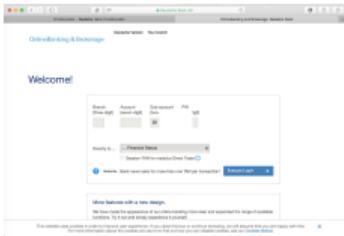


**Woche 9:** 19. Juni 2024

**Thema:** *Primzahlen und Modulare Arithmetik*

## **9.1 Einleitung**

# Beispiel: Internetkommunikation



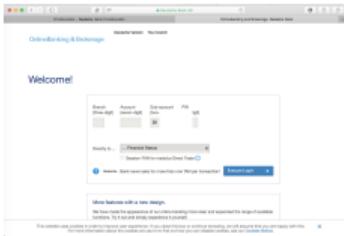
## Beispiel: Internet Banking.

Herr Euler möchte dem St. Petersburgischen Brückenfond Geld spenden.

Er ruft die *Überweisungen*-Webseite der *Abelschen Bank Gruppe* auf.

Dort trägt er die *IBAN Nummer* des Fonds ein und überweist den Betrag.

# Beispiel: Internetkommunikation



## Beispiel: Internet Banking.

Herr Euler möchte dem St. Petersburgischen Brückenfond Geld spenden.

Er ruft die *Überweisungen*-Webseite der *Abelschen Bank Gruppe* auf.

Dort trägt er die *IBAN Nummer* des Fonds ein und überweist den Betrag.

## Herausforderungen.

- Eingabe der IBAN....
- Abhören der Verbindung.

# Typische Eingabesysteme

1-Finger Adler-Such-System.



8, 5-Finger-100.000 Zeichen/ms-System



Problem.

- Tippfehler: falsche Taste getroffen

Problem.

- Tippfehler: vor allem Vertauschung benachbarter Ziffern.

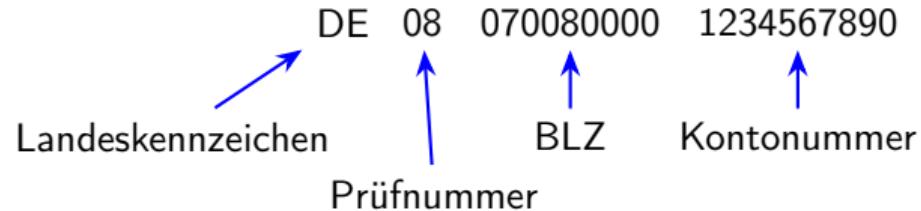
Prüfziffer.

Anders als bei Passwörtern können diese Fehler zur Überweisung auf ein falsches Konto führen.

# Prüfziffern

## Prüfziffern.

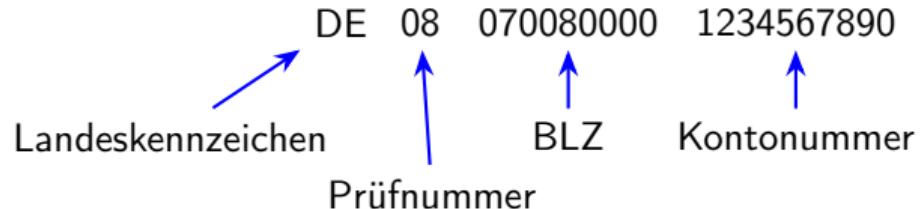
Um Fehler dieser Art zu vermeiden, werden IBAN Nummern mit einer Prüfsumme versehen.



# Prüfziffern

## Prüfziffern.

Um Fehler dieser Art zu vermeiden, werden IBAN Nummern mit einer Prüfsumme versehen.



**Frage.** Wie entwirft man gute Prüfsummen?

**Gesucht.** Prüfsummen, die typische Tippfehler erkennen.

D.h. vertauschen zweier Ziffern oder Fehler in genau einer Ziffer soll eine andere Prüfsumme liefern.

# Verschlüsselung

Verschlüsselung der Internetkommunikation.

