreibt man*

$$f = g + O(h) \quad \text{oder} \quad f(n) = g(n) + O(h(n)),$$

falls $f(n) - g(n) = O(h(n))$ gilt. Anders formuliert: Definiert man $R(n)$ durch

$$f(n) = g(n) + R(n), \quad \text{so gilt} \quad R(n) = O(h(n)).$$

Beispiel: Für die divergierende harmonische Reihe $\sum_{k=1}^{n} \frac{1}{k}$ kann man schreiben

$$\sum_{k=1}^{n} \frac{1}{k} = \ln n + C + O(\frac{1}{n})$$

mit der Eulerschen Konstanten $C = 0.5772\ldots$.

Datei: ki_grund.tex. Version vom 29.10.2019

2. Zahldarstellungen

Sei $b \in \mathbb{N}_{\geq 2}$. Dann lässt sich jede natürliche Zahl n eindeutig schreiben als

$$n = \sum_{i=0}^{k-1} n_i \cdot b^i \quad \text{mit} \quad n_i \in \{0, 1, \ldots, b-1\}.$$

Man schreibt auch

$$n = (n_{k-1} n_{k-2} \ldots n_1 n_0)_b \quad \text{oder} \quad n = (n_{k-1}, n_{k-2}, \ldots, n_1, n_0)_b$$

und nennt dies die **b-adische Darstellung** der Zahl n. Ist $n_{k-1} \neq 0$, so nennt man k die **Stellenzahl** der Zahl n in der b-adischen Darstellung.

- Für $b = 10$ erhält man die übliche Dezimalstellung mit den Ziffern $0, 1, \ldots, 9$. Statt $(n_{k-1} \ldots n_1 n_0)_{10}$ schreiben wir $n_{k-1} \ldots n_1 n_0$.
- Für $b = 2$ hat man die Binärdarstellung mit den Ziffern 0 und 1. In Python3 gibt es die Darstellung
  $$n = 0b n_{k-1} \ldots n_1 n_0.$$
  Python3 liefert mit `bin(n)` die Binärdarstellung von n. Mit `int(...,2)` erhält man daraus wieder die Dezimaldarstellung
- Für $b = 16$ hat man die Hexadezimaldarstellung. Als Ziffern benutzt man $0, 1, \ldots, 9, a, b, c, d, e, f$ oder $0, 1, \ldots, 9, A, B, C, D, E, F$. In Python3 gibt es die Darstellung
  $$n = 0x n_{k-1} \ldots n_1 n_0.$$
  Mit `hex(n)` liefert Python3 die Hexadezimaldarstellung von n, mit `int(...,16)` erhält man wieder die Dezimaldarstellung.

DEFINITION. *Für $a \in \mathbb{R}$ sei $\lfloor a \rfloor$ die größte ganze Zahl, die $\leq a$ ist, also*

$$\lfloor a \rfloor = \max\{n \in \mathbb{Z} : n \leq a\}.$$

(Es gilt dann $\lfloor a \rfloor \leq a < \lfloor a \rfloor + 1$. Im Fall $a \geq 0$ erhält man $\lfloor a \rfloor$ durch Abschneiden des Nachkommaanteils von a.)

LEMMA. *Ist $n \in \mathbb{N}$ und $b \in \mathbb{N}_{\geq 2}$, so hat n in der b-adischen Darstellung*

$$k = \left\lfloor \frac{\ln n}{\ln b} \right\rfloor + 1$$

Stellen.

Beweis: Sei

$$n = \sum_{i=0}^{k-1} n_i \cdot b^i \quad \text{mit} \quad n_i \in \{0, 1, \ldots, b-1\} \quad \text{und} \quad n_{k-1} \neq 0.$$

Dann gilt

$$b^{k-1} \leq n \leq \sum_{i=0}^{k-1} (b-1) b^i = b^k - 1 < b^k,$$

und damit $(k-1) \ln b \leq \ln n < k \ln b$, also

$$k - 1 \leq \frac{\ln n}{\ln b} < k.$$

Dann ist

$$k - 1 = \left\lfloor \frac{\ln n}{\ln b} \right\rfloor,$$

und damit

$$k = \left\lfloor \frac{\ln n}{\ln b} \right\rfloor,$$

wie behauptet. ■

Mit dem Landau-Symbol kann man die Stellenzahl einer natürlichen Zahl n so abschätzen:

$$k = O(\ln n).$$

Stellenzahl

$$k = \lfloor \frac{\ln n}{\ln b} \rfloor + 1,$$

insbesondere ist $k = O(\ln n)$, d.h. die Stellezahl von n wächst wie $\ln n$. Die Anzahl der Bits von n ist die Anzahl der Stellen in der Binärdarstellung, also $\lfloor \frac{\ln n}{\ln 2} \rfloor + 1$.

Beispiel: Hier sind 3 Darstellungen einer 27-Bit-Zahl:

$$123456789 = (123456789)_{10} = (111010110111100110100010101)_2 = (75bcd15)_{16}.$$

(Binär- bzw. Hexadezimaldarstellung einer Zahl n liefert Maple mit den Funktionen 'convert(n,binary)' bzw. 'convert(n,hex)'.)

3. Division mit Rest

Ganze Zahlen kann man addieren, subtrahieren, multiplizieren, wobei eine Reihe von Gesetzmäßigkeiten erfüllt sind, wie beispielsweise die Assoziativität und Kommutativität von Addition und Multiplikation und Distributivgesetze. Algebraisch gesprochen bildet der ganzen Zahlen $\mathbb{Z}$ mit Addition und Multiplikation einen **Integritätsring**.

Außerdem gibt es für natürliche Zahlen die **Division mit Rest**: Teilt man $a \in \mathbb{N}_0$ durch $b \in \mathbb{N}$, so erhält man einen Quotienten $q \in \mathbb{N}_0$ und einen Rest $r \in \mathbb{N}_0$, also

$$a : b = q \text{ Rest } r,$$

wobei der Rest kleiner als b ist. Mathematisch kann man dies auch in der Form

$$a = qb + r \quad \text{mit} \quad q, r \in \mathbb{N}_0 \quad \text{und} \quad 0 \le r \le b - 1$$

schreiben.

Beispiel: Wir teilen 12345 durch 987 nach dem in der Schule gelernten Verfahren:

1	2	3	4	5	:	9	8	7	=	1	2
−	−	−	−								
1	2	3	4								
−	9	8	7								
−	−	−	−								
	2	4	7	5							
−	1	9	7	4							
	−	−	−	−							
		5	0	1							

12345 durch 987 ergibt also 12 Rest 501.

Man kann die Division mit Rest auch leicht auf ganze Zahlen ausdehnen, wobei wir uns hier auf den Fall $a \in \mathbb{Z}$ und $b \in \mathbb{N}$ beschränken: Definiert man

$$q = \left\lfloor \frac{a}{b} \right\rfloor \quad \text{und} \quad r = a - \left\lfloor \frac{a}{b} \right\rfloor b,$$

so gilt

$$a = qb + r \quad \text{mit} \quad q, r \in \mathbb{Z} \quad \text{und} \quad 0 \le r \le b - 1.$$

Für den Divisionsrest r hat sich eine eigene Bezeichnung eingebürgert:

Definition. *Für $a \in \mathbb{Z}$ und $b \in \mathbb{N}$ wird „a modulo b“ definiert als*

$$a \bmod b = a - \left\lfloor \frac{a}{b} \right\rfloor b.$$

Für $a \in \mathbb{Z}$, $b \in \mathbb{N}$ ist also

$$a = \left\lfloor \frac{a}{b} \right\rfloor b + (a \bmod b) \quad \text{und} \quad 0 \leq (a \bmod b) \leq b - 1.$$

Bemerkungen:

(1) Leider taucht die Bezeichung „mod“ in der Vorlesung in zwei Bedeutungen auf. In der obigen Definition ist „mod“ eine Funktion

$$\bmod : \mathbb{Z} \times N \to \mathbb{Z}, \quad (a, b) \mapsto a - \left\lfloor \frac{a}{b} \right\rfloor b,$$

die aber traditionell in der Form „$a \bmod b$“ geschrieben wird. Um $a \bmod b$ als Funktionswert zu kennzeichnen schreiben wir auch manchmal $(a \bmod b)$.

(2) Statt $a \bmod b$ findet sich auch mitunter die Schreibweise $a\%b$, also

$$a\%b = a \bmod b.$$

für $a \bmod b$.

(3) In der Programmiersprache Python3 erhält man für $a \in \mathbb{Z}$ und $b \in \mathbb{Z}$ den Rest $a \bmod b$ als `a%b`, den Quotienten $\left\lfloor \frac{a}{b} \right\rfloor$ als `a//b`.

(4) Die Division mit Rest macht den Ring der ganzen Zahlen $\mathbb{Z}$ algebraisch gesprochen zu einem **euklidischen Ring**.

Dividiert man $a \in \mathbb{Z}$ durch $b \in \mathbb{N}$, so erhält man einen Quotienten q und einen Rest r, sodass gilt

$$a = qb + r \quad \text{mit} \quad q, r \in \mathbb{Z} \quad \text{und} \quad 0 \leq r \leq b - 1.$$

Das folgende Lemma zeigt, dass q und r durch die Bedingungen $a = qb + r$ und $0 \leq r \leq b - 1$ schon eindeutig bestimmt sind:

LEMMA. *Zu $a \in \mathbb{Z}$ und $b \in \mathbb{N}$ gibt es genau eine Zahl $q \in \mathbb{Z}$ und eine Zahl $r \in \mathbb{Z}$, sodass gilt*

$$a = qb + r \quad \textit{und} \quad 0 \leq r \leq b - 1,$$

nämlich

$$q = \left\lfloor \frac{a}{b} \right\rfloor \quad \textit{und} \quad r = (a \bmod b).$$

Beweis: Wir wissen bereits, dass $q = \left\lfloor \frac{a}{b} \right\rfloor$ und $r = (a \bmod b) = a - \left\lfloor \frac{a}{b} \right\rfloor b$ die Gleichung $a = qb + r$ und Abschätzung $0 \leq r \leq b - 1$ erfüllen. Seien nun umgekehrt $q, r \in \mathbb{Z}$ gegeben mit

$$a = qb + r \quad \text{und} \quad 0 \leq r \leq b - 1.$$

Division durch b liefert

$$\frac{a}{b} = q + \frac{r}{b}.$$

Wegen $r \geq 0$ ist

$$q \leq \frac{a}{b},$$

wegen $r \leq b - 1$ ist

$$\frac{a}{b} = q + \frac{r}{b} \leq q + \frac{b-1}{b} < q + \frac{b}{b} = q + 1,$$

also

$$q \leq \frac{a}{b} < q + 1.$$

Damit folgt

$$q = \left\lfloor \frac{a}{b} \right\rfloor \quad \text{und damit} \quad r = a - qb = a - \left\lfloor \frac{a}{b} \right\rfloor b = (a \bmod b).$$

Dies beweist unsere Behauptung. ■

4. Teilbarkeit

DEFINITION. *Für $a, b \in \mathbb{Z}$ sagt man „a **teilt** b“ (oder „a ist ein **Teiler** von b“ oder „b ist ein **Vielfaches** von a“) und schreibt $a \mid b$, falls ein $c \in \mathbb{Z}$ existiert mit $b = ca$. Teilt a die Zahl b nicht, so schreibt man $a \nmid b$.*

Eigenschaften: Folgende Eigenschaften sind leicht einzusehen für $a, b, c, d \in \mathbb{Z}$:

(1) $a \mid a$, $1 \mid a$, $a \mid 0$.
(2) $a \mid b, b \mid c \Longrightarrow a \mid c$ (Transitivität).
(3) $a \mid b \iff \pm a \mid \pm b$.
(4) $a \mid b$ und $b \mid a \iff a = \pm b$.
(5) $a \mid 1 \iff a = \pm 1$.
(6) $0 \mid a \iff a = 0$.
(7) $d \mid a$ und $d \mid b \Longrightarrow d \mid ma + nb$ (für alle $m, n \in \mathbb{Z}$).
(8) Für $a \neq 0$: $a \mid b \iff \frac{b}{a} \in \mathbb{Z}$.
(9) Für $a \neq 0$: $a \mid b \iff b \bmod a = 0$.

Bemerkung: Die letzte Eigenschaft ist für das Rechnen nützlich um Teilbarkeit zu testen, wenn man Divisionsreste ausrechnen kann.

5. Primzahlen und der Fundamentalsatz der Arithmetik

Die Zahlen ± 1 haben nur die beiden Teiler ± 1. Alle ganzen Zahlen $a \neq \pm 1$ haben mindestens die vier Teiler $\pm 1, \pm a$, weswegen man diese Teiler auch „triviale Teiler“ nennt.

DEFINITION. *Eine natürliche Zahl $p > 1$ heißt* **Primzahl**, *wenn die einzigen Teiler von p die Zahlen $\pm 1, \pm p$ sind, d.h. wenn p nur triviale Teiler hat.*

Die ersten Primzahlen sind

$$2, \quad 3, \quad 5, \quad 7, \quad 11, \quad 13, \quad 17, \quad 19, \quad 23, \quad 29, \quad 31, \quad 37, \quad 41, \quad 43, \quad 47, \quad \ldots$$

Von grundlegender Wichtigkeit ist folgender Satz:

SATZ (Fundamentalsatz der Arithmetik). *Jede ganze Zahl $n \neq 0$ lässt sich eindeutig (bis auf die Reihenfolge der Faktoren) schreiben als*

$$n = \pm p_1^{e_1} p_2^{e_2} \ldots p_r^{e_r}$$

mit paarweise verschiedenen Primzahlen $p_1, p_2, \ldots, p_r$ und natürlichen Zahlen $e_1, e_2, \ldots, e_r$ (und $r \geq 0$).

Bemerkungen:

(1) Die Eigenschaften des Fundamentalsatzes der Arithmetik heißt in der Sprache der Algebra: Der Ring der ganzen Zahlen $\mathbb{Z}$ ist ein **faktorieller Ring**.
(2) Gerade wenn die Primfaktorzerlegung mehrerer Zahlen gleichzeitig betrachtet wird, schreibt man oft auch
$$\prod_i p_i^{e_i} \quad \text{mit} \quad e_i \geq 0,$$
wobei dann p_i eventuell auch alle Primzahlen durchlaufen kann. Eine andere Schreibweise ist
$$\prod_p p^{e_p},$$
wobei p alle Primzahlen durchläuft und nur endlich viele $e_p \in \mathbb{N}_0$ von 0 verschieden sind.
(3) Trotz der Wichtigkeit der Primfaktorzerlegung natürlicher Zahlen, ist bis heute kein wirklich schnelles Verfahren zur Primfaktorzerlegung bekannt.
(4) Die Schwierigkeit, die Primfaktorzerlegung natürlicher Zahlen zu bestimmen, wird zur Konstruktion von Verschlüsselungsverfahren benutzt.

Überlegungen für ein naives Faktorisierungsverfahren: Wir wollen die Primfaktorzerlegung der natürlichen Zahl n finden und teilen nacheinander alle Primteiler p_i bzw. Primzahlpotenzen $p_i^{e_i}$ heraus. Im i-ten Schritt haben wir

$$n = p_1^{e_1} \dots p_{i-1}^{e_{i-1}} n_i \quad \text{mit Primzahlen} \quad p_1 < p_2 < \dots < p_{i-1},$$

wobei n_i keinen (echten) Teiler $\leq p_{i-1}$ hat. Wir starten mit $n_1 = n$.

(1) Ist $n_i = 1$, so sind wir fertig und haben
$$n = p_1^{e_1} \dots p_{i-1}^{e_{i-1}}.$$

(2) Sei nun $n_i > 1$ vorausgesetzt. Wir suchen nach dem kleinsten Teiler d_i von n_i. Da n_i keinen Teiler $\leq p_{i-1}$ hat, können wir die Suche auf $d_i > p_{i-1}$ einschränken.
   (a) Finden wir keinen Teiler d mit $p_{i-1} < d \leq \sqrt{n_i}$, so ist $n_i = p_i$ eine Primzahl, denn aus $n_i = m_1 m_2$ mit $m_1, m_2 > 1$ folgt $m_1 \leq \sqrt{n_i}$ oder $m_2 \leq \sqrt{n_i}$, was nicht geht. Auch in diesem Fall sind wir fertig und erhalten
   $$n = p_1^{e_1} \dots p_{i-1}^{e_{i-1}} p_i.$$
   (b) Ist d_i der kleinste Teiler von n_i mit $p_{i-1} < d_i \leq \sqrt{n_i}$, so ist d_i eine Primzahl, also $d_i = p_i$. Wir teilen p_i so oft wie möglich aus n_i heraus und erhalten $n_i = p_i^{e_i} n_{i+1}$, so dass p_i die Zahl n_{i+1} nicht mehr teilt. Der kleinste Teiler von n_{i+1} ist nun größer als p_i. Wir haben nun
   $$n = p_1^{e_1} \dots p_i^{e_i} n_{i+1}.$$
   Wir beginnen mit n_{i+1} statt n_i von vorne.

Beispiel: Wir wollen $n = 100002$ faktorisieren.

