Primzahlen im Schulunterricht – wozu?

FRANZ PAUER, FLORIAN STAMPFER (UNIVERSITÄT INNSBRUCK)

1. Einleitung

Eine natürliche Zahl heißt *Primzahl*, wenn sie genau zwei Teiler hat. Im Lehrplan der Sekundarstufe 1 kommen Primzahlen nicht vor und dies mit gutem Grund. Für das Verständnis der Inhalte des Mathematikunterrichts in dieser Schulstufe werden Primzahlen nicht benötigt. In manchen Lehrplänen der Sekundarstufe 2 spielen Primzahlen aber eine wichtige Rolle, nämlich im Rahmen der Kodierungstheorie und Kryptographie. Ihre große Bedeutung in diesen Bereichen erlangen die Primzahlen durch „Kooperation" mit dem erweiterten Euklidischen Algorithmus.

In diesem Beitrag stellen wir ein Verfahren zur Verschlüsselung vor, bei dem offen bekannt gegeben wird, wie man eine Nachricht verschlüsselt. Das Entschlüsseln ist dann zwar theoretisch auch für jeden möglich, praktisch aber nur für den, der eine Zusatzinformation hat. Grundlegend dafür ist die Tatsache, dass auch die schnellsten Computer für die Zerlegung einer genügend großen natürlichen Zahl in ihre Primfaktoren hundert oder mehr Jahre Rechenzeit brauchen. In den Abschnitten 3 bis 7 begründen wir dieses Verfahren, dazu gehen wir auf den verallgemeinerten Euklidischen Algorithmus, das Rechnen mit Restklassen und den kleinen Satz von Fermat ein. Wir haben die Begründungen so geschrieben, dass sie auch von Schüler/innen ab der 11. Schulstufe gelesen werden könnten. Im letzten Abschnitt lassen wir den Euklidischen Algorithmus gegen das Verfahren zur Berechnung des größten gemeinsamen Teilers zweier positiver ganzer Zahlen mittels Primfaktorzerlegung antreten. Das Ergebnis sei vorweggenommen: 5:0 für den Euklidischen Algorithmus.

Wir schreiben $\mathbb{N}$ für die Menge der natürlichen Zahlen ohne Null und $\mathbb{N}_0$ für die Menge der natürlichen Zahlen mit Null.

2. RSA-Verfahren

Das 1978 von R. Rivest, A. Shamir und L. Adleman entwickelte RSA-Verfahren (vgl. Rivest et al. (1978)) ermöglicht es, dass der Schlüssel zum Verschlüsseln von Nachrichten öffentlich bekanntgegeben, die verschlüsselte Nachricht aber trotzdem nur vom beabsichtigten Empfänger entschlüsselt werden kann. Wie funktioniert das?

- Der Empfänger gibt zwei sehr große natürliche Zahlen n und e öffentlich bekannt.
- Der Sender will dem Empfänger eine natürliche Zahl a, die kleiner als n ist, verschlüsselt mitteilen.
- Dazu berechnet er den Rest b von a^e nach Division durch n:

  $$b := a^e \bmod n\,.$$

  Wie üblich schreiben wir statt „der Rest der natürlichen Zahl z nach Division durch n" kurz $z \bmod n$ (sprich: z modulo n).
- Der Empfänger erhält die Zahl b und macht fast dasselbe wie der Sender: er potenziert und berechnet dann den Rest nach Division durch n. Der Exponent der Potenz ist aber nicht e, sondern eine geeignete andere natürliche Zahl d. So erhält er die ursprüngliche Nachricht a :

  $$b^d \bmod n = a\,.$$

Das Zahlenpaar (e,n) heißt *Chiffrierschlüssel* und wird öffentlich bekannt gegeben. Das Paar (d,n) heißt *Dechiffrierschlüssel*, die Zahl d ist nur dem Empfänger bekannt.