Herr Euler möchte die Nachricht an seine Bank verschlüsseln.

Nachricht. „überweise 1 Euro an St. Petersburg“

Schlüssel. „ein schlüssel den keiner kennt außer meiner Bank“

Verschlüsselte Nachricht: Nachricht 'XOR' Schlüssel:

„325nksjadfnaso[1289yBVB-FCB-5:1-519ahfaewhgfas2356“

# Verschlüsselung

Verschlüsselung der Internetkommunikation.

Herr Euler möchte die Nachricht an seine Bank verschlüsseln.

Nachricht. „überweise 1 Euro an St. Petersburg“

Schlüssel. „ein schlüssel den keiner kennt außer meiner Bank“

Verschlüsselte Nachricht: Nachricht 'XOR' Schlüssel:

„325nksjadfnaso[1289yBVB-FCB-5:1-519ahfaewhgfahgfas2356“

Entschlüsselte Nachricht: Verschlüsselt 'XOR' Schlüssel.

Schlüsselaustausch.

- Möglichkeit 1. Er und seine Bank tauschen vorher einen Schlüssel aus, den nur sie beide kennen. ↵ *symmetrische Verschlüsselung*

# Verschlüsselung

Verschlüsselung der Internetkommunikation.

Herr Euler möchte die Nachricht an seine Bank verschlüsseln.

**Nachricht.** „überweise 1 Euro an St. Petersburg“

**Schlüssel.** „ein schlüssel den keiner kennt außer meiner Bank“

**Verschlüsselte Nachricht:** Nachricht 'XOR' Schlüssel:

„325nksjadfnaso[1289yBVB-FCB-5:1-519ahfaewhgfahgfas2356“

**Entschlüsselte Nachricht:** Verschlüsselt 'XOR' Schlüssel.

Schlüsselaustausch.

- **Möglichkeit 1.** Er und seine Bank tauschen vorher einen Schlüssel aus, den nur sie beide kennen. ↵ *symmetrische Verschlüsselung*
- **Möglichkeit 2.** Oft ist das nicht möglich, z.B. bei EMail-Verschlüsselung.  
Wir suchen also ein Verfahren, mit dem Herr Euler die Nachricht verschlüsseln kann, ohne dass die beiden vorher einen Schlüssel austauschen müssen. ↵ *asymmetrische Verschlüsselung*

# Public-Key Verfahren

## Public-Key Verfahren.

- Es gibt zwei Schlüssel,  $K_{pub}, K_{private}$ .
- Herr Euler verschlüsselt seine Nachricht mit dem Schlüssel  $K_{pub}$ , den die Bank öffentlich verbreitet.
- Zum Entschlüsseln braucht man aber den anderen Schlüssel  $K_{private}$ , den die Bank geheim hält.

# Public-Key Verfahren

## Public-Key Verfahren.

- Es gibt zwei Schlüssel,  $K_{pub}, K_{private}$ .
- Herr Euler verschlüsselt seine Nachricht mit dem Schlüssel  $K_{pub}$ , den die Bank öffentlich verbreitet.
- Zum Entschlüsseln braucht man aber den anderen Schlüssel  $K_{private}$ , den die Bank geheim hält.

## Wie entwirft man solche Verfahren?

**Antwort.** Mit Hilfe der Zahlentheorie.

Genauer: modulare Arithmetik und Primzahlentheorie.

# Zero-Knowledge Proofs

**Pineingabe.** Am Geldautomat ebenso wie auf der Bank-Webseite ebenso wie überall geben wir dauernd unsere PIN, unser Passwort etc. in uns unbekannte Geräte ein.

Wir geben damit unsere PIN preis, obschon das eigentlich unnötig ist.

Die Bank muss eigentlich nicht unsere PIN wissen, sondern letztlich nur, dass wir die richtige kennen.

# Zero-Knowledge Proofs

**Pineingabe.** Am Geldautomat ebenso wie auf der Bank-Webseite ebenso wie überall geben wir dauernd unsere PIN, unser Passwort etc. in uns unbekannte Geräte ein.