- Wir setzen $n_1 = n = 100002$. Es ist $\sqrt{n_1} = 316.23\dots$. Wir suchen nach dem kleinsten nichttrivialen Teiler und finden $p_1 = 2$. Wir zerlegen
  $$n_1 = 2 \cdot 50001$$
  und setzen $n_2 = 50001$, wobei n_2 offensichtlich nicht durch 2 teilbar ist.
- Nun ist $n_2 = 50001$ und $\sqrt{n_2} = 223.60\dots$. Wir suchen nach dem kleinsten Teiler von n_2, der größer 2 sein muss, und finden $p_2 = 3$. Wir zerlegen
  $$n_2 = 3 \cdot 16667,$$
  setzen $n_3 = 16667$, wobei n_3 nicht mehr durch 3 teilbar ist.
- Nun ist $n_3 = 16667$ und $\sqrt{16667} = 129.10\dots$. Wir suchen nach dem kleinsten Teiler von n_3, der größer als 3 sein muss, und finden $p_3 = 7$. Wir zerlegen
  $$n_3 = 7 \cdot 2381,$$
  setzen $n_4 = 2381$, wobei n_4 nicht durch 7 teilbar ist.
- Nun ist $n_4 = 2381$ und $\sqrt{n_4} = 48.79\dots$. Wir suchen nach einem Teiler d mit $7 < d \leq 48$, finden aber nichts. Daher ist $n_4 = p_4 = 2381$ eine Primzahl und wir sind fertig.
- Insgesamt erhalten wir die Zerlegung
  $$100002 = 2 \cdot 3 \cdot 7 \cdot 2381.$$

Obige Überlegungen führen zu folgendem Faktorisierungsverfahren:

Naives Faktorisierungsverfahren:
Eingabe: Eine natürliche Zahl $n > 1$
Ausgabe: Alle Primfaktorpotenzen $p_i^{e_i}$ der Primfaktorzerlegung von n

```
1: if 2 | n then
2:     e ← 1, n ← n/2
3:     while 2 | n do
4:         e ← e + 1, n ← n/2
5:     end while
6:     Gib 2^e aus
```

```
    end if
    p ← 3
    while p^2 ≤ n do
        if p | n then
            e ← 1, n ← n/p
            while p | n do
                e ← e + 1, n ← n/p
            end while
            Gib p^e aus
        end if
        p ← p + 2
    end while
    if n > 1 then
        Gib n (als größten Primteiler von n) aus
    end if
```

Bemerkung: Die folgenden Rechnungen wurden mit einem Computer mit Prozessor Intel Cores i7-3770 CPU 3.40 GHz x 8 und Ubuntu als Betriebssystem gerechnet. Die Zeitangaben beziehen sich darauf.

Beispiele: In der Tabelle findet man die Faktorisierung der Zahlen $10^{20} + r$ mit $0 \leq r \leq 50$.

Bemerkungen:

(1) Die Laufzeit des naiven Faktorisierungsverfahrens lässt sich durch $O(\sqrt{n} \ln^2 n)$ abschätzen.
(2) Mit dem Verfahren kann man auch beweisen, dass eine Zahl n prim ist, wenn man nämlich keinen Teil $t \leq \sqrt{n}$ findet. Die Schrittzahl für einen solchen Primzahlbeweis wächst dann auch wie $\sqrt{n}$.

Bemerkung: Bei den folgenden Zahlen, die sich als Primzahlen p herausstellten, haben wir die benötigte Rechenzeit t und dann den Quotienten $\frac{t}{\sqrt{p}}$ angeschaut:

Primzahl p	Rechenzeit t	$\frac{t}{\sqrt{p}}$
$10^{13} + 37$	0.31 sec - 0.31 s	$9.78 \cdot 10^{-8}$ sec
$10^{14} + 31$	0.72 sec - 0.72 s	$7.18 \cdot 10^{-8}$ sec
$10^{15} + 37$	2.26 sec - 2.26 s	$7.15 \cdot 10^{-8}$ sec
$10^{16} + 61$	7.14 sec - 7.14 s	$7.14 \cdot 10^{-8}$ sec
$10^{17} + 3$	22.40 sec - 22.40 s	$7.08 \cdot 10^{-8}$ sec
$10^{18} + 3$	69.19 sec - 1 m 9.19 s	$6.92 \cdot 10^{-8}$ sec
$10^{19} + 51$	287.15 sec - 4 m 47.15 s	$9.08 \cdot 10^{-8}$ sec
$10^{20} + 39$	966.80 sec - 16 m 6.80 s	$9.67 \cdot 10^{-8}$ sec

Extrapoliert man dies, so braucht der Rechner mit unserem Programm mindestens $7 \cdot 10^{-8} \cdot \sqrt{p}$ Sekunden, um zu zeigen, dass p eine Primzahl ist. Wir haben damit die Laufzeit für einige Zahlen abgeschätzt:

$p \approx$	Laufzeit $\geq$
10^{20}	11.66 Minuten
10^{22}	1.94 Stunden
10^{24}	19.44 Stunden
10^{26}	8.10 Tage
10^{28}	81.02 Tage
10^{30}	810.19 Tage
10^{32}	22.20 Jahre
10^{34}	221.97 Jahre
10^{36}	2219.69 Jahre
10^{38}	22196.85 Jahre
10^{40}	221968.54 Jahre

n	Faktorzerlegung	Rechenzeit
$10^{20}+0$	$2^{20} \cdot 5^{20}$	0.00 sec - 0.00 s
$10^{20}+1$	$73 \cdot 137 \cdot 1676321 \cdot 5964848081$	0.19 sec - 0.19 s
$10^{20}+2$	$2 \cdot 3 \cdot 155977777 \cdot 106852828571$	11.55 sec - 11.55 s
$10^{20}+3$	$373 \cdot 155773 \cdot 1721071782307$	0.10 sec - 0.10 s
$10^{20}+4$	$2^2 \cdot 13 \cdot 1597 \cdot 240841 \cdot 4999900001$	0.02 sec - 0.02 s
$10^{20}+5$	$3 \cdot 5 \cdot 7^2 \cdot 83 \cdot 1663 \cdot 985694468327$	0.07 sec - 0.07 s
$10^{20}+6$	$2 \cdot 31 \cdot 6079 \cdot 265323774602147$	1.22 sec - 1.22 s
$10^{20}+7$	$67 \cdot 166909 \cdot 8942221889969$	0.22 sec - 0.22 s
$10^{20}+8$	$2^3 \cdot 3^3 \cdot 233 \cdot 1986965506278811$	3.23 sec - 3.23 s
$10^{20}+9$	$557 \cdot 72937 \cdot 2461483384901$	0.12 sec - 0.12 s
$10^{20}+10$	$2 \cdot 5 \cdot 11 \cdot 909090909090909091$	69.92 sec - 1 m 9.92 s
$10^{20}+11$	$3 \cdot 37 \cdot 467 \cdot 236993 \cdot 8140004071$	0.02 sec - 0.02 s
$10^{20}+12$	$2^2 \cdot 7 \cdot 3571428571428571429$	168.68 sec - 2 m 48.68 s
$10^{20}+13$	$17 \cdot 1579 \cdot 3917 \cdot 6673 \cdot 142526051$	0.00 sec - 0.00 s
$10^{20}+14$	$2 \cdot 3 \cdot 19 \cdot 877192982456140351$	67.90 sec - 1 m 7.90 s
$10^{20}+15$	$5 \cdot 21977 \cdot 910042316967739$	2.17 sec - 2.17 s
$10^{20}+16$	$2^4 \cdot 611441 \cdot 10221754838161$	0.23 sec - 0.23 s
$10^{20}+17$	$3^2 \cdot 13 \cdot 14071 \cdot 60742012273531$	0.58 sec - 0.58 s
$10^{20}+18$	$2 \cdot 953 \cdot 33961 \cdot 340267 \cdot 4540219$	0.02 sec - 0.02 s
$10^{20}+19$	$7 \cdot 14285714285714285717$	351.44 sec - 5 m 51.44 s
$10^{20}+20$	$2^2 \cdot 3 \cdot 5 \cdot 23 \cdot 643 \cdot 60689 \cdot 1856948927$	0.00 sec - 0.00 s
$10^{20}+21$	$11 \cdot 307 \cdot 29612081729345573$	12.77 sec - 12.77 s
$10^{20}+22$	$2 \cdot 29 \cdot 298723121 \cdot 5771692279$	21.99 sec - 21.99 s
$10^{20}+23$	$3 \cdot 71^2 \cdot 151 \cdot 223 \cdot 196372304837$	0.03 sec - 0.03 s
$10^{20}+24$	$2^3 \cdot 59 \cdot 97 \cdot 3319 \cdot 19763 \cdot 33298613$	0.00 sec - 0.00 s
$10^{20}+25$	$5^2 \cdot 12056437 \cdot 331772977373$	0.92 sec - 0.92 s
$10^{20}+26$	$2 \cdot 3^2 \cdot 7 \cdot 313 \cdot 2535625538820427$	3.76 sec - 3.76 s
$10^{20}+27$	$6547 \cdot 15274171376202841$	9.19 sec - 9.19 s
$10^{20}+28$	$2^2 \cdot 25000000000000000007$	473.13 sec - 7 m 53.13 s
$10^{20}+29$	$3 \cdot 47 \cdot 4592117 \cdot 154442898157$	0.32 sec - 0.32 s
$10^{20}+30$	$2 \cdot 5 \cdot 13 \cdot 17 \cdot 43 \cdot 18679 \cdot 56335953419$	0.02 sec - 0.02 s
$10^{20}+31$	$109 \cdot 2861 \cdot 320668015610119$	1.33 sec - 1.33 s
$10^{20}+32$	$2^5 \cdot 3 \cdot 11 \cdot 251 \cdot 881 \cdot 1667 \cdot 1811 \cdot 141851$	0.00 sec - 0.00 s
$10^{20}+33$	$7 \cdot 19 \cdot 157 \cdot 4789042670370193$	5.03 sec - 5.03 s
$10^{20}+34$	$2 \cdot 479 \cdot 56027093 \cdot 1863101011$	3.93 sec - 3.93 s
$10^{20}+35$	$3^3 \cdot 5 \cdot 257 \cdot 2882259691598213$	3.86 sec - 3.86 s
$10^{20}+36$	$2^2 \cdot 3288757 \cdot 7601656188037$	0.24 sec - 0.24 s
$10^{20}+37$	$31 \cdot 821 \cdot 59004541 \cdot 66590107$	4.30 sec - 4.30 s
$10^{20}+38$	$2 \cdot 3 \cdot 32839 \cdot 507526619771207$	1.66 sec - 1.66 s
$10^{20}+39$	100000000000000000039	969.55 sec - 16 m 9.55 s
$10^{20}+40$	$2^3 \cdot 5 \cdot 7 \cdot 41 \cdot 53 \cdot 164354743277891$	0.92 sec - 0.92 s
$10^{20}+41$	$3 \cdot 2939 \cdot 3313 \cdot 3423400606921$	0.14 sec - 0.14 s
$10^{20}+42$	$2 \cdot 3119 \cdot 97441 \cdot 164517801499$	0.03 sec - 0.03 s
$10^{20}+43$	$11 \cdot 13^2 \cdot 23 \cdot 107 \cdot 21857928274957$	0.34 sec - 0.34 s
$10^{20}+44$	$2^2 \cdot 3^2 \cdot 201450463 \cdot 13788887533$	14.80 sec - 14.80 s
$10^{20}+45$	$5 \cdot 7541 \cdot 52069 \cdot 50935645921$	0.02 sec - 0.02 s
$10^{20}+46$	$2 \cdot 288191 \cdot 173496049494953$	0.98 sec - 0.98 s
$10^{20}+47$	$3 \cdot 7 \cdot 17 \cdot 4289 \cdot 17321 \cdot 3770533259$	0.00 sec - 0.00 s
$10^{20}+48$	$2^4 \cdot 37 \cdot 61 \cdot 193 \cdot 199 \cdot 751 \cdot 96005947$	0.00 sec - 0.00 s
$10^{20}+49$	$229 \cdot 2017 \cdot 180221 \cdot 1201304833$	0.01 sec - 0.01 s
$10^{20}+50$	$2 \cdot 3 \cdot 5^2 \cdot 261382937 \cdot 2550536291$	19.04 sec - 19.04 s

Auch wenn es bedeutend schnellere Rechner gibt, so bedeutet eine Laufzeit, die (mindestens) wie $\sqrt{p}$ wächst: Zwei Dezimalstellen mehr verzehnfacht die Laufzeit. Man sieht, dass das Verfahren schnell nicht mehr praktikabel wird.

Sprechweise: Wir betrachten einen Algorithmus, bei dem Zahlen $\leq n$ eingegeben werden.

- Der Algorithmus heißt polynomial, wenn er eine Laufzeit von $O((\ln n)^\lambda)$ mit einer reellen Zahl $\lambda > 0$ hat.
- Der Algorithmus heißt exponentiell, wenn er eine Laufzeit von $O(n^\lambda)$ mit einer reellen Zahl $\lambda > 0$ hat.

So ist das naive Faktorisierungsverfahren mit einer Laufzeit von $O(n^{\frac{1}{2}+\varepsilon})$ exponentiell, Division mit Rest mit einer Laufzeit von $O((\ln n)^2)$ polynomial.

Mit der eindeutigen Primfaktorzerlegung ganzer Zahlen erhält man eine schöne Charakterisierung der Teilbarkeitsrelation:

SATZ. *Sind $a, b \in \mathbb{Z}$, $a, b \neq 0$, mit den Primfaktorzerlegungen*

$$a = \pm \prod_i p_i^{a_i} \quad \text{und} \quad b = \prod_i p_i^{b_i},$$

so gilt

$$a|b \quad \Longleftrightarrow \quad a_i \leq b_i \text{ für alle } i,$$

oder kurz geschrieben:

$$\pm \prod_i p_i^{a_i} | \pm \prod_i p_i^{b_i} \quad \Longleftrightarrow \quad a_i \leq b_i \text{ für alle } i.$$

Beweis: $\Rightarrow$ Wegen $a|b$ existiert $c \in \mathbb{Z}$, $c \neq 0$, mit $b = ac$. Sei $c = \pm \prod p_i^{c_i}$ die Primfaktorzerlegung von c. Dann folgt aus $b = ac$

$$\pm \prod p_i^{b_i} = \pm \prod p_i^{a_i} \cdot \prod p_i^{c_i} = \pm \prod p_i^{a_i + c_i}.$$

Die Eindeutigkeit der Primfaktorzerlegung liefert $b_i = a_i + c_i$, was wegen $c_i \geq 0$ die Behauptung $b_i \geq a_i$ zeigt.
$\Leftarrow$ Ist $a_i \leq b_i$, so ist $b_i - a_i \geq 0$, also $c = \prod p_i^{b_i - a_i}$ eine natürliche Zahl. Wie eben sieht man $b = \pm ac$ und damit $a|b$, was behauptet war. ■

6. Gemeinsame Teiler und ggT

Wir betrachten jetzt **gemeinsame Teiler** zweier ganzer Zahlen $m, n \in \mathbb{Z}$ und schreiben

$$\mathrm{gT}(m, n) = \{d \in \mathbb{Z} : d|m, d|n\}.$$

Beispiele:

$$\mathrm{gT}(20, 24) = \{\pm 1, \pm 2, \pm 4\}, \quad \mathrm{gT}(6, 0) = \{\pm 1, \pm 2, \pm 3, \pm 6\}, \quad \mathrm{gT}(6, 1) = \{\pm 1\}.$$

Mit den Teilbarkeitsregeln sieht man sofort, dass gilt

$$\mathrm{gT}(m, 0) = \{d \in \mathbb{Z} : d|m\} = \{\text{Menge der Teiler von } m\} \quad \text{und} \quad \mathrm{gT}(m, 1) = \{\pm 1\}.$$

Bei Kenntnis der Primfaktorzerlegung ist die Bestimmung der gemeinsamen Teiler mit dem obigen Lemma ganz einfach:

$$\mathrm{gT}(\pm \prod p_i^{a_i}, \pm \prod p_i^{b_i}) = \{\pm \prod p_i^{c_i} : 0 \leq c_i \leq \min(a_i, b_i)\}.$$

Außerdem:

$$\mathrm{gT}(\pm \prod p_i^{a_i}, 0) = \{\pm \prod p_i^{c_i} : c_i \leq a_i\} \quad \text{und} \quad \mathrm{gT}(0, 0) = \mathbb{Z}.$$

Die Menge $\mathrm{gT}(m,n)$ enthält ein (bzgl. Teilbarkeit) maximales Element, das wir o.E. ≥ 0 wählen, den sogenannten **größten gemeinsamen Teiler** $\mathrm{ggT}(m,n)$:

$$\begin{aligned}\mathrm{ggT}(\pm\prod p_i^{a_i}, \pm\prod p_i^{b_i}) &= \prod p_i^{\min(a_i,b_i)},\\ \mathrm{ggT}(\pm\prod p_i^{a_i}, 0) &= \prod p_i^{a_i},\\ \mathrm{ggT}(0,0) &= 0.\end{aligned}$$

Das folgende, nun leicht zu beweisende Lemma charakterisiert den ggT:

LEMMA. *Seien $m,n\in\mathbb{Z}$. Dann hat $d=\mathrm{ggT}(m,n)$ die folgenden Eigenschaften und ist dadurch eindeutig bestimmt:*

(1) $d\in\mathrm{gT}(m,n)$.
(2) $d'\in\mathrm{gT}(m,n)\Longrightarrow d'\mid d$.
(3) $d\geq 0$.

Bemerkungen:

(1) Durch die ersten beiden Eigenschaften charakterisiert man einen ggT in der Algebra.
(2) Die dritte Eigenschaft $d\geq 0$ dient nur der Normierung. Ohne diese wäre $\mathrm{ggT}(m,n)$ nur bis aufs Vorzeichen bestimmt.