Die Zahlen e, n und d werden vom Empfänger so gewählt bzw. berechnet:

- Der Empfänger wählt zwei sehr große, verschiedene Primzahlen p und q und berechnet die Produkte $n := p \cdot q$ und $m := (p-1) \cdot (q-1)$.
- Dann wählt er e so, dass

  $$\mathrm{ggT}(e, m) = 1$$

  ist. Mit $\mathrm{ggT}(e, m)$ bezeichnen wir die größte ganze Zahl, die sowohl e als auch m teilt.
- Schließlich berechnet er ganze Zahlen c und d so, dass

  $$m \cdot c + e \cdot d = 1$$

  ist.

Es stellen sich nun die folgenden drei Fragen:

1. Die Zahl n ist bekannt. Warum kann nicht jede/r ihre Primfaktoren p und q berechnen (und dann wie oben auch die Zahl d berechnen und die Nachricht entschlüsseln)?
2. Wenn e, p und q bekannt sind, wie werden dann die ganzen Zahlen c und d mit $m \cdot c + e \cdot d = 1$ berechnet?
   Oder: Wie findet man eine ganzzahlige Lösung einer linearen Gleichung mit zwei Unbekannten und ganzzahligen Koeffizienten?
3. Warum ist für alle natürlichen Zahlen a, die kleiner als n sind, der Rest von

   $$(a^e)^d = a^{e \cdot d} = a^{1-m \cdot c} = a \cdot (a^m)^{-c} = a \cdot (a^{(p-1)\cdot(q-1)})^{-c}$$

   nach Division durch $n = p \cdot q$ gleich a ?

Wir beantworten diese Fragen in den nächsten Abschnitten.

3. Faktorisierung großer Zahlen

Jede ganze Zahl, die größer als 2 ist, ist Produkt von Primzahlen. Diese Primzahlen sind bis auf die Reihenfolge eindeutig bestimmt und heißen *Primfaktoren* der gegebenen Zahl. Die Berechnung der Primfaktoren ist auch für sehr große Zahlen theoretisch immer möglich, praktisch aber auch von den leistungsfähigsten Computern nicht in vernünftiger Zeit durchführbar. Der von 1991 bis 2007 laufende Wettbewerb *RSA Factoring Challenge* bot Preisgelder (bis zu 200.000 USD) für die Berechnung der Primfaktoren von speziellen natürlichen Zahlen mit 100 bis 617 Dezimalstellen. Von diesen sogenannten RSA-Zahlen konnten bisher nur die Primfaktoren der Zahlen mit höchstens 232 Dezimalstellen berechnet werden (vgl. RSA Factoring Challenge (2013)). Die Berechnung für die Zahl RSA-768 mit 768 Binär- bzw. 232 Dezimalziffern gelang unter Verwendung von mehreren hundert Computern in rund zweieinhalb Jahren; die Berechnung mit einem handelsüblichen Computer (mit 2.2 GHz-Prozessor) hätte rund 1500 Jahre gedauert (vgl. Kleinjung et al. (2010)). Die Primfaktoren der Zahl RSA-1024

$$\begin{array}{l}
13506641086599522334960321627880596993888147560566702752448514385152651 \\
06048595338339402871505719094417982072821644715513736804197039641917430 \\
46496589274256239341020864383202110372958725762358509643110564073501508 \\
18751067659462920556368552947521350085287941637732853390610975054433499 \\
9811150056977236890927563
\end{array}$$

mit 1024 Binärziffern bzw. 309 Dezimalziffern können mit heutigen Methoden nicht in diesem Jahrhundert berechnet werden. Einige für den Schulunterricht nützliche Überlegungen zum Thema Rechenzeit für die Berechnung von Primfaktoren sind in Schulz & Witten (2010) beschrieben.

4. Der Euklidische Algorithmus

Addition, Multiplikation und Division mit Rest von Zahlen auch der Größe von RSA-1024 sind am Computer leicht durchführbar und brauchen wenig Zeit. Auch der größte gemeinsame Teiler $\mathrm{ggT}(x,y)$ zweier natürlicher Zahlen x und y kann mit dem Euklidischen Algorithmus sehr schnell berechnet werden:

1. Solange x und y nicht gleich sind, ersetze die größere der zwei Zahlen durch die Differenz der größeren und der kleineren.
2. Wenn $x = y$ ist, dann ist $\mathrm{ggT}(x,y) = x$.