Wir geben damit unsere PIN preis, obschon das eigentlich unnötig ist.

Die Bank muss eigentlich nicht unsere PIN wissen, sondern letztlich nur, dass wir die richtige kennen.

**Zero-Knowledge Proofs.** Besser wären Verfahren, in denen ich die Bank überzeugen kann, dass ich die PIN habe, ohne sie aber preisgeben zu müssen.

D.h. ich möchte der Bank beweisen können, dass ich die richtige PIN habe, ohne dass umgekehrt aus meinem Beweis die PIN abgeleitet werden kann oder dass jemand anderes meinen Beweis einfach nochmal vorlegen kann und damit Zugriff auf mein Konto bekommt.

## 9.2 Primzahlen

# Primzahlen

Definition. Seien  $a, b \in \mathbb{Z}$ .

$a$  teilt  $b$ , geschrieben  $a | b$ , wenn es ein  $k \in \mathbb{Z}$  gibt mit  $k \cdot a = b$ .

# Primzahlen

Definition. Seien  $a, b \in \mathbb{Z}$ .

$a$  teilt  $b$ , geschrieben  $a | b$ , wenn es ein  $k \in \mathbb{Z}$  gibt mit  $k \cdot a = b$ .

Größter gemeinsamer Teiler von  $a$  und  $b$ :

$$\text{ggT}(a, b) := \max\{k \in \mathbb{N} : k \text{ teilt } a \text{ und } k \text{ teilt } b\}.$$

Kleinstes gemeinsames Vielfache von  $a$  und  $b$ :

$$\text{kgV}(a, b) := \min\{k \in \mathbb{N} : a \text{ teilt } k \text{ und } b \text{ teilt } k\}.$$

# Primzahlen

Definition. Seien  $a, b \in \mathbb{Z}$ .

$a$  teilt  $b$ , geschrieben  $a | b$ , wenn es ein  $k \in \mathbb{Z}$  gibt mit  $k \cdot a = b$ .

Größter gemeinsamer Teiler von  $a$  und  $b$ :

$$\text{ggT}(a, b) := \max\{k \in \mathbb{N} : k \text{ teilt } a \text{ und } k \text{ teilt } b\}.$$

Kleinstes gemeinsames Vielfache von  $a$  und  $b$ :

$$\text{kgV}(a, b) := \min\{k \in \mathbb{N} : a \text{ teilt } k \text{ und } b \text{ teilt } k\}.$$

Jede Zahl  $a \in \mathbb{Z}$  besitzt mindestens zwei positive Teiler:  $1$  und  $a$  bzw.  $-a$ .

# Primzahlen

Definition. Seien  $a, b \in \mathbb{Z}$ .

$a$  teilt  $b$ , geschrieben  $a | b$ , wenn es ein  $k \in \mathbb{Z}$  gibt mit  $k \cdot a = b$ .

Größter gemeinsamer Teiler von  $a$  und  $b$ :

$$\text{ggT}(a, b) := \max\{k \in \mathbb{N} : k \text{ teilt } a \text{ und } k \text{ teilt } b\}.$$

Kleinstes gemeinsames Vielfache von  $a$  und  $b$ :

$$\text{kgV}(a, b) := \min\{k \in \mathbb{N} : a \text{ teilt } k \text{ und } b \text{ teilt } k\}.$$

Jede Zahl  $a \in \mathbb{Z}$  besitzt mindestens zwei positive Teiler:  $1$  und  $a$  bzw.  $-a$ .

Primzahlen.

Zahlen  $p \geq 2$  für die  $1$  und  $p$  die einzigen positiven Teiler sind.

Beachte:  $1$  ist keine Primzahl.

# Primzahlen

Theorem (Fundamentalsatz der Arithmetik).