Man nennt ganze Zahlen $m,n\in\mathbb{Z}$ **teilerfremd**, wenn $\mathrm{ggT}(m,n)=1$ gilt. Ist $m=\pm\prod p_i^{a_i}$, $n=\pm\prod p_i^{b_i}$, so gilt mit obiger Formel

$$\mathrm{ggT}(m,n)=1 \quad\Longleftrightarrow\quad a_i=0 \text{ oder } b_i=0,$$

d.h. die Menge der m teilenden Primzahlen und die Menge der n teilenden Primzahlen sind disjunkt.

Der folgende Satz gibt noch ein paar weitere nützliche Eigenschaften an:

SATZ. *Für ganze Zahlen $a,b,c\neq 0$ gilt:*

(1) $a\mid c, b\mid c$ *und* $\mathrm{ggT}(a,b)=1\Longrightarrow ab\mid c$.
(2) $a\mid bc$ *und* $\mathrm{ggT}(a,c)=1\Longrightarrow a\mid b$.
(3) *Ist* $d=\mathrm{ggT}(a,b)$, *schreibt man* $a=da'$, $b=db'$, *so gilt* $\mathrm{ggT}(a',b')=1$.

Analog zu gT definiert für $m,n\in\mathbb{Z}$ die Menge der gemeinsamen Vielfachen

$$\mathrm{gV}(m,n)=\{e\in\mathbb{Z} : m\mid e, n\mid e\}.$$

Für Zahlen $\neq 0$ ergibt sich die einfache Charakteriserung durch ihre Primfaktorzerlegung

$$\mathrm{gV}(\pm\prod p_i^{a_i}, \pm\prod p_i^{b_i}) = \{\pm\prod p_i^{c_i} : \max(a_i,b_i)\leq c_i\}\cup\{0\}.$$

Die Menge $\mathrm{gV}(m,n)$ enthält ein bzgl. Teilbarkeit kleinstes Element, das **kleinste gemeinsame Vielfache** $\mathrm{kgV}(m,n)$:

$$\begin{aligned}\mathrm{kgV}(\pm\prod p_i^{a_i}, \pm\prod p_i^{b_i}) &= \prod p_i^{\max(a_i,b_i)},\\ \mathrm{kgV}(\pm\prod p_i^{a_i}, 0) &= \mathrm{kgV}(0,0)=0.\end{aligned}$$

Für $m,n\in\mathbb{Z}$ gilt

$$\mathrm{kgV}(m,n)\cdot\mathrm{ggT}(m,n)=\pm mn.$$

7. Der euklidische Algorithmus

Kennt man die Primfaktorzerlegung der natürlichen Zahlen a und b, so kann man schnell $\mathrm{ggT}(a,b)$ berechnen. Es gibt aber auch einen anderen Weg, der ohne Primfaktorzerlegungen auskommt.

Sind $a,b,q,r \in \mathbb{Z}$ mit $a = qb + r$, so gilt für $d \in \mathbb{Z}$

$$d \in \mathrm{gT}(a,b) \iff d|a, d|b \iff d|a, d|b, d|r \iff d|b, d|r \iff d \in \mathrm{gT}(b,r),$$

also $gT(a,b) = gT(b,r)$ und damit auch $\mathrm{ggT}(a,b) = \mathrm{ggT}(b,r)$. Diese Eigenschaft führt zum euklidischen Algorithmus:

Satz (Euklidischer Algorithmus (Variante I)). *Seien $a,b \in \mathbb{N}_0$ gegeben. Rekursiv werden Zahlen $a_i \in \mathbb{N}_0$ definiert, wobei man mit $a_0 = a$ und $a_1 = b$ beginnt. Sind für einen Index $i \geq 0$ die Zahlen a_i und a_{i+1} bereits definiert, so unterscheidet man:*

- *Ist $a_{i+1} = 0$, so bricht man ab. (Es sei n der größte Index mit $a_{n+1} = 0$.)*
- *Ist $a_{i+1} > 0$, so dividiert man a_i durch a_{i+1} und erhält den Quotienten q_i und den Rest a_{i+2}. Dabei ist $a_{i+2} = a_i \bmod a_{i+1}$ und $0 \leq a_{i+2} < a_{i+1}$.*

Explizit ergibt sich das Schema (im Fall $a_1 > 0$):

$$\begin{aligned}
a_0 &= q_0 a_1 + a_2 \quad \textit{mit} \quad 0 < a_2 < a_1,\\
a_1 &= q_1 a_2 + a_3 \quad \textit{mit} \quad 0 < a_3 < a_2,\\
&\vdots\\
a_i &= q_i a_{i+1} + a_{i+2} \quad \textit{mit} \quad 0 < a_{i+2} < a_{i+1},\\
&\vdots\\
a_{n-2} &= q_{n-2} a_{n-1} + a_n \quad \textit{mit} \quad 0 < a_n < a_{n-1},\\
a_{n-1} &= q_{n-1} a_n + 0.
\end{aligned}$$

Dann gilt

$$\mathrm{ggT}(a,b) = a_n.$$

(Die Zahl n bezeichnen wir auch als Schrittzahl des euklidischen Algorithmus für die Zahlen a,b.)

Beweis: Aus $a_i = q_i a_{i+1} + a_{i+2}$ folgt mit der Vorbemerkung $\mathrm{ggT}(a_i, a_{i+1}) = \mathrm{ggT}(a_{i+1}, a_{i+2})$ und damit

$$\mathrm{ggT}(a_0, a_1) = \mathrm{ggT}(a_1, a_2) = \cdots = \mathrm{ggT}(a_{n-1}, a_n) = a_n,$$

wie behauptet. ■

Bemerkung: Wir haben damit also eine Möglichkeit gefunden, $\mathrm{ggT}(m,n)$ zu berechnen ohne die Primfaktorzerlegung von m und n zu kennen.

Beispiele:

(1) Wir wollen $\mathrm{ggT}(12345, 987)$ berechnen:

$$\begin{aligned}
12345 &= 12 \cdot 987 + 501,\\
987 &= 1 \cdot 501 + 486,\\
501 &= 1 \cdot 486 + 15,\\
486 &= 32 \cdot 15 + 6,\\
15 &= 2 \cdot 6 + 3,\\
6 &= 2 \cdot 3 + 0,
\end{aligned}$$

also gilt $\mathrm{ggT}(12345, 987) = 3$. Zum Vergleich: $12345 = 3 \cdot 5 \cdot 823$ und $987 = 3 \cdot 7 \cdot 47$.

(2) Was ist ggT(9264857236, 2453245253)? Mit 23 Divisonen ergibt sich

$$\begin{aligned}
9264857236 &= 3 \cdot 2453245253 + 1905121477 \\
2453245253 &= 1 \cdot 1905121477 + 548123776 \\
1905121477 &= 3 \cdot 548123776 + 260750149 \\
548123776 &= 2 \cdot 260750149 + 26623478 \\
260750149 &= 9 \cdot 26623478 + 21138847 \\
26623478 &= 1 \cdot 21138847 + 5484631 \\
21138847 &= 3 \cdot 5484631 + 4684954 \\
5484631 &= 1 \cdot 4684954 + 799677 \\
4684954 &= 5 \cdot 799677 + 686569 \\
799677 &= 1 \cdot 686569 + 113108 \\
686569 &= 6 \cdot 113108 + 7921 \\
113108 &= 14 \cdot 7921 + 2214 \\
7921 &= 3 \cdot 2214 + 1279 \\
2214 &= 1 \cdot 1279 + 935 \\
1279 &= 1 \cdot 935 + 344 \\
935 &= 2 \cdot 344 + 247 \\
344 &= 1 \cdot 247 + 97 \\
247 &= 2 \cdot 97 + 53 \\
97 &= 1 \cdot 53 + 44 \\
53 &= 1 \cdot 44 + 9 \\
44 &= 4 \cdot 9 + 8 \\
9 &= 1 \cdot 8 + 1 \\
8 &= 8 \cdot 1 + 0
\end{aligned}$$

Also ist der ggT 1.

(3) Für die 20 stelligen Zahlen 15847523452462634165 und 87648572364875263842 erhält man den ggT nach 40 Divisionen

$$\text{ggT}(15847523452462634165, 87648572364875263842) = 1$$

wie folgt:

$$\begin{aligned}
1584752345246263 4165 &= 0 \cdot 8764857236487526 3842 + 1584752345246263 4165\\
8764857236487526 3842 &= 5 \cdot 1584752345246263 4165 + 841095510256209 3017\\
1584752345246263 4165 &= 1 \cdot 841095510256209 3017 + 743656834990054 1148\\
841095510256209 3017 &= 1 \cdot 743656834990054 1148 + 974386752661551869\\
743656834990054 1148 &= 7 \cdot 974386752661551869 + 615861081269678065\\
974386752661551869 &= 1 \cdot 615861081269678065 + 358525671391873804\\
615861081269678065 &= 1 \cdot 358525671391873804 + 257335409877804261\\
358525671391873804 &= 1 \cdot 257335409877804261 + 101190261514069543\\
257335409877804261 &= 2 \cdot 101190261514069543 + 54954886849665175\\
101190261514069543 &= 1 \cdot 54954886849665175 + 46235374664404368\\
54954886849665175 &= 1 \cdot 46235374664404368 + 8719512185260807\\
46235374664404368 &= 5 \cdot 8719512185260807 + 2637813738100333\\
8719512185260807 &= 3 \cdot 2637813738100333 + 806070970959808\\
2637813738100333 &= 3 \cdot 806070970959808 + 219600825220909\\
806070970959808 &= 3 \cdot 219600825220909 + 147268495297081\\
219600825220909 &= 1 \cdot 147268495297081 + 72332329923828\\
147268495297081 &= 2 \cdot 72332329923828 + 2603835449425\\
72332329923828 &= 27 \cdot 2603835449425 + 2028772789353\\
2603835449425 &= 1 \cdot 2028772789353 + 575062660072\\
2028772789353 &= 3 \cdot 575062660072 + 303584809137\\
575062660072 &= 1 \cdot 303584809137 + 271477850935\\
303584809137 &= 1 \cdot 271477850935 + 32106958202\\
271477850935 &= 8 \cdot 32106958202 + 14622185319\\
32106958202 &= 2 \cdot 14622185319 + 2862587564\\
14622185319 &= 5 \cdot 2862587564 + 309247499\\
2862587564 &= 9 \cdot 309247499 + 79360073\\
309247499 &= 3 \cdot 79360073 + 71167280\\
79360073 &= 1 \cdot 71167280 + 8192793\\
71167280 &= 8 \cdot 8192793 + 5624936\\
8192793 &= 1 \cdot 5624936 + 2567857\\
5624936 &= 2 \cdot 2567857 + 489222\\
2567857 &= 5 \cdot 489222 + 121747\\
489222 &= 4 \cdot 121747 + 2234\\
121747 &= 54 \cdot 2234 + 1111\\
2234 &= 2 \cdot 1111 + 12\\
1111 &= 92 \cdot 12 + 7\\
12 &= 1 \cdot 7 + 5\\
7 &= 1 \cdot 5 + 2\\
5 &= 2 \cdot 2 + 1\\
2 &= 2 \cdot 1 + 0
\end{aligned}$$

(4) Mit dem euklidischen Algorithmus erhalten wir in 79 Schritten

$$\mathrm{ggT}(17^{60} + 20882693916, 13^{40} + 14609017703) = 25937424629.$$

(5) Für die (zufällig gewählten) 1000-stelligen Zahlen

$$a = 71^{540} + 92600321179110855935 \quad \text{und} \quad b = 83^{521} + 75133748210326629851$$

erhält man mit dem euklidischen Algorithmus nach 1978 Schritten $\mathrm{ggT}(a, b) = 2$.

Der ggT lässt sich mit dem euklidischen Algorithmus also sehr schnell berechnen. Daher wird dieser Algorithmus in der Praxis auch oft eingesetzt.

Für die ggT-Berechnung braucht man im euklidischen Algorithmus die Quotienten q_i nicht, sofern man $a_i \bmod a_{i+1}$ anders als durch $a_i - \left\lfloor \frac{a_i}{a_{i+1}} \right\rfloor a_{i+1}$ berechnen kann. Man erhält dann folgende Variante:

Satz (Euklidischer Algorithmus (Variante II)). *Seien $a, b \in \mathbb{N}_0$ gegeben. Rekursiv werden Zahlen $a_i \in \mathbb{N}_0$ definiert, wobei mit $a_0 = a$ und $a_1 = b$ begonnen wird. Sind für einen Index $i \geq 0$ die Zahlen a_i und a_{i+1} bereits definiert, so unterscheidet man:*

- *Ist $a_{i+1} = 0$, so bricht man ab. Es ist dann $\mathrm{ggT}(a, b) = a_i$.*
- *Ist $a_{i+1} > 0$, so definiert man $a_{i+2} = a_i \bmod a_{i+1}$.*

Explizit:

$$a_0, \quad a_1, \quad a_2 = a_0 \bmod a_1, \quad a_3 = a_1 \bmod a_2, \quad a_4 = a_2 \bmod a_3, \quad \dots \quad a_{n+1} = a_{n-1} \bmod a_n = 0,$$

dann ist

$$\mathrm{ggT}(a, b) = a_n.$$

Beispiel: $a = 12345$, $b = 987$ liefern folgende Zahlenfolge a_i:

$$12345, \quad 987, \quad 501, \quad 486, \quad 15, \quad 6, \quad 3, \quad 0.$$

Startet man mit $a = 987$, $b = 12345$, so erhält man die Folge

$$987, \quad 12345, \quad 987, \quad 501, \quad 486, \quad 15, \quad 6, \quad 3, \quad 0.$$

Hier ist eine zugehörige Python3-Funktion:

```
def ggT_II(a,b):
  a=[a,b]
  while a[-1]>0:
    a.append(a[-2]%a[-1])
  return a[-2]
```

Wir geben jetzt noch eine weitere Formulierung des euklidischen Algorithmus an, die sich dann einfach in einen Algorithmus umsetzen lässt.

Satz (Euklidischer Algorithmus (Variante III)). *Seien $a, b \in \mathbb{N}_0$ gegeben. Rekursiv werden Zahlen $a_i, b_i \in \mathbb{N}_0$ definiert, wobei mit $a_0 = a$ und $b_0 = b$ begonnen wird. Sind für einen Index $i \geq 0$ die Zahlen a_i und b_i bereits definiert, so unterscheidet man:*

- *Ist $b_i = 0$, so bricht man ab. Dann gilt $\mathrm{ggT}(a, b) = a_i$.*
- *Ist $b_i > 0$, so definiert man $a_{i+1} = b_i$ und $b_{i+1} = a_i \bmod b_i$.*

Beweis: Man erhält diese Darstellung sofort, wenn man in der letzten Variante des euklidischen Algorithmus b_i durch $b_i = a_{i+1}$ einführt. ■

Damit ergibt sich schnell folgender Algorithmus:

Euklidischer Algorithmus:
Eingabe: $a, b \in \mathbb{N}_0$
Ausgabe: $\mathrm{ggT}(a, b)$
1: **while** $b > 0$ **do**
2: $\quad a, b \leftarrow b, a \bmod b$
3: **end while**
4: **return** a

Eine mögliche Python3-Funktion sieht aus aus:

```
def ggT(a,b):
  while b>0:
    a,b=b,a%b
  return a
```

Man kann recht gut abschätzen, wie schnell der euklidische Algorithmus ist. Dazu brauchen wir ein paar Aussagen über Fibonacci-Zahlen.

8. Fibonacci-Zahlen

Die **Folge der Fibonacci-Zahlen** $(f_n)_{n\geq 0}$ wird rekursiv durch

$$f_0 = 0, \quad f_1 = 1 \quad \text{und} \quad f_n = f_{n-1} + f_{n-2} \text{ für } n \geq 2$$

definiert. Die Folge ist also

$$0, \quad 1, \quad 1, \quad 2, \quad 3, \quad 5, \quad 8, \quad 13, \quad 21, \quad 34, \quad 55, \quad 89, \quad 144, \quad 233, \quad 377, \quad 610, \quad 987, \quad \ldots$$

Bemerkung: Die folgende Python3-Funktion liefert eine Liste mit den ersten n Fibonacci-Zahlen:

```
def fibonacci_liste(n):
  f=[0,1]
  while len(f)<n:
    f.append(f[-1]+f[-2])
  return f
```

Das folgende Lemma gibt eine explizite Gestalt für f_n an, sowie eine Abschätzung für das Wachstumsverhalten, die später benötigt wird.

LEMMA. *Für die Fibonacci-Zahlen f_n gilt ($n \geq 0$):*

(1)
$$f_n = \frac{1}{\sqrt{5}}[(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n].$$

(2)
$$n \leq 4.785 \log_{10} f_{n+2}.$$

Beweis:

(1) Man kann die explizite Gestalt auch durch Induktion beweisen. Wir geben hier aber einen Ansatz an, der auf die explizite Gestalt führt.
Sei α eine reelle Zahl $\neq 0$. Wann erfüllt $g_n = \alpha^n$ die Rekursionsformel $g_n = g_{n-1} + g_{n-2}$? Genau dann, wenn $\alpha^2 = \alpha + 1$, d.h. wenn $\alpha = \frac{1\pm\sqrt{5}}{2}$ gilt. Wir setzen

$$\alpha = \frac{1+\sqrt{5}}{2}, \quad \beta = \frac{1-\sqrt{5}}{2}.$$

Für reelle Zahlen x, y erfüllt dann auch

$$h_n = x\alpha^n + y\beta^n$$

die Rekursionsformel $h_n = h_{n-1} + h_{n-2}$. Nun ist

$$h_0 = x + y, \quad h_1 = (x+y)\frac{1}{2} + (x-y)\frac{\sqrt{5}}{2}.$$

Wählen wir $y = -x$, so wird $h_0 = 0$ und $h_1 = x\sqrt{5}$. Wählen wir weiter $x = \frac{1}{\sqrt{5}}$, so wird auch noch $h_1 = 1$. Aus $h_0 = 0 = f_0$, $h_1 = 1 = f_1$ und der gleichen Rekursionsformel folgt nun sofort $f_n = h_n$ für alle $n \geq 0$, also

$$f_n = \frac{1}{\sqrt{5}}[(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n],$$

wie behauptet.