Weil die größere der zwei natürlichen Zahlen im ersten Fall immer kleiner wird, tritt der zweite Fall nach endlich vielen Schritten ein.

Diesem Verfahren liegt die folgende einfache Aussage für zwei natürliche Zahlen x und y zugrunde:

$$\mathrm{ggT}(x,y) = \mathrm{ggT}(x-y,y)\,.$$

Alle gemeinsamen Teiler von x und y sind nämlich auch Teiler von $x-y$ und $x+y$, denn: Ist t ein Teiler von x und von y, dann gibt es ganze Zahlen u und v so, dass $u \cdot t = x$ und $v \cdot t = y$ ist. Durch „Herausheben" erhalten wir $x-y = u\cdot t - v\cdot t = (u-v)\cdot t$ und $x+y = u\cdot t + v\cdot t = (u+v)\cdot t$, also teilt t auch $x-y$ und $x+y$. Daher ist jeder gemeinsame Teiler von x und y auch ein gemeinsamer Teiler von x und $x-y$, umgekehrt ist jeder gemeinsame Teiler von $x-y$ und y auch ein gemeinsamer Teiler von y und $x = y + (x-y)$. Also müssen auch die größten gemeinsamen Teiler von x und y sowie von x und $x-y$ gleich sein.

Das obige Verfahren führt die Berechnung von $\mathrm{ggT}(x,y)$ auf die Berechnung des ggT der kleineren Zahlen x und $x-y$ zurück. Diese Vereinfachung führt man solange aus, bis die zwei Zahlen gleich sind. Dieses Verfahren ist ein gutes Beispiel für die folgende wichtige Strategie zum Problemlösen: „Wenn du ein Problem nicht lösen kannst, ersetze es durch ein einfacheres Problem mit der gleichen Lösung". Auch der Gauß-Algorithmus zum Lösen von Systemen linearer Gleichungen wendet diese Strategie an.

Wenn die Zahl x viel größer als y ist, wird in diesem Verfahren y so lange von x subtrahiert, bis die Differenz kleiner als y ist. Also: x wird mit Rest durch y dividiert und durch den Rest nach dieser Division ersetzt. Wir erhalten so eine schnellere Variante des gerade besprochenen Verfahrens:

1. Solange der Rest von x nach Division durch y nicht 0 ist, ersetze die Zahl x durch ihren Rest nach Division durch y (also durch r so, dass $x = s\cdot y + r$ und $0 \le r < y$ ist).
2. Wenn y ein Teiler von x ist, dann ist $y = \mathrm{ggT}(x,y)$.

Man könnte diese Variante auch direkt aus

$$\mathrm{ggT}(x,y) = \mathrm{ggT}(x - s\cdot y, y)\,.$$

ableiten.

5. Ganzzahlige lineare Gleichungen in zwei Unbekannten

Wir betrachten nun die folgende Aufgabe: Gegeben sind ganze Zahlen u,v,w. Finde ein Paar (x,y) von ganzen Zahlen so, dass

$$u\cdot x + v\cdot y = w$$

ist.
Wenn eine Lösung (x,y) existiert, dann ist jeder gemeinsame Teiler von u und v auch ein Teiler von $u\cdot x + v\cdot y = w$. Falls eine Lösung existiert, muss also $\mathrm{ggT}(u,v)$ ein Teiler von w sein.

Man rechnet direkt nach: Wenn (x_1,y_1) bzw. (x_2,y_2) eine Lösung von $u\cdot x + v\cdot y = w_1$ bzw. $u\cdot x + v\cdot y = w_2$ ist, dann ist $(x_1,y_1) - s\cdot(x_2,y_2) = (x_1 - s\cdot x_2, y_1 - s\cdot y_2)$ eine Lösung von $u\cdot x + v\cdot y = w_1 - s\cdot w_2$.