Jede Zahl  $n \in \mathbb{N}$  mit  $n \geq 2$  lässt sich eindeutig als Produkt von Primzahlen darstellen:

$$n = p_1^{e_1} \cdot p_2^{e_2} \cdots \cdot p_k^{e_k}$$

wobei  $p_1 < p_2 < \cdots < p_k$  Primzahlen sind und  $e_1, e_2, \dots, e_k \in \mathbb{N}$ .

# Primzahlen

Theorem (Fundamentalsatz der Arithmetik).

Jede Zahl  $n \in \mathbb{N}$  mit  $n \geq 2$  lässt sich eindeutig als Produkt von Primzahlen darstellen:

$$n = p_1^{e_1} \cdot p_2^{e_2} \cdots \cdot p_k^{e_k}$$

wobei  $p_1 < p_2 < \cdots < p_k$  Primzahlen sind und  $e_1, e_2, \dots, e_k \in \mathbb{N}$ .

Primzahlen in der Kryptologie.

In den angesprochenen Verschlüsselungsverfahren werden sehr große Primzahlen (1000 Stellen und mehr) verwendet.

Wir müssen also erst einmal sicherstellen, dass es so große Primzahlen überhaupt gibt.

# Primzahlen

Theorem (Fundamentalsatz der Arithmetik).

Jede Zahl  $n \in \mathbb{N}$  mit  $n \geq 2$  lässt sich eindeutig als Produkt von Primzahlen darstellen:

$$n = p_1^{e_1} \cdot p_2^{e_2} \cdots \cdot p_k^{e_k}$$

wobei  $p_1 < p_2 < \cdots < p_k$  Primzahlen sind und  $e_1, e_2, \dots, e_k \in \mathbb{N}$ .

Primzahlen in der Kryptologie.

In den angesprochenen Verschlüsselungsverfahren werden sehr große Primzahlen (1000 Stellen und mehr) verwendet.

Wir müssen also erst einmal sicherstellen, dass es so große Primzahlen überhaupt gibt.

Satz. Es gibt unendlich viele Primzahlen.

# Beweis

Beweis durch Widerspruch.

Ang. es gäbe nur endlich viele Primzahlen:  $p_1, \dots, p_k$ .

Satz.

Es gibt unendlich viele Primzahlen.

# Beweis

Beweis durch Widerspruch.

Ang. es gäbe nur endlich viele Primzahlen:  $p_1, \dots, p_k$ .

Dann ist das Produkt  $p_1 \cdot p_2 \cdot \dots \cdot p_k$  wieder eine natürliche Zahl und wir setzen

$$n = 1 + \prod_{i=1}^k p_i. \quad (*)$$

Satz.

Es gibt unendlich viele Primzahlen.

# Beweis

Beweis durch Widerspruch.

Ang. es gäbe nur endlich viele Primzahlen:  $p_1, \dots, p_k$ .

Dann ist das Produkt  $p_1 \cdot p_2 \cdot \dots \cdot p_k$  wieder eine natürliche Zahl und wir setzen

$$n = 1 + \prod_{i=1}^k p_i. \quad (*)$$

Nach dem Fundamentalsatz, kann  $n$  als Produkt von Primzahlen geschrieben werden.

Da es nur die Primzahlen  $p_1, \dots, p_k$  gibt, gilt also

$$n = p_1^{e_1} \cdots p_k^{e_k}$$

wobei  $e_i \neq 0$  für ein  $1 \leq i \leq k$  gelten muss.

Satz.

Es gibt unendlich viele Primzahlen.

# Beweis

Beweis durch Widerspruch.

Ang. es gäbe nur endlich viele Primzahlen:  $p_1, \dots, p_k$ .

Dann ist das Produkt  $p_1 \cdot p_2 \cdot \dots \cdot p_k$  wieder eine natürliche Zahl und wir setzen

$$n = 1 + \prod_{i=1}^k p_i. \quad (*)$$

Nach dem Fundamentalsatz, kann  $n$  als Produkt von Primzahlen geschrieben werden.