(2) Obige Formel für f_n liefert

$$\begin{aligned} f_{n+2} &= \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{n+2} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+2}\right] = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n+2}\left[1 - \left(\frac{1-\sqrt{5}}{1+\sqrt{5}}\right)^{n+2}\right] = \\ &= \left(\frac{1+\sqrt{5}}{2}\right)^{n} \cdot \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{2}\left[1 - \left(\frac{1-\sqrt{5}}{1+\sqrt{5}}\right)^{n+2}\right] = \\ &= \left(\frac{1+\sqrt{5}}{2}\right)^{n} \cdot \frac{5+3\sqrt{5}}{10}\left[1 - (-1)^n\left(\frac{3-\sqrt{5}}{2}\right)^{n+2}\right], \end{aligned}$$

sodass folgt

$$\begin{aligned} f_{n+2} \geq \left(\frac{1+\sqrt{5}}{2}\right)^{n} &\iff \frac{5+3\sqrt{5}}{10}\left[1 - (-1)^n\left(\frac{3-\sqrt{5}}{2}\right)^{n+2}\right] \geq 1 \iff \\ &\iff 1 - (-1)^n\left(\frac{3-\sqrt{5}}{2}\right)^{n+2} \geq \frac{-5+3\sqrt{5}}{2} \iff \\ &\iff \frac{7-3\sqrt{5}}{2} \geq (-1)^n\left(\frac{3-\sqrt{5}}{2}\right)^{n+2} = (-1)^n\left(\frac{3-\sqrt{5}}{2}\right)^{n} \cdot \frac{7-3\sqrt{5}}{2} \\ &\iff 1 \geq (-1)^n\left(\frac{3-\sqrt{5}}{2}\right)^{n} \approx (-1)^n \cdot 0.38^n. \end{aligned}$$

Die letzte Ungleichung ist für $n \geq 0$ richtig, also auch die erste, d.h. es gilt

$$f_{n+2} \geq \left(\frac{1+\sqrt{5}}{2}\right)^{n},$$

was sofort

$$n \leq \frac{1}{\log_{10}\frac{1+\sqrt{5}}{2}} \log_{10} f_{n+2} \leq 4.785 \log_{10} f_{n+2}$$

impliziert. ■

Bemerkung: Die folgenden Beispiele zeigen die Qualität der Abschätzung des letztes Lemmas:

n	0	1	2	3	4	5	6	10	1000	1001
$4.785 \log_{10} f_{n+2}$	0	1.44	2.28	3.34	4.32	5.33	6.33	10.33	1000.33	1001.33

Beispiel: Mit der Formel findet man

$$f_{480} \approx 0.92 \cdot 10^{100}, \quad f_{479} \approx 0.57 \cdot 10^{100},$$

d.h. f_{479} und f_{480} sind 100-stellige Dezimalzahlen.

Wir wenden jetzt die Aussagen über Fibonacci-Zahlen an um folgenden Satz zu beweisen:

Satz. *Um den ggT zweier natürlicher Zahl $a > b$ zu berechnen, braucht man mit dem euklidischen Algorithmus*

$$\leq 4.785 \log_{10} a$$

Divisionen mit Rest.

Beweis: Die Schrittzahl zur Berechnung des ggT sei n. Wir schreiben dann $a = g_{n+1}$, $b = g_n$ und

$$\begin{aligned} g_{n+1} &= q_{n+1}g_n + g_{n-1}, \quad 1 \le g_{n-1} < g_n \\ g_n &= q_n g_{n-1} + g_{n-2}, \quad 1 \le g_{n-2} < g_{n-2} \\ &\vdots \quad\quad \vdots \\ g_3 &= q_3 g_2 + g_1, \quad 1 \le g_1 < g_2 \\ g_2 &= q_2 g_1 + g_0 \text{ mit } g_0 = 0 \text{ und } g_1 = \mathrm{ggT}(a,b). \end{aligned}$$

Wir wollen die g_i's mit der Fibonacci-Folge $f_0 = 0$, $f_1 = 1$, $f_2 = 1$, $f_3 = 2, \ldots, f_i = f_{i-1}+f_{i-2}$ vergleichen. Wir zeigen zunächst $g_i \ge f_{i+1}$ für $i \ge 1$ durch Induktion: Für $i = 1$ ist $g_1 = \mathrm{ggT}(a,b) \ge 1 = f_2$. Für $i = 2$ gilt $g_2 > g_1 \ge 1$, also $g_2 \ge 2 = f_3$. Wir machen jetzt den Induktionsschluss für $i \ge 3$, wobei wir $q_i \ge 1$ benutzen:

$$g_i = q_i g_{i-1} + g_{i-2} \ge g_{i-1} + g_{i-2} \ge f_i + f_{i-1} = f_{i+1},$$

womit $g_i \ge f_{i+1}$ für $i \ge 1$ durch Induktion bewiesen wäre. Es folgt:

$$a = g_{n+1} \ge f_{n+2}$$

und daher mit obigem Lemma

$$n \le 4.785 \log_{10} f_{n+2} \le 4.785 \log_{10} a,$$

was die Behauptung zeigt. ■

Der ggT lässt sich mit dem euklidischen Algorithmus also sehr schnell berechnen. Daher wird dieser Algorithmus in der Praxis auch oft eingesetzt.

9. Der erweiterte euklidische Algorithmus

Der folgende Satz gibt eine wichtige Eigenschaft des ggT an, die sowohl theoretisch als auch praktisch oft benutzt wird:

SATZ. *Zu $a, b \in \mathbb{Z}$ gibt es $x, y \in \mathbb{Z}$ mit*

$$\mathrm{ggT}(a,b) = xa + yb.$$

Der nachfolgende Satz gibt einen konstruktiven Beweis der Aussage.

Bemerkungen:

(1) Der Satz wird manchmal auch als *Satz von Bézout* bezeichnet.

(2) Wendet man den euklidischen Algorithmus auf $a = a_0$ und $b = a_1$ an, so erhält man das folgende Schema:

$$\begin{aligned} a_0 &= q_0 a_1 + a_2 \quad \text{mit} \quad 0 < a_2 < a_1, \\ a_1 &= q_1 a_2 + a_3 \quad \text{mit} \quad 0 < a_3 < a_2, \\ &\vdots \\ a_i &= q_i a_{i+1} + a_{i+2} \quad \text{mit} \quad 0 < a_{i+2} < a_{i+1}, \\ &\vdots \\ a_{n-2} &= q_{n-2} a_{n-1} + a_n \quad \text{mit} \quad 0 < a_n < a_{n-1}, \\ a_{n-1} &= q_{n-1} a_n + 0 \quad \text{mit} \quad a_{n+1} = 0. \end{aligned}$$

Dann ist $a_n = \mathrm{ggT}(a,b)$. Man kann nun beginnend mit der vorletzten Zeile durch sukzessives Eliminieren a_n als Linearkombination von a_0 und a_1 schreiben, also $x, y \in \mathbb{Z}$ finden mit $a_n = xa_0 + ya_1$. Ist n klein, funktioniert dies auch praktisch.

Beispiel: Für 10 und 7 liefert der euklidische Algorithmus das Schema

$$\begin{aligned} 10 &= 1 \cdot 7 + 3, \\ 7 &= 2 \cdot 3 + 1, \\ 3 &= 3 \cdot 1 + 0. \end{aligned}$$

Beginnend mit der vorletzten Zeile folgt

$$1 = 7 - 2 \cdot 3 = 7 - 2 \cdot (10 - 1 \cdot 7) = -2 \cdot 10 + 3 \cdot 7.$$

Eine systematischere Vorgehensweise liefert der erweiterte euklidische Algorithmus.

SATZ (Erweiterter euklidischer Algorithmus (Variante I)). *Seien $a, b \in \mathbb{N}_0$. Man definiert rekursiv Zahlenfolgen q_i, a_i, x_i, y_i durch folgende Vorschrift:*

- $a_0 = a$, $a_1 = b$, $x_0 = 1$, $x_1 = 0$, $y_0 = 0$, $y_1 = 1$.
- *Sind $a_i, a_{i+1}, x_i, x_{i+1}, y_i, y_{i+1}$ bereits definiert, so unterscheidet man:*
  - *Ist $a_{i+1} = 0$, so endet die Konstruktion.*
  - *Ist $a_{i+1} > 0$, so definiert man*

$$q_i = \left\lfloor \frac{a_i}{a_{i+1}} \right\rfloor \quad \text{und} \quad a_{i+2} = a_i \bmod a_{i+1} \ (\text{oder } a_{i+2} = a_i - q_i a_{i+1})$$

und

$$x_{i+2} = x_i - q_i x_{i+1} \quad \text{und} \quad y_{i+2} = y_i - q_i y_{i+1}.$$

Ist $n \in \mathbb{N}_0$ der Index mit $a_{n+1} = 0$, so kann man die Vorgehensweise in folgender Tabelle zusammenfassen:

	a_0	$x_0 = 1$	$y_0 = 0$
	a_1	$x_1 = 0$	$y_1 = 1$
$q_0 = \left\lfloor \frac{a_0}{a_1} \right\rfloor$	$a_2 = a_0 - q_0 a_1$	$x_2 = x_0 - q_0 x_1$	$y_2 = y_0 - q_0 y_1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$*$	a_i	x_i	y_i
$*$	a_{i+1}	x_{i+1}	y_{i+1}
$q_i = \left\lfloor \frac{a_i}{a_{i+1}} \right\rfloor$	$a_{i+2} = a_i - q_i a_{i+1}$	$x_{i+2} = x_i - q_i x_{i+1}$	$y_{i+2} = y_i - q_i y_{i+1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$*$	a_{n-1}	x_{n-1}	y_{n-1}
$*$	a_n	x_n	y_n
$q_{n-1} = \left\lfloor \frac{a_{n-1}}{a_n} \right\rfloor = \frac{a_{n-1}}{a_n}$	$a_{n+1} = 0$	x_{n+1}	y_{n+1}

Dann gilt

$$a_i = x_i a + y_i b \text{ für } 0 \leq i \leq n+1,$$

und insbesondere

$$\mathrm{ggT}(a, b) = a_n \quad \text{und} \quad \mathrm{ggT}(a, b) = x_n a + y_n b.$$

Beweis:

(1) Die Aussage $\mathrm{ggT}(a, b) = a_n$ haben wir schon beim euklidischen Algorithmus gezeigt.

(2) Wir denken uns die Zeilen der Tabelle von 0 bis $n+1$ nummeriert. Wir zeigen durch Induktion, dass

$$a_i = x_i a + y_i b \text{ für } i = 0, \ldots, n+1$$

gilt. Für $i = 0$ und $i = 1$ folgt dies aus der Definition von x_0, y_0, x_1, y_1. Ist nun $i \geq 0$ und die Aussage bereits für i und $i+1$ gezeigt, also

$$\begin{aligned} a_i &= x_i a + y_i b, \\ a_{i+1} &= x_{i+1} a + y_{i+1} b, \end{aligned}$$

so folgt sofort

$$\begin{aligned} x_{i+2}a + y_{i+2}b &= (x_i - q_i x_{i+1})a + (y_i - q_i y_{i+1})b = \\ &= (ax_i + y_i b) - q_i(x_{i+1}a + y_{i+1}b) = a_i - q_i a_{i+1} = a_{i+2}, \end{aligned}$$

also die Behauptung. ■

Bemerkung: Das im Satz angegebene Verfahren lässt sich auch per Hand gut ausführen: Seien $a, b \in \mathbb{N}_0$ gegeben.

- START: Lege eine Tabelle mit 4 Spalten (für q-Werte und a_i, x_i, y_i): In die ersten beiden Zeilen trägt man folgende Werte ein, wobei die q-Spalte noch frei bleibt:

q	a_i	x_i	y_i
	a	1	0
	b	0	1

- WIEDERHOLE: Hat man die Zeilen i und $i+1$ bereits bestimmt, wobei die Zählung mit 0 begonnen wird, und ist $a_{i+1} \neq 0$, so erhält man die Zeile $i+2$ wie folgt:

$\vdots$	$\vdots$	$\vdots$	$\vdots$
$*$	a_i	x_i	y_i
$*$	a_{i+1}	x_{i+1}	y_{i+1}
$q_i = \left\lfloor \frac{a_i}{a_{i+1}} \right\rfloor$	$a_{i+2} = a_i - q_i a_{i+1}$	$x_{i+2} = x_i - q_i x_{i+1}$	$y_{i+2} = y_i - q_i y_{i+1}$

  Man berechnet also zunächst den abgerundeten Quotienten $q_i = \left\lfloor \frac{a_i}{a_{i+1}} \right\rfloor$ und dann damit $a_{i+2}, x_{i+2}, y_{i+2}$. (Man beachte, dass $a_{i+2} = a_i - q_i a_{i+1} = a_i \bmod a_{i+1}$ ist.)
- ENDE: Falls $a_{i+1} = 0$ ist, so ist man in folgender Situation:

$\vdots$	$\vdots$	$\vdots$	$\vdots$
q_{i-2}	a_i	x_i	y_i
q_{i-1}	$a_{i+1} = 0$	x_{i+1}	y_{i+1}

  In diesem Fall ist man fertig: $a_i = x_i a + y_i b$ und $a_i = \mathrm{ggT}(a, b)$. (Die Werte x_{i+1} und y_{i+1} muss man nicht mehr berechnen, da sie nicht benötigt werden.)

Beispiele:

(1) $a = 14$, $b = 11$

q	a_i	x_i	y_i
	14	1	0
	11	0	1
1	3	1	-1
3	2	-3	4
1	1	4	-5
2	0	-11	14

Die Zahlen der vorletzten Zeile liefern

$$1 = \mathrm{ggT}(a, b) = 4 \cdot a - 5 \cdot b$$

(2) $a = 91$, $b = 83$

q	a_i	x_i	y_i
	91	1	0
	83	0	1
1	8	1	-1
10	3	-10	11
2	2	21	-23
1	1	-31	34
2	0	83	-91

Die Zahlen der vorletzten Zeile liefern

$$1 = \mathrm{ggT}(a, b) = -31 \cdot a + 34 \cdot b$$

(3) $a = 12345$, $b = 987$

q	a_i	x_i	y_i
	12345	1	0
	987	0	1
12	501	1	-12
1	486	-1	13
1	15	2	-25
32	6	-65	813
2	3	132	-1651
2	0	-329	4115

Die Zahlen der vorletzten Zeile liefern

$$3 = \mathrm{ggT}(a, b) = 132 \cdot a - 1651 \cdot b$$

(4) $a = 8462$, $b = 3876$

q	a_i	x_i	y_i
	8462	1	0
	3876	0	1
2	710	1	-2
5	326	-5	11
2	58	11	-24
5	36	-60	131
1	22	71	-155
1	14	-131	286
1	8	202	-441
1	6	-333	727
1	2	535	-1168
3	0	-1938	4231

Die Zahlen der vorletzten Zeile liefern

$$2 = \mathrm{ggT}(a, b) = 535 \cdot a - 1168 \cdot b$$

Nun tauschen wir die Zahlen: $a = 3876$, $b = 8462$

q	a_i	x_i	y_i
	3876	1	0
	8462	0	1
0	3876	1	0
2	710	-2	1
5	326	11	-5
2	58	-24	11
5	36	131	-60
1	22	-155	71
1	14	286	-131
1	8	-441	202
1	6	727	-333
1	2	-1168	535
3	0	4231	-1938

Die Zahlen der vorletzten Zeile liefern

$$2 = \mathrm{ggT}(a, b) = -1168 \cdot a + 535 \cdot b$$

(5) $a = 1234567$, $b = 7654321$

q	a_i	x_i	y_i
	1234567	1	0
	7654321	0	1
0	1234567	1	0
6	246919	-6	1
4	246891	25	-4
1	28	-31	5
8817	15	273352	-44089
1	13	-273383	44094
1	2	546735	-88183
6	1	-3553793	573192
2	0	7654321	-1234567

Die Zahlen der vorletzten Zeile liefern

$$1 = \mathrm{ggT}(a, b) = -3553793 \cdot a + 573192 \cdot b$$

(6) $a = 987654321$, $b = 123456789$

q	a_i	x_i	y_i
	987654321	1	0
	123456789	0	1
8	9	1	-8
13717421	0	-13717421	109739369

Die Zahlen der vorletzten Zeile liefern

$$9 = \mathrm{ggT}(a, b) = 1 \cdot a - 8 \cdot b$$

(7) $a = 9876543211$, $b = 1234567891$

q	a_i	x_i	y_i
	9876543211	1	0
	1234567891	0	1
8	83	1	-8
14874311	78	-14874311	118994489
1	5	14874312	-118994497
15	3	-237988991	1903911944
1	2	252863303	-2022906441
1	1	-490852294	3926818385
2	0	1234567891	-9876543211