Wenn $w = u$ oder $w = v$ ist, sind je eine Lösung unmittelbar ersichtlich:

- Eine Lösung von $u \cdot x + v \cdot y = u$ ist $(1,0)$.
- Eine Lösung von $u \cdot x + v \cdot y = v$ ist $(0,1)$.

Wir dividieren nun u mit Rest durch v: $u = s \cdot v + r,\ 0 \leq r < |v|$. Wir erhalten dadurch auch eine Lösung von

$$u \cdot x + v \cdot y = u - s \cdot v$$

nämlich $(1,0) - s \cdot (0,1) = (1,-s)$. Wir wiederholen diese Vorgehensweise für die jeweils letzten zwei Gleichungen, und erhalten schließlich die Lösung der Gleichung

$$u \cdot x + v \cdot y = \mathrm{ggT}(u,v),$$

weil wir auf der rechten Seite des Gleichheitszeichens den Euklidischen Algorithmus ausgeführt haben. Dieses Verfahren ist der *erweiterte Euklidische Algorithmus* (vgl. Pauer (2005), Kap. 9).

Diese Vorgehensweise verwendet die folgende Strategie: „Wenn du ein Problem nicht lösen kannst, dann löse zunächst einige einfache Probleme und versuche, aus deren Lösungen die des ursprünglichen Problems zusammenzubauen."

Ist

$$u \cdot x + v \cdot y = \mathrm{ggT}(u,v),$$

dann ist für alle ganzen Zahlen m auch

$$u \cdot (x - m \cdot v) + v \cdot (y + m \cdot u) = u \cdot x + v \cdot y = \mathrm{ggT}(u,v).$$

Durch eventuelle Addition eines geeigneten Vielfachen von v kann man also immer erreichen, dass die erste Komponente der Lösung eine nicht negative ganze Zahl ist.

Beispiel 5.1 *Finde ganze Zahlen x und y so, dass x positiv und*

$$19 \cdot x + 11 \cdot y = 1$$

ist. Wir lösen nacheinander die Gleichungen

$$\begin{array}{ll}
19 \cdot x + 11 \cdot y = 19 & \textit{eine Lösung: } (1,0) \\
19 \cdot x + 11 \cdot y = 11 & \textit{eine Lösung: } (0,1) \\
19 \cdot x + 11 \cdot y = 19 - 11 = 8 & \textit{eine Lösung: } (1,-1) \\
19 \cdot x + 11 \cdot y = 11 - 8 = 3 & \textit{eine Lösung: } (-1,2) \\
19 \cdot x + 11 \cdot y = 8 - 2 \cdot 3 = 2 & \textit{eine Lösung: } (3,-5) \\
19 \cdot x + 11 \cdot y = 3 - 2 = 1 & \textit{eine Lösung: } (-4,7)
\end{array}$$

Da -4 *negativ ist, ersetzen wir* $(-4,7)$ *durch* $(-4+11, 7-19) = (7,-12)$, *um eine Lösung mit positiver erster Komponente zu erhalten.*

Eine verbreitete Schreibweise für die Durchführung des erweiterten Euklidischen Algorithmus wie im letzten Beispiel ist:

$$\begin{array}{c|cc|cccc}
 & 19 & 11 & 8 & 3 & 2 & 1 \\
\hline
x & 1 & 0 & 1 & -1 & 3 & -4 \\
y & 0 & 1 & -1 & 2 & -5 & 7
\end{array}$$

◇

Wenn w von $\mathrm{ggT}(u,v)$ geteilt wird, es also eine ganze Zahl t mit $w = t \cdot \mathrm{ggT}(u,v)$ gibt, hat die Gleichung

$$u \cdot x + v \cdot y = w$$

eine Lösung. Man berechnet zuerst (x', y') so, dass $u \cdot x' + v \cdot y' = \mathrm{ggT}(u,v)$ ist. Dann ist $(t \cdot x', t \cdot y')$ eine Lösung von $u \cdot x + v \cdot y = w$.

Aus dem erweiterten Euklidischen Algorithmus folgt das *Euklidische Lemma*:

Teilt eine Primzahl p ein Produkt ganzer Zahlen, so teilt p auch einen der Faktoren.