Da es nur die Primzahlen  $p_1, \dots, p_k$  gibt, gilt also

$$n = p_1^{e_1} \cdots p_k^{e_k}$$

wobei  $e_i \neq 0$  für ein  $1 \leq i \leq k$  gelten muss.

Also folgt  $p_i | n$ .

Satz.

Es gibt unendlich viele Primzahlen.

# Beweis

Beweis durch Widerspruch.

Ang. es gäbe nur endlich viele Primzahlen:  $p_1, \dots, p_k$ .

Dann ist das Produkt  $p_1 \cdot p_2 \cdot \dots \cdot p_k$  wieder eine natürliche Zahl und wir setzen

$$n = 1 + \prod_{i=1}^k p_i. \quad (*)$$

Nach dem Fundamentalsatz, kann  $n$  als Produkt von Primzahlen geschrieben werden.

Da es nur die Primzahlen  $p_1, \dots, p_k$  gibt, gilt also

$$n = p_1^{e_1} \cdots p_k^{e_k}$$

wobei  $e_i \neq 0$  für ein  $1 \leq i \leq k$  gelten muss.

Also folgt  $p_i | n$ . Nach  $(*)$  gilt aber auch  $p_i | n - 1$ .

Da  $p_i \geq 2$ , führt dies zum Widerspruch.  $\square$

**Satz.**

Es gibt unendlich viele Primzahlen.

# Berechnen von Primzahlen

Wie berechnet man große Primzahlen?

Für die vorher beschriebenen Anwendungen muss man als erstes sehr große Primzahlen berechnen.

# Berechnen von Primzahlen

Wie berechnet man große Primzahlen?

Für die vorher beschriebenen Anwendungen muss man als erstes sehr große Primzahlen berechnen.

**Das Sieb des Eratosthenes.** Für eine obere Schranke  $n$  kann man die Primzahlen  $p \leq n$  wie folgt berechnen:

1. **for**  $i$  in  $[2, \sqrt{n}]$ :
2.     Wenn  $i$  noch nicht gestrichen, streiche alle Vielfachen  $2i, 3i, 4i, \dots$

Die Zahlen, die am Ende übrig bleiben, sind die Primzahlen  $p \leq n$ .

# Berechnen von Primzahlen

Wie berechnet man große Primzahlen?

Für die vorher beschriebenen Anwendungen muss man als erstes sehr große Primzahlen berechnen.

**Das Sieb des Eratosthenes.** Für eine obere Schranke  $n$  kann man die Primzahlen  $p \leq n$  wie folgt berechnen:

1. **for**  $i$  **in**  $[2, \sqrt{n}]$ :
2.     Wenn  $i$  noch nicht gestrichen, streiche alle Vielfachen  $2i, 3i, 4i, \dots$

Die Zahlen, die am Ende übrig bleiben, sind die Primzahlen  $p \leq n$ .

**Effizienz.** Das Sieb des Eratosthenes ist ein sehr einfaches Verfahren für kleine Werte von  $n$ .

Für größere Primzahlen ist es aber viel zu ineffizient.

## Berechnen von Primzahlen

Wie häufig kommen Primzahlen vor?

Für  $n \geq 1$  sei  $\pi(n)$  die Zahl der Primzahlen  $\leq n$ .

Primzahlensatz. Für alle  $n \in \mathbb{N}$  gilt

$$\pi(n) = (1 + o(1)) \cdot \frac{n}{\ln(n)}.$$

# Berechnen von Primzahlen

Wie häufig kommen Primzahlen vor?

Für  $n \geq 1$  sei  $\pi(n)$  die Zahl der Primzahlen  $\leq n$ .

Primzahlensatz. Für alle  $n \in \mathbb{N}$  gilt

$$\pi(n) = (1 + o(1)) \cdot \frac{n}{\ln(n)}.$$

Effiziente Berechnung großer Primzahlen.

Ungefährt jede  $\ln(n)$ -te Zahl ist also eine Primzahl.