Die Zahlen der vorletzten Zeile liefern

$$1 = \mathrm{ggT}(a, b) = -490852294 \cdot a + 3926818385 \cdot b$$

(8) Das Verfahren funktioniert auch, wenn $a =$ oder $b = 0$ gilt:
$a = 2$, $b = 0$

q	a_i	x_i	y_i
	2	1	0
	0	0	1

Die Zahlen der vorletzten Zeile liefern

$$2 = \mathrm{ggT}(a, b) = 1 \cdot a + 0 \cdot b$$

Für $a = 0$, $b = 2$ erhält man

q	a_i	x_i	y_i
	0	1	0
	2	0	1
0	0	1	0

Die Zahlen der vorletzten Zeile liefern

$$2 = \mathrm{ggT}(a, b) = 0 \cdot a + 1 \cdot b$$

Im Fall $a = 0$, $b = 0$ ergibt sich

q	a_i	x_i	y_i
	0	1	0
	0	0	1

Die Zahlen der vorletzten Zeile liefern

$$0 = \mathrm{ggT}(a, b) = 1 \cdot a + 0 \cdot b$$

(9) $a = 15847523452462634165$, $b = 87648572364875263842$

q	a_i	x_i	y_i
	15847523452462634165	1	0
	87648572364875263842	0	1
0	15847523452462634165	1	0
5	8410955102562093017	-5	1
1	7436568349900541148	6	-1
1	974386752661551869	-11	2
7	615861081269678065	83	-15
1	358525671391873804	-94	17
1	257335409877804261	177	-32
1	101190261514069543	-271	49
2	54954886849665175	719	-130
1	46235374664404368	-990	179
1	8719512185260807	1709	-309
5	2637813738100333	-9535	1724
3	806070970959808	30314	-5481
3	219600825220909	-100477	18167
3	147268495297081	331745	-59982
1	72332329923828	-432222	78149
2	2603835449425	1196189	-216280
27	2028772789353	-32729325	5917709
1	575062660072	33925514	-6133989
3	303584809137	-134505867	24319676
1	271477850935	168431381	-30453665
1	32106958202	-302937248	54773341
8	14622185319	2591929365	-468640393
2	2862587564	-5486795978	992054127
5	309247499	30025909255	-5428911028
9	79360073	-275719979273	49852253379
3	71167280	857185847074	-154985671165
1	8192793	-1132905826347	204837924544
8	5624936	9920432457850	-1793689067517
1	2567857	-11053338284197	1998526992061
2	489222	32027109026244	-5790743051639
5	121747	-171188883415417	30952242250256
4	2234	716782642687912	-129599712052663
54	1111	-38877451588562665	7029336693094058
2	12	78471685819813242	-14188273098240779
92	7	-7258272547011380929	1312350461731245726
1	5	7336744232831194171	-1326538734829486505
1	2	-14595016779842575100	2638889196560732231
2	1	36526777792516344371	-6604317127950950967
2	0	-87648572364875263842	15847523452462634165

Die Zahlen der vorletzten Zeile liefern

$$1 = \mathrm{ggT}(a, b) = 36526777792516344371 \cdot a - 6604317127950950967 \cdot b$$

Bemerkung: Bei der vorangegangenen Methode berechnet man bei Kenntnis von a_i, x_i, y_i und $a_{i+1}, x_{i+1}, y_{i+1}$ die neuen Zahlen $a_{i+2}, x_{i+2}, y_{i+2}$ mittels

$$a_{i+2} = a_i - q a_{i+1}, \quad x_{i+2} = x_i - q x_{i+1}, \quad y_{i+2} = y_i - q y_{i+1} \quad \text{mit} \quad q = \left\lfloor \frac{a_i}{a_{i+2}} \right\rfloor .$$

Um laufende Indizes im Algorithmus zu vermeiden, verwenden wir die Python-übliche Bezeichnung c_{-1} für das letzte, c_{-2} für das vorletzte Element der Folge/Liste c. Dann lässt sich unser Satz wie folgt umsetzen:

Erweiterter euklidischer Algorithmus (Variante I):

Eingabe: Zahlen $\widetilde{a}, \widetilde{b} \in \mathbb{N}_0$
Ausgabe: $\mathrm{ggT}(\widetilde{a}, \widetilde{b})$ und $x, y \in \mathbb{Z}$ mit $x\widetilde{a} + y\widetilde{b} = \mathrm{ggT}(\widetilde{a}, \widetilde{b})$.

1: Lege drei Listen/Folgen a, x, y an mit folgenden Startwerten: $a = [\widetilde{a}, \widetilde{b}]$, $x = [1, 0]$, $y = [0, 1]$.
2: **while** $a_{-1} > 0$ **do**
3: $\quad q \leftarrow \left\lfloor \frac{a_{-2}}{a_{-1}} \right\rfloor$
4: $\quad$ Hänge an a die Zahl $a_{-2} - qa_{-1}$ an.
5: $\quad$ Hänge an x die Zahl $x_{-2} - qx_{-1}$ an.
6: $\quad$ Hänge an y die Zahl $y_{-2} - qy_{-1}$ an.
7: **end while**
8: **return** Gib die Zahlen $a_{-2} = \mathrm{ggT}(\widetilde{a}, \widetilde{b})$ und x_{-2}, y_{-2} mit $\mathrm{ggT}(\widetilde{a}, \widetilde{b}) = x_{-2}\widetilde{a} + y_{-2}\widetilde{b}$ aus.

Eine zugehörige Python3-Funktion könnte so aussehen:

```
def eea(a,b):
    q=[]
    a=[a,b]
    x=[1,0]
    y=[0,1]
    while a[-1]>0:
        q=a[-2]//a[-1]
        a.append(a[-2]-q*a[-1])
        x.append(x[-2]-q*x[-1])
        y.append(y[-2]-q*y[-1])
    return a[-2],x[-2],y[-2]
```

Ein (kleiner) Nachteil der vorangegangenen Variante des erweiterten euklidischen Algorithmus ist, dass man sich die die Folgen a_i, x_i, y_i merkt. Für eine Programmiervariante ist es bequem, zusätzliche Größen b_i, x'_i, y'_i durch die Definition

$$b_i = a_{i+1}, \quad x'_i = x_{i+1}, \quad y'_i = y_{i+1}$$

einzuführen. Man erhält dann folgende Version:

Satz (Erweiterter euklidischer Algorithmus (Variante II)). *Seien $a, b \in \mathbb{N}_0$ gegeben. Man definiert rekursiv Zahlenfolgen $a_i, b_i, x_i, x'_i, y_i, y'_i$ wie folgt:*

- $a_0 = a$, $b_0 = b$, $x_0 = 1$, $x'_0 = 0$, $y_0 = 0$, $y'_0 = 1$.
- *Sind $a_i, b_i, x_i, x'_i, y_i, y'_i$ bereits definiert, so unterscheidet man:*
  - *Ist $b_i = 0$, so endet die Konstruktion.*
  - *Ist $b_i > 0$, so definiert man*

$$q_i = \left\lfloor \frac{a_i}{b_i} \right\rfloor$$

und damit

$$\begin{aligned} a_{i+1} &= b_i, \\ b_{i+1} &= a_i - q_i b_i = a_i \bmod b_i, \\ x_{i+1} &= x'_i, \\ x'_{i+1} &= x_i - q_i x'_i, \\ y_{i+1} &= y'_i, \\ y'_{i+1} &= y_i - q_i y'_i. \end{aligned}$$

Ist $n \geq 0$ der Index mit $b_n = 0$, so gilt

$$\mathrm{ggT}(a,b) = a_n \quad \textit{und} \quad \mathrm{ggT}(a,b) = x_n a + y_n b.$$

Erweiterter euklidischer Algorithmus (Variante II):
Eingabe: $a, b \in \mathbb{N}_0$
Ausgabe: $\mathrm{ggT}(a,b)$ und $x, y \in \mathbb{Z}$ mit $\mathrm{ggT}(a,b) = xa + yb$

```
1: x ← 1, x' ← 0, y ← 0, y' ← 1
2: while b > 0 do
3:     q ← ⌊a/b⌋
4:     a, b ← b, a − qb
5:     x, x' ← x', x − qx'
6:     y, y' ← y', y − qy'
7: end while
8: return a, x, y
```

Eine zugehörige Python3-Funktion könnte so aussehen:

```
def eea(a,b):
    x,y=1,0
    xx,yy=0,1
    while b>0:
        q=a//b
        a,b=b,a-q*b # Alternativ: a,b=b,a%b
        x,xx=xx,x-q*xx
        y,yy=yy,y-q*yy
    return a,x,y
```

Im Folgenden werden wir noch einige Anwendungen des erweiterten euklidischen Algorithmus geben.

10. Exkurs: Die Gleichung $ax + by = c$

Wir wissen, dass die Gleichung $\mathrm{ggT}(a,b) = ax + by$ lösbar ist. Der folgende Satz verallgemeinert diese Gleichung und gibt die Gesamtheit der Lösungen an:

SATZ. *Seien $a, b \in \mathbb{Z}$, $(a,b) \neq (0,0)$. Dann gilt:*

(1) *Die Gleichung $ax + by = c$ ist genau dann lösbar, wenn $\mathrm{ggT}(a,b)|c$ gilt.*
(2) *Gilt $\mathrm{ggT}(a,b)|c$ und sind $u, v \in \mathbb{Z}$ mit $au + bv = \mathrm{ggT}(a,b)$, so ist*

$$\{(x,y) \in \mathbb{Z}\times\mathbb{Z} : ax+by = c\} = \{(\frac{c}{\mathrm{ggT}(a,b)}u + \frac{b}{\mathrm{ggT}(a,b)}m, \frac{c}{\mathrm{ggT}(a,b)}v - \frac{a}{\mathrm{ggT}(a,b)}m) : m \in \mathbb{Z}\}.$$

Beweis: Sei $d = \mathrm{ggT}(a,b)$, $a = da'$, $b = db'$ und $au + bv = d$.

(1) Sind $x, y \in \mathbb{Z}$ mit $c = ax + by$, so folgt $c = d(a'x + b'y)$, was sofort die Behauptung $d|c$ ergibt. Sei nun umgekehrt $d|c$, also $c = de$. Dann ist $a(ue) + b(ve) = de = c$, die Gleichung $ax + by = c$ also lösbar. Anders geschrieben:
$$(\frac{c}{\mathrm{ggT}(a,b)}u, \frac{c}{\mathrm{ggT}(a,b)}v)$$
löst die Gleichung.
(2) Man sieht leicht, dass $\supseteq$ gilt. Sei jetzt $ax + by = c$. Dann ist $ax + by = a(ue) + b(ve)$, also $a(x - ue) = b(ve - y)$ bzw. $a'(x - ue) = b'(ve - y)$. Wegen $b'|a'(x - ue)$ und $\mathrm{ggT}(a', b') = 1$ folgt $b'|x - ue$, d.h. es existiert $m \in \mathbb{Z}$ mit $x - ue = mb'$. Einsetzen liefert $ve - y = ma'$. Damit folgt $\subseteq$. ■

FOLGERUNG. *Sind $a, b \in \mathbb{Z}$ mit $\mathrm{ggT}(a,b) = 1$, so gibt es $x_0, y_0 \in \mathbb{Z}$ mit $ax_0 + by_0 = 1$ und es gilt*

$$\{(x,y) \in \mathbb{Z}\times\mathbb{Z} : ax + by = 1\} = \{(x_0 + bm, y_0 - am) : m \in \mathbb{Z}\}.$$

11. Kongruenzrechnung

Bei der Division von $a \in \mathbb{Z}$ durch $b \in \mathbb{N}$ haben wir den Rest mit $(a \bmod b)$ bezeichnet, sodass gilt

$$a = \left\lfloor \frac{a}{b} \right\rfloor \cdot b + (a \bmod b).$$

Nun gibt es eine weitere Bedeutung von „mod“.

DEFINITION. *Für $a, b, m \in \mathbb{Z}$ sagt man, a ist* **kongruent** b **modulo** m*, in Zeichen $a \equiv b \bmod m$, falls $m \mid a - b$ gilt.*

Beispiele:

(1) $n \equiv 0 \bmod 2$, falls n gerade, $n \equiv 1 \bmod 2$, falls n ungerade.
(2) Gibt a durch m geteilt Rest r, d.h. $a = qm + r$, so ist $a \equiv r \bmod m$.
(3) Genau dann gilt $a \equiv 0 \bmod m$, wenn $m|a$ gilt.
(4) Genau dann gilt $a \equiv b \bmod 0$, wenn $a = b$ ist.
(5) Es ist
$$a \equiv b \bmod m \quad \Longleftrightarrow \quad a \equiv b \bmod (-m),$$
weswegen man sich oft auf $m \geq 0$ beschränkt.

Einige Grundaussagen über Kongruenzen sind in folgendem Satz zusammengestellt:

SATZ. *Sei $m \geq 1$.*

(1) *Durch $a \equiv b \bmod m$ wird eine Äquivalenzrelation definiert, die Äquivalenzklasse von a wird auch mit $\overline{a}$ bezeichnet (bei festem m), also: $a \equiv b \bmod m \iff \overline{a} = \overline{b}$. Die Menge der Äquivalenzklassen wird mit $\mathbb{Z}/m\mathbb{Z}$ bezeichnet.*
(2) *Ergibt a bei Division durch m den Rest r (mit $0 \leq r < m$), so ist $a \equiv r \bmod m$ und r der einzige Repräsentant der Äquivalenzklasse $\overline{a}$ zwischen 0 und $m-1$, d.h.*
$$\{r\} = \overline{a} \cap \{0, 1, 2, \ldots, m-1\}.$$
Diesen Rest haben wir zuvor mit $r = (a \bmod m)$ bezeichnet.
(3) *Für $a, b \in \mathbb{Z}$ gilt die Charakterisierung*
$$a \equiv b \bmod m \quad \Longleftrightarrow \quad (a \bmod m) = (b \bmod m).$$
(4) *Als Repräsentanten der Äquivalenzklassen kann man $0, 1, 2, \ldots, m-1$ wählen, d.h.*
$$\mathbb{Z}/m\mathbb{Z} = \{\overline{0}, \overline{1}, \overline{2}, \ldots, \overline{m-1}\}$$
und damit $\#\mathbb{Z}/m\mathbb{Z} = m$.
(5) *Die Äquivalenzrelation $a \equiv b \bmod m$ ist mit Addition und Multiplikation verträglich, d.h. aus $a_1 \equiv a_2 \bmod m$ und $b_1 \equiv b_2 \bmod m$ folgt*
$$a_1 + b_1 \equiv a_2 + b_2 \bmod m \quad \text{und} \quad a_1 b_1 \equiv a_2 b_2 \bmod m.$$
Durch die Definition
$$\overline{a} + \overline{b} = \overline{a+b} \quad \text{und} \quad \overline{a}\,\overline{b} = \overline{ab}$$
wird daher $\mathbb{Z}/m\mathbb{Z}$ zu einem kommutativen Ring mit Null $\overline{0}$ und Eins $\overline{1}$.

Beweis:

(1) Einfache Übung.
(2) Aus $a = qm + r$ folgt sofort $a \equiv r \bmod m$ und damit auch $r \in \overline{a} \cap \{0, 1, \ldots, m-1\}$. Ist nun $s \in \overline{a} \cap \{0, 1, \ldots, m-1\}$, so folgt $m \mid r - s$ und $|r - s| \leq m - 1$, also $r = s$, was die Eindeutigkeit von r beweist.
(3) Dies folgt aus (2).
(4) Dies ist nach (2) sofort klar.
(5) Einfache Übung. ■

In der Zahlentheorie wird sehr oft mit Kongruenzen gerechnet.

Beispiele:

(1) Wie sieht man einer Dezimalzahl an, ob sie durch 3 teilbar ist? Die Dezimalschreibweise $n = (n_d n_{d-1} \dots n_1 n_0)_{10}$ bedeutet
$$n = n_0 + n_1 \cdot 10 + n_2 \cdot 100 + \dots + n_d \cdot 10^d.$$
Nun ist $10 \equiv 1 \bmod 3$ und damit auch $10^i \equiv 1 \bmod 3$, d.h.
$$n \equiv n_0 + n_1 + \dots + n_d \bmod 3.$$
Also ist n genau dann durch 3 teilbar, wenn $n \equiv n_0 + \dots + n_d \equiv 0 \bmod 3$ gilt, d.h. wenn die Quersumme $n_0 + \dots + n_d$ durch 3 teilbar ist. (Auf gleiche Weise zeigt man die bekannten Teilbarkeitsregeln für 9 und 11.)

(2) Sei $e = (111 \dots 111)_{10}$ die Zahl mit 100 Einsen in der Dezimaldarstellung. Was ist die letzte Ziffer in der Dezimaldarstellung von 3^e? Uns interessiert also $3^e \bmod 10$. Es gilt
$$3^1 \equiv 3 \bmod 10, \quad 3^2 \equiv 9 \bmod 10, \quad 3^3 \equiv 7 \bmod 10, \quad 3^4 \equiv 1 \bmod 10.$$
Wir schreiben $e = 4q + r$, dann ist nämlich $3^e = 3^{4q+r} = 81^q \cdot 3^r \equiv 3^r \bmod 10$. Nun ist
$$r \equiv e = 100 \cdot \text{Zahl} + 11 \equiv 3 \bmod 4,$$
also $r = 3$ und damit
$$3^e \equiv 3^r = 3^3 \equiv 7 \bmod 10.$$
Die letzte Ziffer von 3^e ist also 7.

(3) Durch Ausprobieren findet man, dass für $x, y \in \mathbb{Z}$ gilt $x^2 + y^2 \equiv 0, 1, 2 \bmod 4$. Ist also n eine natürliche Zahl mit $n \equiv 3 \bmod 4$, so hat die (diophantische) Gleichung $x^2 + y^2 = n$ keine ganzzahligen Lösungen x und y.