Für den Beweis nehmen wir an, dass p das Produkt $a \cdot b$ zweier ganzer Zahlen teilt, aber p die Zahl a nicht teilt. Da p eine Primzahl ist und a nicht teilt, ist $\mathrm{ggT}(a,p) = 1$. Es gibt daher ganze Zahlen x und y mit

$$p \cdot x + a \cdot y = \mathrm{ggT}(a,p) = 1\,.$$

Wir multiplizieren auf beiden Seiten des Gleichheitszeichens mit b und erhalten

$$p \cdot x \cdot b + a \cdot b \cdot y = b\,.$$

Da p die Zahl $a \cdot b$ teilt, gibt es eine ganze Zahl t mit $p \cdot t = a \cdot b$. Einsetzen und Herausheben liefert

$$p \cdot (x \cdot b + t \cdot y) = b\,.$$

Daher teilt p die Zahl b $\square$

Mit Hilfe des Euklidischen Lemmas kann die Eindeutigkeit der Primfaktorzerlegung leicht gezeigt werden (vgl. Winkler (2007), Kap. 2.3).

6. Rechnen mit Restklassen

Für eine natürliche Zahl $n > 2$ betrachten wir die Menge

$$\mathbb{Z}_n := \{0, 1, \ldots, n-1\}\,.$$

Wir führen nun in $\mathbb{Z}_n$ zwei Rechenoperationen ein. Da die übliche Summe von zwei Elementen in $\mathbb{Z}_n$ im Allgemeinen nicht mehr ein Element von $\mathbb{Z}_n$ ist, definieren wir die Addition in $\mathbb{Z}_n$ durch:

Für alle $a, b \in \mathbb{Z}_n$ ist $a +_n b := (a+b) \bmod n$.

Insbesondere kann in $\mathbb{Z}_n$ subtrahiert werden, da für jedes $a \in \mathbb{Z}_n$ auch $n - a \in \mathbb{Z}_n$ ist und

$$a +_n (n-a) = n \bmod n = 0$$

ist.
Die Multiplikation in $\mathbb{Z}_n$ definieren wir durch:

Für alle $a, b \in \mathbb{Z}_n$ ist $a \cdot_n b := (a \cdot b) \bmod n$.

Für diese Addition und Multiplikation gelten (fast) die gleichen Rechenregeln wie für das Rechnen mit ganzen Zahlen.

Kann man in $\mathbb{Z}_n$ auch dividieren? Für welche Zahlen $0 \neq a \in \mathbb{Z}_n$ gibt es eine Zahl $b \in \mathbb{Z}_n$ mit $a \cdot_n b = 1$? Anders formuliert: Für welche $0 \neq a \in \mathbb{Z}_n$ besitzt die Gleichung

$$a \cdot x + n \cdot y = 1$$

eine ganzzahlige Lösung? Nach Abschnitt 5 gibt es eine derartige Lösung genau dann, wenn $\mathrm{ggT}(a,n) = 1$ ist.

Falls p ein Primzahl und $0 \neq a$ kleiner als p ist, gilt immer $\mathrm{ggT}(a,p) = 1$, und daher kann in $\mathbb{Z}_p$ durch alle Elemente $\neq 0$ dividiert werden. Zum Dividieren in $\mathbb{Z}_p$ verwendet man den erweiterten Euklidischen Algorithmus.

Beispiel 6.1 *Wir berechnen $x \in \mathbb{Z}_{17}$ mit $10 \cdot_{17} x = 1$. Wir suchen also eine ganzzahlige Lösung der Gleichung*

$$10 \cdot x + 17 \cdot y = 1\,.$$

Da $\mathrm{ggT}(10,17) = 1$ *ist, können wir eine Lösung dieser Gleichung wie in Abschnitt 5 berechnen und erhalten zum Beispiel* $(12,-7)$. *In* $\mathbb{Z}_{17}$ *ist also* $10^{-1} = 12$, *weil* $10 \cdot_{17} 12 = 120 \bmod 17 = 1$ *ist.*