Wenn wir ein schnelles Verfahren haben um für ein Zahl  $p$  zu entscheiden, ob sie prim ist, dann brauchen wir also nur zufällig Zahlen  $p$  in der gewünschten Größe zu wählen und zu testen, ob sie prim sind.

↪ randomisierte Verfahren zum Primzahlentest.

## 9.3 Modulare Arithmetik

# Modulare Arithmetik

**Definition.** Seien  $a, b \in \mathbb{Z}$  und sei  $m \geq 2$ .  $a$  ist *kongruent zu  $b$  modulo  $m$* , geschrieben

$$a \equiv b \pmod{m}$$

wenn  $a - b$  durch  $m$  teilbar ist.

**Mit anderen Worten.**  $a \equiv b \pmod{m}$  gilt genau dann, wenn man  $a$  aus  $b$  durch Addieren oder Subtrahieren eines Vielfachen von  $m$  erhalten kann.

**Beispiel.**  $2 \equiv 12 \pmod{10}$  da  $(12 - 2) = 10$  durch 10 teilbar ist.

# Modulare Arithmetik

**Definition.** Seien  $a, b \in \mathbb{Z}$  und sei  $m \geq 2$ .  $a$  ist *kongruent zu  $b$  modulo  $m$* , geschrieben

$$a \equiv b \pmod{m}$$

wenn  $a - b$  durch  $m$  teilbar ist.

**Mit anderen Worten.**  $a \equiv b \pmod{m}$  gilt genau dann, wenn man  $a$  aus  $b$  durch Addieren oder Subtrahieren eines Vielfachen von  $m$  erhalten kann.

**Beispiel.**  $2 \equiv 12 \pmod{10}$  da  $(12 - 2) = 10$  durch 10 teilbar ist.

**Beobachtung.**

Aus  $a \equiv b \pmod{m}$  folgt  $a + 2m \equiv b - 3m \pmod{m}$ .

# Modulare Arithmetik

Beobachtung. Für jedes  $m \geq 2$  ist  $\equiv$  eine Äquivalenzrelation auf  $\mathbb{Z}$ .

Die Relation hat  $m$  Äquivalenzklassen.

Ein Repräsentantensystem ist zum Beispiel  $\mathbb{Z}_m := \{0, \dots, m-1\}$ .

Definition. Sei  $a, b \in \mathbb{Z}$ ,  $m \geq 2$ .

$a \equiv b \pmod{m}$ ,  
wenn  $a - b$  durch  $m$  teilbar ist.

# Modulare Arithmetik

**Beobachtung.** Für jedes  $m \geq 2$  ist  $\equiv$  eine Äquivalenzrelation auf  $\mathbb{Z}$ .

Die Relation hat  $m$  Äquivalenzklassen.

Ein Repräsentantensystem ist zum Beispiel  $\mathbb{Z}_m := \{0, \dots, m-1\}$ .

**Modulus.** Wir schreiben  $a \bmod m$  für den Rest von  $a$  modulo  $m$ .

**Beachte.** Die Modulooperation ist als Rest der Division definiert.

Jede Zahl  $z \in \mathbb{Z}$  lässt sich als ein Vielfaches von  $m$  und einem positiven Rest  $r$  mit  $m > r \geq 0$  darstellen. D.h.  $z = qm + r$  für ein  $q \in \mathbb{Z}$ .

Für negative Zahlen betrachten wir also den positiven Rest.

Es gilt also z.B.  $-1 \bmod 3 = 2$ , da  $-1 = -1 \cdot 3 + 2$ .

**Definition.** Sei  $a, b \in \mathbb{Z}$ ,  $m \geq 2$ .  
 $a \equiv b \pmod{m}$ ,  
wenn  $a - b$  durch  $m$  teilbar ist.

# Modulare Arithmetik

**Beobachtung.** Für jedes  $m \geq 2$  ist  $\equiv$  eine Äquivalenzrelation auf  $\mathbb{Z}$ .

Die Relation hat  $m$  Äquivalenzklassen.