Das folgende Lemma ist oft hilfreich.

LEMMA. *Seien $a, b \in \mathbb{Z}$ und $d, m, n \in \mathbb{N}$.*

(1) *Gilt $a \equiv b \bmod m$ und $d|m$, so gilt auch $a \equiv b \bmod d$.*
(2) *Gilt $a \equiv b \bmod m$, $a \equiv b \bmod n$ und $\mathrm{ggT}(m, n) = 1$, so folgt $a \equiv b \bmod mn$.*

Beweis:

(1) $a \equiv b \bmod m$ bedeutet $m|(a - b)$, so daß mit $d|m$ sofort $d|(a - b)$ und damit $a \equiv b \bmod d$ folgt.
(2) Die Kongruenzen bedeuten $m|(a - b)$ und $n|(a - b)$. Wegen $\mathrm{ggT}(m, n) = 1$ folgt $mn|(a - b)$ und damit die Behauptung. ■

12. Invertierbarkeit modulo n und die Eulersche φ-Funktion

Für $n \in \mathbb{N}$ und $a \in \mathbb{Z}$ sagt man, a **ist invertierbar modulo** n, falls ein $b \in \mathbb{Z}$ existiert mit $ab \equiv 1 \bmod n$. In diesem Fall nennt man b ein **Inverses von** a **modulo** n.

SATZ. *Sei $n \in \mathbb{N}$ und $a \in \mathbb{Z}$.*

(1) *a ist genau dann invertierbar modulo n, wenn $\mathrm{ggT}(n, a) = 1$ ist.*
(2) *Ist $\mathrm{ggT}(n, a) = 1$ so findet man mit dem erweiterten euklidischen Algorithmus $x, y \in \mathbb{Z}$ mit $xn + ya = 1$. Dann gilt $ya \equiv 1 \bmod n$, d.h. y ist invers zu a modulo n.*
(3) *Ist a invertierbar modulo n, sind b, b' invers zu a modulo n, so gilt $b \equiv b' \bmod n$, d.h. modulo n ist das Inverse von a eindeutig bestimmt.*

Beweis:

(1) Ist a invertierbar modulo n, so gibt es ein $b \in \mathbb{Z}$ mit $ab \equiv 1 \bmod n$, also $n \mid ab - 1$, es gibt ein $k \in \mathbb{Z}$ mit $kn = ab - 1$, d.h. $-kn + ba = 1$, woraus sofort $\mathrm{ggT}(n, a) = 1$ folgt. Die Umkehrung folgt sofort aus (2).
(2) Aus $xn + ya = 1$ folgt sofort $ya \equiv 1 \bmod n$, d.h. y ist invers zu a modulo n.

(3) Aus $ab \equiv 1 \bmod n$ und $ab' \equiv 1 \bmod n$ folgt $ab \equiv ab' \bmod n$, also $n \mid a(b-b')$. Wegen $\mathrm{ggT}(n,a) = 1$ gilt $n \mid b - b'$, also $b \equiv b' \bmod n$, was zu zeigen war. ■

Bemerkung: Gilt $ab \equiv 1 \bmod m$, so findet man auch die Schreibweise $b \equiv \frac{1}{a} \bmod m$ oder $b \equiv a^{-1} \bmod m$.

Wir formulieren die Aussage (2) des Satzes nochmals explizit:

Inversenberechnung von a modulo n:
Eingabe: $n \in \mathbb{N}$ und $a \in \mathbb{Z}$ mit $\mathrm{ggT}(n,a) = 1$ und $0 \le a \le n-1$
Ausgabe: $y \in \mathbb{Z}$ mit $ya \equiv 1 \bmod n$ und $0 \le y \le n-1$
1: Wende den erweiterter euklidischen Algorithmus auf n, a an. Man erhält $x, y \in \mathbb{Z}$ mit $xn + ya = 1$
2: **return** $y \bmod n$

Beispiel: $n = 101$, $a = 37$. Wir wenden den erweiterten euklidischen Algorithmus auf n, a an und erhalten folgendes Schema:

q	a_i	x_i	y_i
	101	1	0
	37	0	1
2	27	1	-2
1	10	-1	3
2	7	3	-8
1	3	-4	11
2	1	11	-30
3	0	-37	101

Die Zahlen der vorletzten Zeile liefern

$$1 = \mathrm{ggT}(a,b) = 11 \cdot n - 30 \cdot a.$$

Dann ist

$$1 \equiv -30a \equiv (101 - 30)a \equiv 71a \bmod n,$$

d.h. 71 ist invers zu 37 modulo 101.

Bemerkung: Ist $\mathrm{ggT}(n,a) = 1$, so bestimmen wir mit dem erweiterten euklidischen Algorithmus $x, y \in \mathbb{Z}$ mit $xn + ya = 1$ um $ya \equiv 1 \bmod n$ zu erhalten. Der x-Wert wird nicht benötigt. Daher könnten wir auch beim erweiterten euklidischen Algorithmus darauf verzichten. Eine mögliche Variante könnte so aussehen:

Inversenberechnung modulo n:
Eingabe: $a, n \in \mathbb{N}$ mit $\mathrm{ggT}(n,a) = 1$
Ausgabe: y mit $ya \equiv 1 \bmod n$ und $0 \le y \le n-1$
1: $u \leftarrow n$, $v \leftarrow a$, $y \leftarrow 0$, $y' \leftarrow 1$
2: **while** $v > 0$ **do**
3: $\quad q \leftarrow \left\lfloor \frac{u}{v} \right\rfloor$
4: $\quad u, v \leftarrow v, u - qv$
5: $\quad y, y' \leftarrow y', y - qy'$
6: **end while**
7: **return** $y \bmod n$

Ein zugehörige Python3-Funktion könnte so aussehen:

```
def invmod(a,n):
  u,v,y,yy=n,a,0,1
  while v>0:
    q=u//v
    u,v,y,yy=v,u-q*v,yy,y-q*yy
  return y%n
```

Bevor wir eine andere Interpretation des Satzes geben, erinnern wir an einen Begriff aus der Algebra: Ist R ein Ring mit Eins 1, so heißt $u \in R$ eine Einheit, falls $v \in R$ existiert mit $uv = vu = 1$. Die Menge der Einheiten bildet bzgl. Multiplikation eine Gruppe, die mit R^* bezeichnet wird.

Sei jetzt $m \in \mathbb{N}$ gegeben und $a \in \mathbb{Z}$. Wir wollen charakterisieren, wann $\overline{a} \in \mathbb{Z}/m\mathbb{Z}$ eine Einheit, also ein Element von $(\mathbb{Z}/m\mathbb{Z})^*$ ist:

$$\begin{aligned} \overline{a} \in (\mathbb{Z}/m\mathbb{Z})^* \quad &\Longleftrightarrow \quad \overline{a} \cdot \overline{b} = \overline{1} \text{ für ein } b \in \mathbb{Z} \quad \Longleftrightarrow \quad ab \equiv 1 \bmod m \text{ für ein } b \in \mathbb{Z} \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad \mathrm{ggT}(a, m) = 1. \end{aligned}$$

Also ist $\overline{a}$ genau dann Einheit in $\mathbb{Z}/m\mathbb{Z}$, wenn $\mathrm{ggT}(a, m) = 1$ gilt. Wir erhalten also:

FOLGERUNG. *Für die Einheiten von $\mathbb{Z}/m\mathbb{Z}$ gilt:*

$$(\mathbb{Z}/m\mathbb{Z})^* = \{\overline{a} : 0 \leq a \leq m-1, \mathrm{ggT}(a, m) = 1\}.$$

Beispiele: Mit der eben angegebenen Beschreibung finden wir:

m	Repräsentanten von $(\mathbb{Z}/m\mathbb{Z})^*$
0	0
1	0
2	1
3	1,2
4	1,3
5	1,2,3,4
6	1,5
7	1,2,3,4,5,6
8	1,3,5,7
9	1,2,4,5,7,8
10	1,3,7,9

DEFINITION. *Die Eulersche φ-Funktion wird definiert durch*

$$\varphi(n) = \#\{a : 0 \leq a \leq n-1, \mathrm{ggT}(a, n) = 1\} = \#(\mathbb{Z}/n\mathbb{Z})^*.$$

Wir geben eine kleine Tabelle:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\varphi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	12	6	8	8	16	6	18	8

SATZ. *Für $n \in \mathbb{N}$ gelten die Äquivalenzen:*

$$n \text{ ist Primzahl} \quad \Longleftrightarrow \quad \varphi(n) = n-1 \quad \Longleftrightarrow \quad \mathbb{Z}/n\mathbb{Z} \text{ ist Körper.}$$

Beweis: Für $n = 1$ sind alle drei Aussagen falsch, die Behauptung also richtig. Daher können wir o.E. $n \geq 2$ voraussetzen.

(1) Wegen $n \geq 2$ ist $\overline{0} \notin (\mathbb{Z}/n\mathbb{Z})^*$, also

$$(\mathbb{Z}/n\mathbb{Z})^* \subseteq \{\overline{1}, \overline{2}, \ldots, \overline{n-1}\} \quad \text{und damit} \quad \varphi(n) \leq n-1.$$

Genau dann gilt $\varphi(n) = n-1$, wenn alle Zahlen zwischen 1 und $n-1$ teilerfremd zu n sind. Dies ist aber äquivalent dazu, dass n keinen nichttrivialen Teiler zwischen 2 und $n-1$ hat, was aber gleichwertig damit ist, dass n eine Primzahl ist.

(2) $\mathbb{Z}/n\mathbb{Z}$ ist genau dann ein Körper, wenn sich alle Elemente $\neq \overline{0}$ invertieren lassen, d.h. wenn gilt $\#(\mathbb{Z}/n\mathbb{Z})^* = \#(\mathbb{Z}/n\mathbb{Z}) - 1 = n-1$, was nach (1) genau dann der Fall ist, wenn n eine Primzahl ist. ■

Bemerkungen:

(1) Ist p eine Primzahl, so schreiben wir statt $\mathbb{Z}/p\mathbb{Z}$ auch $\mathbb{F}_p$.

(2) Ist n keine Primzahl, $n = ab$ mit $1 < a, b < n$, so gilt in $\mathbb{Z}/n\mathbb{Z}$ natürlich $\overline{a} \cdot \overline{b} = \overline{0}$, $\overline{a} \neq \overline{0}$, $\overline{b} \neq \overline{0}$, d.h. $\mathbb{Z}/n\mathbb{Z}$ ist kein Integritätsring.

SATZ. *Für die Eulersche φ-Funktion gilt:*

(1) *Sind m, n natürliche Zahlen, so gilt*

$$\mathrm{ggT}(m, n) = 1 \quad \Longrightarrow \quad \varphi(mn) = \varphi(m)\varphi(n).$$

(Zahlentheoretische Funktionen mit dieser Eigenschaft nennt man multiplikativ.)

(2) *Für $n = p_1^{e_1} \dots p_r^{e_r}$ mit verschiedenen Primzahlen p_i und $e_i \geq 1$ gilt:*

$$\varphi(n) = \prod_i p_i^{e_i - 1}(p_i - 1) = n(1 - \frac{1}{p_1}) \dots (1 - \frac{1}{p_r}) = n \prod_{p|n} (1 - \frac{1}{p}).$$

Beweis:

(1) Dies wird später bewiesen werden.

(2) Für eine Primzahl p und $e \geq 1$ gilt

$$\begin{aligned} \{0 \leq a \leq p^e - 1 : \mathrm{ggT}(a, p^e) = 1\} &= \{0 \leq a \leq p^e - 1 : \mathrm{ggT}(a, p) = 1\} = \\ &= \{a : 0 \leq a \leq p^e - 1\} \setminus \{pb : 0 \leq b \leq p^{e-1} - 1\}, \end{aligned}$$

woraus $\varphi(p^e) = p^e - p^{e-1} = p^{e-1}(p - 1) = p^e(1 - \frac{1}{p})$ folgt. Mit Teil 1 ergibt sich daraus

$$\begin{aligned} \varphi(n) &= \varphi(\prod_i p_i^{e_i}) = \prod_i \varphi(p_i^{e_i}) = \prod_i p_i^{e_i - 1}(p_i - 1) = \\ &= \prod_i p_i^{e_i}(1 - \frac{1}{p_i}) = n \prod_i (1 - \frac{1}{p_i}), \end{aligned}$$

was zu zeigen war. ■

Bemerkung: Der letzte Satz zeigt, dass man $\varphi(n)$ leicht berechnen kann, wenn man die Primfaktorzerlegung von n kennt. Heutzutage ist für große n keine andere Berechnungsmöglichkeit bekannt.

13. Die Gleichung $ax \equiv b \bmod m$

SATZ. *Seien $a, b \in \mathbb{Z}$ und $m \in \mathbb{N}$. Mit dem erweiterten euklidischen Algorithmus findet man $u, v \in \mathbb{Z}$ mit $au + mv = \mathrm{ggT}(a, m)$. Dann gilt:*

(1) *Die Gleichung $ax \equiv b \bmod m$ ist genau dann lösbar, wenn $\mathrm{ggT}(a, m) \mid b$ gilt.*

(2) *Gilt $\mathrm{ggT}(a, m) \mid b$, so erfüllt*

$$x_0 = \frac{b}{\mathrm{ggT}(a, m)} u \quad \text{die Gleichung} \quad ax_0 \equiv b \bmod m.$$

(3) *Gilt $ax_0 \equiv b \bmod m$ (und damit auch $\mathrm{ggT}(a, m) \mid b$), so folgt*

$$\{x \in \mathbb{Z} : ax \equiv b \bmod m\} = \{x_0 + i \cdot \frac{m}{\mathrm{ggT}(a, m)} : i \in \mathbb{Z}\}.$$

(4) *Gilt $\mathrm{ggT}(a, m) \mid b$, so gibt es genau $\mathrm{ggT}(a, m)$ modulo m verschiedene Lösungen der Gleichung $ax \equiv b \bmod m$, z.B.*

$$x_0 + i\frac{m}{\mathrm{ggT}(a, m)} \quad \text{für} \quad i = 0, \dots, \mathrm{ggT}(a, m) - 1,$$

wenn x_0 eine Lösung der Gleichung ist.

Beweis:

(1) Gibt es ein $x \in \mathbb{Z}$ mit $ax \equiv b \bmod m$, so gibt es ein $y \in \mathbb{Z}$ mit $ax = b + my$. Da $\mathrm{ggT}(a, m)$ die Zahlen ax und by teilt, folgt $\mathrm{ggT}(a, m) | b$, wie behauptet. Die Umkehrung wird nachfolgend gezeigt.

(2) Die Voraussetzung $\mathrm{ggT}(a,m) \mid b$ liefert, dass $\frac{b}{\mathrm{ggT}(a,m)}$ eine ganze Zahl ist. Es folgt

$$ax_0 = \frac{abu}{\mathrm{ggT}(a,m)} = \frac{b(\mathrm{ggT}(a,m) - mv)}{\mathrm{ggT}(a,m)} = b - \frac{b}{\mathrm{ggT}(a,m)}mv,$$

was zu $ax_0 \equiv b \bmod m$ führt.

(3) $\subseteq$ Sei $ax \equiv b \bmod m$. Es folgt $ax \equiv ax_0 \bmod m$, also $m \mid a(x - x_0)$. Division durch $\mathrm{ggT}(a,m)$ liefert

$$\frac{m}{\mathrm{ggT}(a,m)} \mid \frac{a}{\mathrm{ggT}(a,m)}(x - x_0).$$

Wegen $\mathrm{ggT}(\frac{a}{\mathrm{ggT}(a,m)}, \frac{m}{\mathrm{ggT}(a,m)}) = 1$ erhält man $\frac{m}{\mathrm{ggT}(a,m)} \mid x - x_0$, also gibt es ein $i \in \mathbb{Z}$ mit

$$x - x_0 = i \cdot \frac{m}{\mathrm{ggT}(a,m)},$$

was die Behauptung liefert.

$\supseteq$ Es ist

$$a\left(x_0 + i\frac{m}{\mathrm{ggT}(a,m)}\right) = ax_0 + i\frac{a}{\mathrm{ggT}(a,m)}m \equiv ax_0 \equiv b \bmod m.$$

(4) Die Zahlen $x_i = x_0 + i\frac{m}{\mathrm{ggT}(a,m)}$, $i \in \mathbb{Z}$, sind genau die ganzzahligen Lösungen der Gleichung $ax \equiv b \bmod m$. Wann sind zwei Lösungen gleich modulo m?

$$\begin{aligned} x_i \equiv x_j \bmod m \quad &\iff \quad i\frac{m}{\mathrm{ggT}(a,m)} \equiv j\frac{m}{\mathrm{ggT}(a,m)} \bmod m \quad \iff \\ &\iff \quad m \mid (i-j)\frac{m}{\mathrm{ggT}(a,m)} \quad \iff \quad \mathrm{ggT}(a,m)m \mid (i-j)m \quad \iff \\ &\iff \quad \mathrm{ggT}(a,m) \mid i - j \quad \iff \quad i \equiv j \bmod \mathrm{ggT}(a,m). \end{aligned}$$

Damit folgt die Behauptung. ■

Beispiel: $6x \equiv 4 \bmod 10$ hat die zwei Lösungen 4 und 9 modulo 10. Dagegen hat $6x \equiv 5 \bmod 10$ keine Lösung.