Dividiert man in $\mathbb{Z}_{17}$ die Zahl 4 durch 10, erhält man

$$4 :_{17} 10 = 4 \cdot_{17} 10^{-1} = 4 \cdot_{17} 12 = 48 \bmod 17 = 14\,.$$

$\diamond$

Bemerkung 6.2 *In $\mathbb{Z}_p$ kann wie mit rationalen oder reellen Zahlen gerechnet werden, insbesondere ist der Gauß-Algorithmus für Systeme linearer Gleichungen anwendbar. Das Rechnen in $\mathbb{Z}_p$ ist für die Kodierungstheorie, die Kryptographie und des „schnelle Rechnen“ mit ganzen Zahlen von großer Bedeutung.*

7. Der kleine Satz von Fermat

Wir verwenden den Binomischen Lehrsatz:
Für zwei ganze Zahlen x, y und eine natürliche Zahl n ist

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}\,,$$

dabei ist

$$\binom{n}{k} := \frac{n!}{k! \cdot (n-k)!}\,,$$

mit $n! := n \cdot (n-1) \cdot \ldots \cdot 1$ (n Fakultät) und $0! := 1$.

Ist p ein Primzahl und $1 \le k \le p-1$, so kann p nicht $k!$ teilen, denn sonst würde p bereits einen der Faktor $k < p$ teilen (siehe Abschnitt 5). Daher ist bereits

$$\frac{(p-1) \cdot (p-2) \cdot \ldots \cdot (p-k+1)}{k!}$$

eine natürliche Zahl und $\binom{p}{k}$ ist für $1 \le k \le p-1$ durch p teilbar.

Wegen

$$(x+y)^p = \sum_{k=0}^{p} \binom{p}{k} x^k y^{p-k} = x^p + y^p + \sum_{k=1}^{p-1} \binom{p}{k} x^k y^{p-k}\,.$$

und da $\binom{p}{k}$ für $1 \le k \le p-1$ durch p teilbar ist, erhalten wir

$$(x+y)^p = (x^p + y^p) \bmod p\,. \tag{1}$$

Der kleine Satz von Fermat besagt, dass für jede Primzahl p und jede natürliche Zahl a

$$a^p \bmod p = a \bmod p \tag{2}$$

ist.

Wir können diesen Satz mittels Induktion nach a beweisen.

Offensichtlich ist $0^p = 0$ und daher auch $0^p = 0 \bmod p$. Damit ist der Induktionsanfang gezeigt. Die Aussage (2) gelte nun für eine Zahl $a \in \mathbb{N}_0$. Wir zeigen, dass sie dann auch für $a+1$ gilt:

$$\begin{aligned}(a+1)^p \bmod p &= \left[\text{nach (1)}\right] = a^p + 1^p \bmod p \\ &= \left[\text{Induktionsvoraussetzung}\right] = a+1 \bmod p\,.\end{aligned}$$

□

Wenn $a \bmod p \neq 0$ ist (also a nicht von p geteilt wird), erhalten wir aus (2):

$$a^{p-1} \bmod p = 1\,.$$

Wir sind nun in der Lage, die Frage 3 von Abschnitt 2 zu beantworten. Zunächst erhalten wir die folgenden zwei Aussagen

$$\begin{aligned}a^{e\cdot d} \bmod p &= a^{1-(p-1)(q-1)\cdot c} \bmod p = \\ &= \left.\begin{cases} a\cdot(\underbrace{a^{(p-1)}}_{=1\in\mathbb{Z}_p})^{-c\cdot(q-1)} \bmod p = a \bmod p & \text{, wenn } a \bmod p \neq 0 \text{ ist} \\ a\cdot a^{-(p-1)(q-1)\cdot c} \bmod p = 0 & \text{, wenn } a \bmod p = 0 \text{ ist}\end{cases}\right\} = a \bmod p\end{aligned}$$

und analog

$$\begin{aligned}a^{e\cdot d} \bmod q &= a^{1-(p-1)(q-1)\cdot c} \bmod q = \\ &= \left.\begin{cases} a\cdot(\underbrace{a^{(q-1)}}_{=1\in\mathbb{Z}_q})^{-c\cdot(p-1)} \bmod p = a \bmod q & \text{, wenn } a \bmod q \neq 0 \text{ ist} \\ a\cdot a^{-(p-1)(q-1)\cdot c} \bmod q = 0 & \text{, wenn } a \bmod q = 0 \text{ ist}\end{cases}\right\} = a \bmod q\,.\end{aligned}$$