Ein Repräsentantensystem ist zum Beispiel  $\mathbb{Z}_m := \{0, \dots, m-1\}$ .

**Modulus.** Wir schreiben  $a \bmod m$  für den Rest von  $a$  modulo  $m$ .

**Beachte.** Die Modulooperation ist als Rest der Division definiert.

Jede Zahl  $z \in \mathbb{Z}$  lässt sich als ein Vielfaches von  $m$  und einem positiven Rest  $r$  mit  $m > r \geq 0$  darstellen. D.h.  $z = qm + r$  für ein  $q \in \mathbb{Z}$ .

Für negative Zahlen betrachten wir also den positiven Rest.

Es gilt also z.B.  $-1 \bmod 3 = 2$ , da  $-1 = -1 \cdot 3 + 2$ .

**Beachte.** Unterschied zwischen  $a \equiv b \pmod{m}$  und  $a \bmod m$ :

Der erste Ausdruck sagt aus, dass  $a$  und  $b$  in einer Relation stehen und der Wert des zweiten Ausdrucks ist eine Zahl.

**Definition.** Sei  $a, b \in \mathbb{Z}$ ,  $m \geq 2$ .  
 $a \equiv b \pmod{m}$ ,  
wenn  $a - b$  durch  $m$  teilbar ist.

# Modulare Arithmetik

**Modulares Rechnen.** Modulo einer Zahl  $m$  zu rechnen hat den Vorteil, dass der Wert jedes Ausdrucks modulo  $m$  eine Zahl aus  $\mathbb{Z}_m$  ist, also mit  $\lceil \log m \rceil$  Bits dargestellt werden kann.

# Modulare Arithmetik

**Modulares Rechnen.** Modulo einer Zahl  $m$  zu rechnen hat den Vorteil, dass der Wert jedes Ausdrucks modulo  $m$  eine Zahl aus  $\mathbb{Z}_m$  ist, also mit  $\lceil \log m \rceil$  Bits dargestellt werden kann.

Wir wollen daher auch zum Berechnen von  $(a + b) \bmod m$  oder  $(a \cdot b) \bmod m$  möglichst vermeiden, erst die Summe bzw. das Produkt ausrechnen zu müssen, um dann den Rest modulo  $m$  zu bilden.

Das nächste Lemma zeigt, wie man große Zwischenergebnisse vermeidet.

# Modulare Arithmetik

**Modulares Rechnen.** Modulo einer Zahl  $m$  zu rechnen hat den Vorteil, dass der Wert jedes Ausdrucks modulo  $m$  eine Zahl aus  $\mathbb{Z}_m$  ist, also mit  $\lceil \log m \rceil$  Bits dargestellt werden kann.

Wir wollen daher auch zum Berechnen von  $(a + b) \bmod m$  oder  $(a \cdot b) \bmod m$  möglichst vermeiden, erst die Summe bzw. das Produkt ausrechnen zu müssen, um dann den Rest modulo  $m$  zu bilden.

Das nächste Lemma zeigt, wie man große Zwischenergebnisse vermeidet.

**Lemma.** Für alle  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$  gilt

$$(a + b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$$

und

$$(a \cdot b) \bmod m = ((a \bmod m) \cdot (b \bmod m)) \bmod m.$$

# Beweis

**Beweis.** Aus der Definition folgt: Es gibt  $t_a, t_b \in \mathbb{Z}$  mit  
 $a = t_a \cdot m + (a \bmod m)$  und  $b = t_b \cdot m + (b \bmod m)$ .

Ferner gilt nach Definition für alle  $t, z \in \mathbb{Z}$ :  
 $(tm + z) \bmod m = z \bmod m$ .