Oft sind auch die (einfachen) Aussagen des folgenden Satzes hilfreich:

Satz. *Seien $a, b \in \mathbb{Z}$ und $m \in \mathbb{N}$.*

(1) *Ist $d = \mathrm{ggT}(a,b,m)$ und schreibt man $a = da'$, $b = db'$, $m = dm'$, so gilt $\mathrm{ggT}(a',b',m') = 1$, und für $x \in \mathbb{Z}$ hat man die Äquivalenz:*

$$ax \equiv b \bmod m \quad \iff \quad a'x \equiv b' \bmod m'.$$

(2) *Gilt $\mathrm{ggT}(a,m) = 1$ und ist $\widetilde{a}$ invers zu a modulo m, d.h. $\widetilde{a}a \equiv 1 \bmod m$, so hat man die Äquivalenz:*

$$ax \equiv b \bmod m \quad \iff \quad x \equiv \widetilde{a}b \bmod m.$$

(3) *Gilt $\mathrm{ggT}(a,m) > 1$ und $\mathrm{ggT}(a,m) \nmid b$, so hat die Gleichung $ax \equiv b \bmod m$ keine Lösungen.*

Beweis:

(1) Es gilt:

$$\begin{aligned} ax \equiv b \bmod m \quad &\iff \quad m \mid ax - b \quad \iff \quad dm' \mid da'x - db' \quad \iff \\ &\iff \quad m' \mid a'x - b' \quad \iff \quad a'x \equiv b' \bmod m'. \end{aligned}$$

(2) Es gilt:

$$\begin{aligned} ax \equiv b \bmod m \quad &\overset{\cdot\widetilde{a}}{\Longrightarrow} \quad \widetilde{a}ax \equiv \widetilde{a}b \bmod m \quad \iff \quad x \equiv \widetilde{a}b \bmod m \quad \Longrightarrow \\ &\overset{\cdot a}{\Longrightarrow} \quad ax \equiv a\widetilde{a}b \bmod m \quad \iff \quad ax \equiv b \bmod m, \end{aligned}$$

woraus die Behauptung folgt.

(3) Die Aussage findet sich schon im vorangegangenen Satz. ■

14. Der chinesische Restsatz

Der folgende Satz spielt ebenfalls eine wichtige Rolle. Wir geben ihn in der meist gebrauchten Form an:

SATZ (Chinesischer Restsatz). *Seien $m_1, m_2, \ldots, m_r \in \mathbb{N}$ mit $\mathrm{ggT}(m_i, m_j) = 1$ für alle $i \neq j$ und $a_1, \ldots, a_r \in \mathbb{Z}$ gegeben. Dann ist das Kongruenzgleichungssystem*

$$\begin{aligned} x &\equiv a_1 \bmod m_1, \\ x &\equiv a_2 \bmod m_2, \\ &\vdots \\ x &\equiv a_r \bmod m_r \end{aligned}$$

lösbar und die Lösung ist eindeutig bestimmt modulo $m_1 \ldots m_r$.
Genauer:

(1) *Definiert man*

$$M = \prod_{i=1}^{r} m_i \quad \text{und} \quad M_i = \frac{M}{m_i} = \prod_{\substack{1 \leq j \leq r \\ j \neq i}} m_j \text{ für } 1 \leq i \leq r,$$

so gilt $\mathrm{ggT}(M_i, m_i) = 1$, sodass M_i invertierbar modulo m_i ist, d.h. es existiert ein $N_i \in \mathbb{Z}$ mit

$$M_i N_i \equiv 1 \bmod m_i.$$

Dann löst

$$a = \sum_{i=1}^{r} a_i M_i N_i \quad \text{und damit auch} \quad a = \left(\sum_{i=1}^{r} a_i M_i N_i \right) \bmod m_1 \ldots m_r$$

das Kongruenzgleichungssystem. d.h. $a \equiv a_i \bmod m_i$ für alle i.

(2) *Sind x und x' Lösungen des Kongruenzgleichungssystems, so gilt $x \equiv x' \bmod M$.*

(3) *Ist x eine Lösung des Kongruenzgleichungssystem und ist $x' \equiv x \bmod M$, so ist auch x' eine Lösung des Kongruenzgleichungssystems.*

Beweis:

(1) Da die Zahlen $m_1, \ldots, m_r$ als paarweise teilerfremd vorausgesetzt waren, sind auch m_i und $M_i = m_1 m_2 \ldots m_{i-1} m_{i+1} \ldots m_r$ teilerfremd. Also ist M_i invertierbar modulo m_i, d.h. es gibt ein $N_i \in \mathbb{Z}$ mit $M_i N_i \equiv 1 \bmod m_i$. Wir betrachten nun die Zahl

$$a = \sum_j a_j M_j N_j.$$

Für einen Index i gilt $M_i N_i \equiv 1 \bmod m_i$ nach Konstruktion und $M_j N_j \equiv 0 \bmod m_i$ für $j \neq i$ wegen $m_i \mid M_j$. Es folgt

$$a \equiv \sum_j a_j M_j N_j \equiv a_i M_i N_i \equiv a_i \bmod m_i.$$

Dies zeigt, dass a das Kongruenzgleichungssystem löst.

(2) Ist x' eine weitere Lösung des Kongruenzensystems, so folgt $x' \equiv a \equiv x \bmod m_i$ und damit $m_i \mid x' - x$. Die Teilerfremdheit der m_i's liefert $m_1 m_2 \ldots m_r \mid x' - x$, also $x' \equiv x \bmod m_1 m_2 \ldots m_r$, wie behauptet.

(3) Ist x eine Lösung des Kongruenzgleichungssystems, so gilt $x \equiv a_i \bmod m_i$ für alle i. Gilt nun für $x' \in \mathbb{Z}$ die Beziehung $x' \equiv x \bmod M$, so folgt $x' \equiv x \bmod m_i$ und damit auch $x' \equiv x \equiv a_i \bmod m_i$, d.h. auch x' ist eine Lösung des Kongruenzgleichungssystems. Damit folgt auch die Richtigkeit der zweiten Formeln in (1). ■

Beispiele:

(1) Wir suchen eine Lösung des Kongruenzgleichungssystems

$$x \equiv 1 \bmod 2, \quad x \equiv 0 \bmod 3, \quad x \equiv 4 \bmod 5.$$

Der chinesische Restsatz besagt, dass eine Lösung existiert, und dass die Lösung modulo $30 = 2 \cdot 3 \cdot 5$ bestimmt ist. Es gibt also genau eine Lösung zwischen 0 und 29. Wir schreiben zuerst die Zahlen zwischen 0 und 29 mit $x \equiv 4 \bmod 5$ an:

$$4, \quad 9, \quad 14, \quad 19, \quad 24, \quad 29.$$

Nun wählen wir aus, welcher dieser Zahlen auch $\equiv 0 \bmod 3$ sind:

$$9, \quad 24.$$

Nun betrachten wir die Zahlen, die zusätzlich $x \equiv 1 \bmod 2$ erfüllen:

$$9.$$

Also löst 9 das gegebene Kongruenzgleichungssystem.

(2) Wir behandeln das letzte Beispiel nochmals mit den Formeln des vorangegangenen Satzes, wählen also

$$a_1 = 1, \quad a_2 = 0, \quad a_3 = 4, \quad m_1 = 2, \quad m_2 = 3, \quad m_3 = 5.$$

Wir erhalten zunächst mit $M = m_1 m_2 m_3 = 30$:

i	m_i	M_i	$M_i \bmod m_i$	N_i	$M_i N_i$	$a_i M_i N_i$
1	2	15	1	1	15	15
2	3	10	1	1	10	0
3	5	6	1	1	6	24

Also ist $\sum_{i=1}^{3} a_i M_i N_i = 39 \equiv 9 \bmod 30$ eine Lösung.

(3) Wir starten mit

$$m_1 = 2, m_2 = 3, m_3 = 5, m_4 = 7, m_5 = 11, m_6 = 13, m_7 = 17, m_8 = 19, m_9 = 23, m_{10} = 29$$

und

$$a_1 = 1, a_2 = 2, a_3 = 3, a_4 = 4, a_5 = 5, a_6 = 6, a_7 = 7, a_8 = 8, a_9 = 9, a_{10} = 10.$$

Mit $M = m_1 \dots m_{10} = 6469693230$, $M_i = \frac{M}{m_i}$ und $N_i = \frac{1}{M_i} \bmod m_i$ gilt

i	m_i	M_i	$M_i \bmod m_i$	N_i	$M_i N_i$	$a_i M_i N_i$
1	2	3234846615	1	1	3234846615	3234846615
2	3	2156564410	1	1	2156564410	4313128820
3	5	1293938646	1	1	1293938646	3881815938
4	7	924241890	5	3	2772725670	11090902680
5	11	588153930	1	1	588153930	2940769650
6	13	497668710	6	11	5474355810	32846134860
7	17	380570190	13	4	1522280760	10655965320
8	19	340510170	17	9	3064591530	24516732240
9	23	281291010	21	11	3094201110	27847809990
10	29	223092870	17	12	2677114440	26771144400

Daher ist

$$\sum a_i M_i N_i = 148099250513 \quad \text{bzw.} \quad (\sum a_i M_i N_i) \bmod M) = 5765999453$$

eine Lösung des Kongruenzgleichungssystems.

Bemerkung: Computeralgebrasysteme haben üblicherweise den chinesischen Restsatz implementiert. So erhält man bei Maple mit dem Befehl `chrem([a_1,...,a_r],[m_1,...,m_r])`, bei Sage mit dem Befehl `crt([a_1,...,a_r],[m_1,...,m_r])` eine Lösung des Kongruenzgleichungssystems $x \equiv a_i \bmod m_i$ ($i = 1, \dots, r$).

Bemerkung: Wir betrachten jetzt den Fall $r = 2$, also ein Kongruenzgleichungssystem

$$x \equiv a_1 \bmod m_1, \quad x \equiv a_2 \bmod m_2.$$

Mit den Bezeichnungen des Satzes ist $M_1 = m_2$, $M_2 = m_1$. Mit dem erweiterten euklidischen Algorithmus bestimmen wir jetzt N_1, N_2 mit

$$N_1 M_1 + N_2 M_2 = 1.$$

(Dann gilt nämlich $N_1 M_1 \equiv 1 \bmod m_1$ und $N_2 M_2 \equiv 1 \bmod m_2$.) Dann lösen

$$a_1 M_1 N_1 + a_2 M_2 N_2 \quad \text{und} \quad (a_1 M_1 N_1 + a_2 M_2 N_2) \bmod m_1 m_2$$

das Kongruenzgleichungssystem. Wir formulieren das Ergebnis als Satz:

SATZ. *Seien $m_1, m_2 \in \mathbb{N}$ mit $\mathrm{ggT}(m_1, m_2) = 1$ und $a_1, a_2 \in \mathbb{Z}$. Sei $M_1 = m_2$ und $M_2 = m_1$. Bestimme mit dem erweiterten euklidischen Algorithmus $N_1, N_2 \in \mathbb{Z}$ mit $N_1 M_1 + N_2 M_2 = 1$. Dann löst*

$$a = (a_1 M_1 N_1 + a_2 M_2 N_2) \bmod m_1 m_2$$

das Kongruenzgleichungssystem

$$x \equiv a_1 \bmod m_1, \quad x \equiv a_2 \bmod m_2.$$

Beispiel: Wir suchen eine Lösung des Gleichungssystems

$$x \equiv 123 \bmod 256, \quad x \equiv 456 \bmod 997.$$

Wir setzen $M_1 = 997$, $M_2 = 256$ und wenden den erweiterten euklidischen Algorithmus darauf an:

q	a_i	x_i	y_i
	997	1	0
	256	0	1
3	229	1	-3
1	27	-1	4
8	13	9	-35
2	1	-19	74
13	0	256	-997

Die Zahlen der vorletzten Zeile liefern

$$1 = \mathrm{ggT}(M_1, M_2) = -19 \cdot M_1 + 74 \cdot M_2.$$

Wir setzen $N_1 = -19$, $N_2 = 74$ und erhalten mit $a_1 = 123$ und $a_2 = 456$ als Lösung des Kongruenzgleichungssystems

$$a = (a_1 M_1 N_1 + a_2 M_2 N_2) \bmod 256 \cdot 997 = (6308475 \bmod 255232) = 182907.$$

Wir betrachten nun kurz die Umwandlung von Kongruenzgleichungssystemen. Der nächste Satz ist eine andere Formulierung des chinesischen Restsatzes.

SATZ. *Seien $m_1, m_2, \ldots, m_r$ paarweise teilerfremde natürliche Zahlen und $a_1, a_2, \ldots, a_r$ ganze Zahlen. Dann gibt es eine ganze Zahl a, die man mit dem chinesischen Restsatz finden kann, mit der Eigenschaft:*

$$\begin{cases} x \equiv a_1 \bmod m_1 \\ x \equiv a_2 \bmod m_2 \\ \vdots \\ x \equiv x_r \bmod m_r \end{cases} \iff x \equiv a \bmod m_1 m_2 \ldots m_r.$$

Man kann also verschiedene Kongruenzgleichungen zu einer zusammenfassen.
Mit dem folgenden Lemma kann man eine Kongruenzgleichung in mehrere aufteilen:

LEMMA. *Seien $m_1, m_2, \ldots, m_r$ paarweise teilerfremde natürliche Zahlen und a eine ganze Zahl. Dann sind folgende Gleichungen äquivalent:*

$$x \equiv a \bmod m_1 m_2 \ldots m_r \quad \Longleftrightarrow \quad \begin{cases} x \equiv a \bmod m_1 \\ x \equiv a \bmod m_2 \\ \vdots \\ x \equiv a \bmod m_r \end{cases}$$

Bemerkung: Was passiert mit Primzahlpotenzen? Seien $a, b \in \mathbb{Z}$ und $m, n \in \mathbb{N}$, o.E. $m \geq n$.

$$\begin{aligned} & x \equiv a \bmod p^m \text{ und } x \equiv b \bmod p^n \\ \Longleftrightarrow \quad & x \equiv a \bmod p^m \text{ und } x \equiv a \bmod p^n \text{ und } x \equiv b \bmod p^n \\ \Longleftrightarrow \quad & x \equiv a \bmod p^m \text{ und } a \equiv b \bmod p^n. \end{aligned}$$

Es gibt also zwei Möglichkeiten:

- Ist $a \equiv b \bmod p^n$, so lässt sich das Kongruenzgleichungssystem durch die eine Gleichung $x \equiv a \bmod p^m$ ausdrücken.
- Ist $a \not\equiv b \bmod p^n$, so ist das Gleichungssystem nicht lösbar.

Wir fassen das Ergebnis zusammen:

SATZ. *Sei p eine Primzahl, $m, n \in \mathbb{N}$ mit $m \geq n$ und $a, b \in \mathbb{Z}$. Dann gilt:*

$$\{x \in \mathbb{Z} : x \equiv a \bmod p^m \text{ und } x \equiv b \bmod p^n\} = \begin{cases} \{x \in \mathbb{Z} : x \equiv a \bmod p^m\}, & \text{falls } a \equiv b \bmod p^n, \\ \emptyset, & \text{falls } a \not\equiv b \bmod p^n. \end{cases}$$

Beispiel: Wir betrachten das Kongruenzgleichungssystems:

$$x \equiv 3 \bmod 10, \quad x \equiv 5 \bmod 12.$$

Wir formen äquivalent um:

$$\begin{aligned} & x \equiv 3 \bmod 10, \quad x \equiv 5 \bmod 12 \quad \Longleftrightarrow \\ \Longleftrightarrow \quad & x \equiv 3 \bmod 2, \quad x \equiv 3 \bmod 5, \quad x \equiv 5 \bmod 3, \quad x \equiv 5 \bmod 4 \quad \Longleftrightarrow \\ \Longleftrightarrow \quad & x \equiv 1 \bmod 2, \quad x \equiv 1 \bmod 4, \quad x \equiv 2 \bmod 3, \quad x \equiv 3 \bmod 5 \quad \Longleftrightarrow \\ \Longleftrightarrow \quad & x \equiv 1 \bmod 4, \quad x \equiv 2 \bmod 3, \quad x \equiv 3 \bmod 5 \quad \Longleftrightarrow \\ \Longleftrightarrow \quad & x \equiv 5 \bmod 12, \quad x \equiv 3 \bmod 5 \quad \Longleftrightarrow \\ \Longleftrightarrow \quad & x \equiv 53 \bmod 60. \end{aligned}$$

15. Varianten des chinesischen Restsatzes und eine Anwendung

Hier ist noch eine andere Form des chinesischen Restsatzes für zwei Kongruenzen:

SATZ. *Seien m_1 und m_2 zwei teilerfremde natürliche Zahlen, a_1 und a_2 zwei ganze Zahlen.*

(1) *Mit dem erweiterten euklidischen Algorithmus findet man eine ganze Zahl M_2 mit*

$$M_2 m_2 \equiv 1 \bmod m_1.$$

(2) *Man berechnet sich eine ganze Zahl y mit*

$$y \equiv M_2(a_1 - a_2) \bmod m_1.$$

(3) *Sei*

$$x = a_2 + y m_2.$$

Dann gilt

$$x \equiv a_1 \bmod m_1 \quad \text{und} \quad x \equiv a_2 \bmod m_2.$$

Ist $0 \leq a_2 \leq m_2 - 1$ und $0 \leq y \leq m_1 - 1$, so gilt $0 \leq x \leq m_1 m_2 - 1$.

Beweis: Seien M_2, y und x wie angegeben definiert.