Dies bedeutet, dass $a^{e\cdot d} - a$ sowohl von p als auch von q geteilt wird. Es gibt also ganze Zahlen s, t mit

$$s\cdot p = t\cdot q = a^{e\cdot d} - a\,.$$

Weil p und q verschieden sind und die Primzahl p das Produkt $t\cdot q$ teilt, muss p ein Teiler von t sein (siehe Abschnitt 5). Somit ist

$$t\cdot q = (p\cdot u)\cdot q = n\cdot u$$

für eine ganze Zahl u. Also teilt n die Zahl $t\cdot q = a^{e\cdot d} - a$ und

$$a^{e\cdot d} \bmod n = a \bmod n\,.$$

□

8. Euklidischer Algorithmus versus ggT-Berechnung mittels Primfaktorzerlegung

Die Berechnung des ggT ist für das Rechnen mit rationalen Zahlen wichtig. Nach jeder Rechenoperation sollten Zähler und Nenner des Ergebnisses bestmöglich – also durch ihren ggT – gekürzt werden, um alle auftretenden Zahlen möglichst klein zu halten. Häufig wird im Schulunterricht der ggT von zwei positiven Zahlen nicht mit dem Euklidischen Algorithmus, sondern mithilfe der Primfaktorzerlegung als Produkt der gemeinsamen Primfaktoren der zwei Zahlen berechnet.

Vergleichen wir die zwei Verfahren an Hand des Beispiels

Kürze $\dfrac{78}{117}$.

Dazu berechnen wir den ggT von 117 und 78 mittels Primfaktorzerlegung und mit dem Euklidischem Algorithmus:

Primfaktorzerlegung	Euklidischer Algorithmus
$78 : 2 = 39$	$117 - 78 = 39$
$39 : 2 \notin \mathbb{N}$	$78 - 39 = 39$
$39 : 3 = 13$	$\Rightarrow \mathrm{ggT}(78, 117) = \mathrm{ggT}(39, 39) = 39$
$13 : 3 \notin \mathbb{N}$, fertig (weil $5^2 > 13$ ist)	
$\Rightarrow 78 = 2 \cdot 3 \cdot 13$	
$117 : 2 \notin \mathbb{N}$	
$117 : 3 = 39$	
$39 : 3 = 13$, fertig (weil $5^2 > 13$ ist)	
$\Rightarrow 117 = 3 \cdot 3 \cdot 13$	
$\Rightarrow \mathrm{ggT}(78, 117) = 3 \cdot 13 = 39$	

Daher ist $\dfrac{78}{117} = \dfrac{2}{3}$.

Welche Argumente gibt es, um zwei Algorithmen gegeneinander abzuwägen? Es liegt nahe, dafür die folgenden Kriterien zu verwenden:

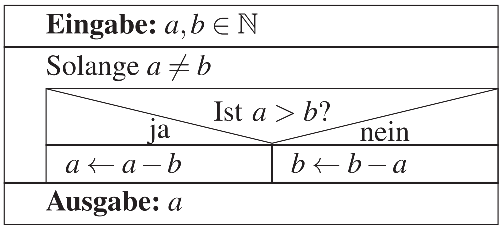
1. Welches der zwei Verfahren ist einfacher zu erklären, verlangt weniger Fachbegriffe und für welches ist die Korrektheit mit einfacheren Mitteln zu beweisen?
2. Welches der zwei Verfahren ist rechnerisch effizienter, d. h. benötigt weniger Rechenaufwand, um das Ergebnis zu berechnen?
3. Welches der zwei Verfahren ist leichter zu programmieren?
4. Welches der zwei Verfahren kann auf andere im Schulunterricht bedeutsame Situationen angewendet werden?
5. Welches der zwei Verfahren vermittelt eine grundlegende Strategie?