(1) Wir rechnen modulo m_1:

$$x \equiv a_2 + ym_2 \equiv a_2 + M_2(a_1 - a_2)m_2 \equiv a_2 + M_2m_2(a_1 - a_2) \equiv a_2 + 1 \cdot (a_1 - a_2) \equiv a_1 \bmod m_1.$$

(2) Trivialerweise gilt

$$x \equiv a_2 \bmod m_2.$$

(3) Gilt $0 \leq a_2 \leq m_2 - 1$ und $0 \leq y \leq m_1 - 1$, so folgt

$$0 \leq x = a_2 + ym_2 \leq m_2 - 1 + (m_1 - 1)m_2 = m_1m_2 - 1,$$

was die Behauptung beweist. ■

Der folgende Satz zeigt eine Variante, bei der die paarweise Teilerfremdheit nicht mehr vorausgesetzt wird:

SATZ. *Gegeben seien Zahlen $m_1, m_2 \in \mathbb{N}$ und $a_1, a_2 \in \mathbb{Z}$. Durch*

$$d = \mathrm{ggT}(m_1, m_2), \quad m_1 = dn_1, \quad m_2 dn_2$$

werden dann natürliche Zahlen d, n_1, n_2 definiert mit $\mathrm{ggT}(n_1, n_2) = 1$.

(1) *Das Kongruenzgleichungssystem*

$$x \equiv a_1 \bmod m_1, \quad x \equiv a_2 \bmod m_2$$

ist genau dann lösbar, wenn

$$d \mid a_2 - a_1$$

gilt.

(2) *Es gelte $d \mid a_2 - a_1$. Wegen $\mathrm{ggT}(n_1, n_2) = 1$ existiert ein $l \in \mathbb{Z}$ mit $ln_1 \equiv 1 \bmod n_2$. Dann ist*

$$a_0 = a_1 + l\frac{a_2 - a_1}{d}m_1$$

eine Lösung des Kongruenzgleichunssystems. Für $x \in \mathbb{Z}$ sind äquivalent:

$$x \equiv a_1 \bmod m_1 \text{ und } x \equiv a_2 \bmod m_2 \quad \Longleftrightarrow \quad x \equiv a_0 \bmod \mathrm{kgV}(m_1, m_2).$$

Beweis:

(1) Genau dann löst x die Kongruenz $x \equiv a_1 \bmod m_1$, wenn es ein $u \in \mathbb{Z}$ gibt mit

$$x = a_1 + um_1.$$

(2) Wann löst $x = a_1 + um_1$ die zweite Kongruenzgleichung? Es gilt:

$$\begin{aligned} a_1 + um_1 \equiv a_2 \bmod m_2 \quad &\Longleftrightarrow \quad um_1 \equiv a_2 - a_1 \bmod m_2 \iff \\ &\Longleftrightarrow \quad dun_1 \equiv a_2 - a_1 \bmod dn_2. \end{aligned}$$

Gilt $d \nmid a_2 - a_1$, so ist die Gleichung nicht lösbar. Dies zeigt bereits einen Teil von (1).

(3) Wir setzen jetzt voraus, dass $d \mid a_2 - a_1$ gilt, d.h. $\frac{a_2 - a_1}{d} \in \mathbb{Z}$. Mit $ln_1 \equiv 1 \bmod n_2$ können wir dann weiter umformen:

$$\begin{aligned} a_1 + um_1 \equiv a_2 \bmod m_2 \quad &\Longleftrightarrow \quad dun_1 \equiv a_2 - a_1 \bmod dn_2 \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad un_1 \equiv \frac{a_2 - a_1}{d} \bmod n_2 \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad un_1 l \equiv l\frac{a_2 - a_1}{d} \bmod n_2 \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad u \equiv l\frac{a_2 - a_1}{d} \bmod n_2. \end{aligned}$$

Dies ist genau dann der Fall, wenn es ein $v \in \mathbb{Z}$ gibt mit

$$u = l\frac{a_2 - a_1}{d} + vn_2.$$

Eingesetzt in $x = a_1 + um_1$ erhalten wir als Lösungen (mit $\mathrm{kgV}(m_1m_2) = dn_1n_2 = n_2m_1$)

$$\begin{aligned} x &= a_1 + um_1 = a_1 + \left(l\frac{a_2 - a_1}{d} + vn_2\right) \cdot m_1 = \\ &= a_1 + l\frac{a_2 - a_1}{d}m_1 + vn_2m_1 = a_1 + l\frac{a_2 - a_1}{d}m_1 + v\mathrm{kgV}(m_1, m_2). \end{aligned}$$

Die angegebene Äquivalenz der Kongruenzgleichungen gilt, weil die Lösungsmengen gleich sind. ■

Wir beweisen jetzt mit dem chinesischen Restsatz eine zuvor behauptete Eigenschaft der Eulerschen φ-Funktion:

LEMMA. *Für natürliche Zahlen m und n gilt:*

$$\mathrm{ggT}(m, n) = 1 \quad \Longrightarrow \quad \varphi(mn) = \varphi(m)\varphi(n).$$

Beweis: Wir betrachten die drei Mengen

$$\begin{aligned} A &= \{0 \le a \le mn - 1 : \mathrm{ggT}(a, mn) = 1\}, \\ B &= \{0 \le b \le m - 1 : \mathrm{ggT}(b, m) = 1\}, \\ C &= \{0 \le c \le n - 1 : \mathrm{ggT}(c, n) = 1\}, \end{aligned}$$

sodass gilt

$$\varphi(mn) = \#A, \quad \varphi(m) = \#B, \quad \varphi(n) = \#C.$$

- Für $a \in A$ sei $(a \bmod m)$ der Repräsentant der Äquivalenzklasse im Intervall zwischen 0 und $m - 1$, d.h. $(a \bmod m) = r$ mit $a = qm + r$ und $0 \le r \le m - 1$. Dann ist $\mathrm{ggT}(a \bmod m, m) = \mathrm{ggT}(r, m) = \mathrm{ggT}(a, m) = 1$ wegen $\mathrm{ggT}(a, mn) = 1$, und somit $(a \bmod m) \in B$. Analog definiert man $(a \bmod n)$ und erhält $(a \bmod n) \in C$. Durch
$$f : A \to B \times C, \quad a \mapsto (a \bmod m, a \bmod n)$$
wird daher eine Abbildung definiert.
- *f ist injektiv:* Aus $f(a) = f(a')$ folgt $a \equiv a' \bmod m$ und $a \equiv a' \bmod n$, was wegen $\mathrm{ggT}(m, n) = 1$ sofort $a \equiv a' \bmod mn$, und damit wegen $a, a' \in A$ auch $a = a'$ liefert.
- *f ist surjektiv:* Sei $(b, c) \in B \times C$. Mit dem chinesischen Restsatz findet man ein $a \in \mathbb{Z}$ mit $a \equiv b \bmod m$, $a \equiv c \bmod n$. Da a nur modulo mn bestimmt ist, kann man $0 \le a \le mn - 1$ wählen. Man sieht auch gleich, dass $\mathrm{ggT}(a, mn) = 1$ gilt, d.h. $a \in A$ und damit $f(a) = (b, c)$.

Die Bijektivität von f liefert dann sofort $\#A = \#(B \times C) = \#B \cdot \#C$ und damit $\varphi(mn) = \varphi(m)\varphi(n)$. ■

Mitunter versteht man unter dem chinesischen Restsatz auch folgenden algebraischen Satz:

SATZ. *Seien $m_1, \ldots, m_r \in \mathbb{N}$ mit $\mathrm{ggT}(m_i, m_j) = 1$ für $i \ne j$. Dann definiert*

$$\phi : \mathbb{Z}/m_1m_2 \ldots m_r\mathbb{Z} \to \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \times \cdots \times \mathbb{Z}/m_r\mathbb{Z}, \quad x \mapsto (x \bmod m_1, x \bmod m_2, \ldots, x \bmod m_r)$$

einen Ringisomorphismus.

Beweisskizze: Man sieht zunächst, dass die Abbildung wohldefiniert ist, dass Definitions- und Bildbereich gleiche Mächtigkeit $m_1 \ldots m_r$ haben. Man überlegt, dass ϕ mit den Rechenoperationen verträglich ist, d.h. ϕ ist ein Ringhomomorphismus. Der chinesische Restsatz liefert nun, dass die Abbildung surjektiv ist. Also ist ϕ auch bijektiv wegen der Mächtigkeitsaussagen. Dies beweist die Behauptung. ■

Alternativer Beweis von $\varphi(mn) = \varphi(m)\varphi(n)$ für $\mathrm{ggT}(m, n) = 1$: Der letzte Satz liefert einen Ringisomorphismus

$$\phi : \mathbb{Z}/mn\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}.$$

Da ein Ringisomorphismus natürlich Einheiten in Einheiten überführt, folgt

$$(\mathbb{Z}/mn\mathbb{Z})^* \simeq (\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z})^* = (\mathbb{Z}/m\mathbb{Z})^* \times (\mathbb{Z}/n\mathbb{Z})^*,$$

sodass man durch Betrachtung der Mächtigkeiten sofort die Behauptung $\varphi(mn) = \varphi(m)\varphi(n)$ erhält. ■

16. Anhang: Ein Algorithmus zur Berechnung von $\lfloor\sqrt{n}\rfloor$

LEMMA. *Sei $n \in \mathbb{N}$ gegeben. Wir definieren eine Funktion*

$$f : \mathbb{N} \to \mathbb{N} \quad \textit{mit} \quad f(x) = \left\lfloor \frac{1}{2}\left(x + \left\lfloor \frac{n}{x} \right\rfloor\right) \right\rfloor .$$

Es gilt:

(1)
$$f(x) = \left\lfloor \frac{1}{2}\left(x + \frac{n}{x}\right) \right\rfloor = \left\lfloor \sqrt{n} + \frac{1}{2x}\left(x - \sqrt{n}\right)^2 \right\rfloor .$$

(2)
$$f(x) \geq \lfloor \sqrt{n} \rfloor .$$

(3) *Für $x > \sqrt{n}$ ist $f(x) < x$.*

(4) *Schreibt man $n = \lfloor\sqrt{n}\rfloor^2 + k$, so gilt $0 \leq k \leq 2\lfloor\sqrt{n}\rfloor$ und*
$$f(\lfloor\sqrt{n}\rfloor) = \begin{cases} \lfloor\sqrt{n}\rfloor & \textit{im Fall } 0 \leq k < 2\lfloor\sqrt{n}\rfloor , \\ \lfloor\sqrt{n}\rfloor + 1 & \textit{im Fall } k = 2\lfloor\sqrt{n}\rfloor . \end{cases}$$

(5) *Im Fall $x > \sqrt{n}$ und $f(x) > \sqrt{n}$ gilt*
$$0 < f(x) - \sqrt{n} < \frac{1}{2}(x - \sqrt{n}).$$

Beweis:

(0) Zunächst liegen die Werte von f offensichtlich in $\mathbb{Z}$. Nun ist aber
$$f(1) = \left\lfloor \frac{1}{2}(1+n) \right\rfloor \geq 1 \quad \text{und} \quad f(x) = \left\lfloor \frac{1}{2}\left(x + \left\lfloor \frac{n}{x} \right\rfloor\right) \right\rfloor \geq \left\lfloor \frac{1}{2}x \right\rfloor \geq 1 \text{ für } x \geq 2.$$
Daher ist $\mathbb{N}$ als Wertebereich von f sinnvoll.

(1) Für $x \in \mathbb{N}$ ist $\frac{1}{2}(x + \lfloor \frac{n}{x} \rfloor) \in \frac{1}{2}\mathbb{Z}$, sodass mit $0 \leq \frac{1}{2}(\frac{n}{x} - \lfloor \frac{n}{x} \rfloor) < \frac{1}{2}$ folgt
$$\begin{aligned} f(x) &= \left\lfloor \frac{1}{2}\left(x + \left\lfloor \frac{n}{x} \right\rfloor\right) \right\rfloor = \left\lfloor \frac{1}{2}\left(x + \left\lfloor \frac{n}{x} \right\rfloor\right) + \frac{1}{2}\left(\frac{n}{x} - \left\lfloor \frac{n}{x} \right\rfloor\right) \right\rfloor = \left\lfloor \frac{1}{2}\left(x + \frac{n}{x}\right) \right\rfloor = \\ &= \left\lfloor \sqrt{n} + \frac{1}{2x}\left(x^2 - 2x\sqrt{n} + n\right) \right\rfloor = \left\lfloor \sqrt{n} + \frac{1}{2x}\left(x - \sqrt{n}\right)^2 \right\rfloor . \end{aligned}$$

(2) Dies ergibt sich sofort aus der zweiten Darstellung in (1).

(3) Für $x > \sqrt{n}$ folgt mit $n < x^2$
$$f(x) = \left\lfloor \frac{1}{2}\left(x + \frac{n}{x}\right) \right\rfloor \leq \frac{1}{2}\left(x + \frac{n}{x}\right) < \frac{1}{2}\left(x + \frac{x^2}{x}\right) = x.$$

(4) Wir schreiben der Kürze halber $m = \lfloor\sqrt{n}\rfloor$ und haben dann $n = m^2 + k$. Es ist $m \leq \sqrt{n} < m+1$ und damit $m^2 \leq n < m^2 + 2m + 1$, also $m^2 \leq n \leq m^2 + 2m$. Setzen wir $n = m^2 + k$ ein, so folgt $m^2 \leq m^2 + k \leq m^2 + 2m$, also $0 \leq k \leq 2m$, wie behauptet. Mit der ersten Darstellung aus (1) ergibt sich
$$f(m) = \left\lfloor \frac{1}{2}\left(m + \frac{n}{m}\right) \right\rfloor = \left\lfloor \frac{1}{2}\left(m + \frac{m^2 + k}{m}\right) \right\rfloor = \left\lfloor m + \frac{k}{2m} \right\rfloor = \begin{cases} m & \text{für } 0 \leq k < 2m, \\ m+1 & \text{für } k = 2m, \end{cases}$$
wie behauptet.

(5) Es gelte jetzt $x > \sqrt{n}$ und $f(x) > \sqrt{n}$. Dann folgt
$$0 \quad < \quad \frac{f(x) - \sqrt{n}}{x - \sqrt{n}} \leq \frac{\frac{1}{2}(x + \frac{n}{x}) - \sqrt{n}}{x - \sqrt{n}} = \frac{x^2 + n - 2x\sqrt{n}}{2x(x - \sqrt{n})} = \frac{(x - \sqrt{n})^2}{2x(x - \sqrt{n})} = \frac{x - \sqrt{n}}{2x} < \frac{1}{2}.$$
Dies kann man auch in der Form
$$0 < f(x) - \sqrt{n} < \frac{1}{2}(x - \sqrt{n})$$
schreiben. ■

SATZ. *Sei $n \in \mathbb{N}$ gegeben. Wir definieren rekursiv eine Folge natürlicher Zahlen durch*

$$x_1 = n, \quad x_{i+1} = \left\lfloor \frac{1}{2}\left(x_i + \left\lfloor \frac{n}{x_i} \right\rfloor\right)\right\rfloor \text{ für } i \geq 1.$$

Sei $\ell \in \mathbb{N}$ der kleinste Index mit $x_{\ell+1} \geq x_\ell$. (So etwas gibt es, weil eine Folge natürlicher Zahlen nicht streng monoton fallend sein kann.) Es ist

$$x_\ell = \lfloor\sqrt{n}\rfloor \quad \text{und} \quad \ell < 3 + \frac{\log n}{\log 2}.$$

Beweis:

(1) Wir verwenden die Definition und die Aussagen des vorangegangenen Lemmas. Es ist

$$x_{i+1} = f(x_i).$$

(2) Zunächst ist nach Teil (2) des Lemmas $x_i \geq \lfloor\sqrt{n}\rfloor$ für alle $i \geq 1$. Ist $x_i > \sqrt{n}$, so folgt aus Teil (3) sofort $x_{i+1} < x_i$. Die Forderung $x_{\ell+1} \geq x_\ell$ liefert daher $x_\ell \leq \sqrt{n}$, also $x_\ell \leq \lfloor\sqrt{n}\rfloor$, was mit (2) sofort $x_\ell = \lfloor\sqrt{n}\rfloor$ liefert.

(3) Es gilt

$$\sqrt{n} < x_{\ell-1} < x_{\ell-2} < \cdots < x_1$$

und damit

$$x_{\ell-2} \geq x_{\ell-1} + 1 > \sqrt{n} + 1, \quad \text{also} \quad 1 < x_{\ell-2} - \sqrt{n},$$

und mit Teil (5) des Lemmas

$$\begin{aligned} 1 &< x_{\ell-2} - \sqrt{n} < \frac{1}{2}(x_{\ell-3} - \sqrt{n}) < \frac{1}{2^2}(x_{\ell-4} - \sqrt{n}) < \cdots < \frac{1}{2^{\ell-3}}(x_1 - \sqrt{n}) = \\ &= \frac{1}{2^{\ell-3}}(n - \sqrt{n}) < \frac{1}{2^{\ell-3}} \cdot n \end{aligned}$$

also $2^{\ell-3} < n$, und damit

$$\ell - 3 < \frac{\log n}{\log 2}, \quad \text{also} \quad \ell < 3 + \frac{\log n}{\log 2},$$

wie behauptet. ■

Der Satz liefert sofort folgenden Algorithmus:

Eingabe: $n \in \mathbb{N}$
Ausgabe: $\lfloor\sqrt{n}\rfloor$

```
1: x ← n
2: loop
3:     y ← ⌊½(x + ⌊n/x⌋)⌋
4:     if y ≥ x then
5:         return x
6:     end if
7:     x ← y
8: end loop
```

Eine zugehörige Python3-Funktion könnte so aussehen:

```
def sqrt(n):
  x=n
  while True:
    y=(x+n//x)//2
    if y>=x:
      return x
    x=y
```