Bei all diesen Fragen liegt der Euklidische Algorithmus weit voran!

1. Die Theorie für die Berechnung des ggT mit dem Euklidischen Algorithmus ist sehr einfach (siehe Abschnitt 4) und auch von Schüler/innen der Sekundarstufe 1 nachvollziehbar. Hingegen braucht man nicht nur die Existenz, sondern auch die Eindeutigkeit der Primfaktorzerlegung, um zu zeigen, dass das Produkt der gemeinsamen Primfaktoren die *größte* Zahl ist, die beide teilt.
2. Kein Computeralgebrasystem berechnet den ggT mittels Primfaktorzerlegung. Die Berechnung des ggT von Zahlen in der Größenordnung von RSA-1024 ist mit dem Euklidischen Algorithmus rasch erledigt, die Primfaktorzerlegung einer einzigen solchen Zahl würde aber Jahrhunderte dauern. Bei der Berechnung des ggT mittels Primfaktorzerlegung werden zwei viel aufwändigere Probleme, nämlich die Zerlegung der zwei Zahlen in Primfaktoren gelöst d. h. es wird unnötigerweise viel mehr ausgerechnet als benötigt wird.

3. Die Programmierung und Implementierung des Euklidischen Algorithmus ist eine leichte Übungsaufgabe und kann im Schulunterricht durchgeführt werden (siehe Struktogramm); um ein Programm für die Berechnung des ggT mittels Primfaktorzerlegung zu schreiben, müsste zuerst eine Liste der Primzahlen erzeugt werden.

ggT



4. Der Euklidische Algorithmus ist auf das Rechnen mit Polynomen direkt übertragbar. Damit können rationale Funktionen gekürzt werden, auch wenn die Grade von Zähler und Nenner groß sind. Der Primfaktorzerlegung von ganzen Zahlen entspricht die Zerlegung von Polynomen in irreduzible Faktoren, diese kann bei Polynomen mit reellen oder komplexen Koeffizienten im allgemeinen nicht mehr ermittelt werden.

5. Die beim Euklidischen Algorithmus verwendete Strategie „Wenn du ein Problem nicht lösen kannst, ersetze es durch ein einfacheres Problem mit der gleichen Lösung“ wird beim Lösen von Gleichungssystemen als „Strategie des erlaubten Umformens“ wieder verwendet: „Gehe von einem Gleichungssystem zu einem einfacheren mit gleicher Lösungsmenge über“.

Zu bedenken ist auch, dass für Schüler/innen der Sekundarstufe 2 das Verständnis für das RSA-Verfahren erschwert wird, wenn in der Sekundarstufe 1 eine „falsche Fährte“ gelegt wurde, indem die Primfaktorzerlegung als etwas Einfaches dargestellt wurde.

Literatur

Kleinjung, T., et al.: Factorization of a 768-bit RSA modulus, *Proceedings of the 30th annual conference on Advances in cryptology*, Santa Barbara, CA, USA (2010), 333–350.

Pauer, F.: Division mit Rest – der heimliche Hauptsatz der Algebra. *Didaktikheft der Österreichischen Mathematischen Gesellschaft*, Nr. 37 (2005), 100–111.

Rivest, R. L., Shamir, A. and Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21/2 (1978), 120–126.

RSA Factoring Challenge. *Wikipedia: The Free Encyclopedia.* Version vom 31. Oktober 2013, 10:45 Uhr, abrufbar unter: `http://en.wikipedia.org/w/index.php?title=RSA_Factoring_Challenge&oldid=569179861`.

Schulz, R.-H. und H. Witten.: Zeit-Experimente zur Faktorisierung. *LOG IN*, Heft Nr. 166/167 (2010), 107–114.

Winkler, R.: Sinn und Unsinn des Rechnens im Mathematikunterricht. *Didaktikheft der Österreichischen Mathematischen Gesellschaft*, Nr. 39 (2007), 155–165.