**Lemma.**

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned}(a+b) \bmod m &= \\ ((a \bmod m) + (b \bmod m)) \bmod m\end{aligned}$$

und

$$\begin{aligned}(a \cdot b) \bmod m &= \\ ((a \bmod m) \cdot (b \bmod m)) \bmod m.\end{aligned}$$

# Beweis

**Beweis.** Aus der Definition folgt: Es gibt  $t_a, t_b \in \mathbb{Z}$  mit  $a = t_a \cdot m + (a \bmod m)$  und  $b = t_b \cdot m + (b \bmod m)$ .

Ferner gilt nach Definition für alle  $t, z \in \mathbb{Z}$ :  
 $(tm + z) \bmod m = z \bmod m$ .

Also haben wir

$$(a + b) \bmod m$$

**Lemma.**

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned} (a + b) \bmod m &= \\ ((a \bmod m) + (b \bmod m)) \bmod m \end{aligned}$$

und

$$\begin{aligned} (a \cdot b) \bmod m &= \\ ((a \bmod m) \cdot (b \bmod m)) \bmod m. \end{aligned}$$

# Beweis

**Beweis.** Aus der Definition folgt: Es gibt  $t_a, t_b \in \mathbb{Z}$  mit  $a = t_a \cdot m + (a \bmod m)$  und  $b = t_b \cdot m + (b \bmod m)$ .

Ferner gilt nach Definition für alle  $t, z \in \mathbb{Z}$ :

$$(tm + z) \bmod m = z \bmod m.$$

Also haben wir

$$(a + b) \bmod m = (t_a m + (a \bmod m) + t_b m + (b \bmod m)) \bmod m$$

**Lemma.**

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned} (a + b) \bmod m &= \\ ((a \bmod m) + (b \bmod m)) \bmod m \end{aligned}$$

und

$$\begin{aligned} (a \cdot b) \bmod m &= \\ ((a \bmod m) \cdot (b \bmod m)) \bmod m. \end{aligned}$$

# Beweis

**Beweis.** Aus der Definition folgt: Es gibt  $t_a, t_b \in \mathbb{Z}$  mit  $a = t_a \cdot m + (a \bmod m)$  und  $b = t_b \cdot m + (b \bmod m)$ .

Ferner gilt nach Definition für alle  $t, z \in \mathbb{Z}$ :

$$(tm + z) \bmod m = z \bmod m.$$

Also haben wir

$$\begin{aligned}(a + b) \bmod m &= (t_a m + (a \bmod m) + t_b m + (b \bmod m)) \bmod m \\ &= ((t_a + t_b)m + (a \bmod m) + (b \bmod m)) \bmod m\end{aligned}$$

**Lemma.**

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned}(a + b) \bmod m &= \\ &\quad ((a \bmod m) + (b \bmod m)) \bmod m\end{aligned}$$

und

$$\begin{aligned}(a \cdot b) \bmod m &= \\ &\quad ((a \bmod m) \cdot (b \bmod m)) \bmod m.\end{aligned}$$

# Beweis

**Beweis.** Aus der Definition folgt: Es gibt  $t_a, t_b \in \mathbb{Z}$  mit  $a = t_a \cdot m + (a \bmod m)$  und  $b = t_b \cdot m + (b \bmod m)$ .

Ferner gilt nach Definition für alle  $t, z \in \mathbb{Z}$ :

$$(tm + z) \bmod m = z \bmod m.$$

Also haben wir

$$\begin{aligned} (a + b) \bmod m &= (t_a m + (a \bmod m) + t_b m + (b \bmod m)) \bmod m \\ &= ((t_a + t_b)m + (a \bmod m) + (b \bmod m)) \bmod m \\ &= ((a \bmod m) + (b \bmod m)) \bmod m \end{aligned}$$

und

$$(a \cdot b) \bmod m =$$

**Lemma.**

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned} (a + b) \bmod m &= \\ &\quad ((a \bmod m) + (b \bmod m)) \\ &\quad \bmod m \end{aligned}$$

und

$$\begin{aligned} (a \cdot b) \bmod m &= \\ &\quad ((a \bmod m) \cdot (b \bmod m)) \\ &\quad \bmod m. \end{aligned}$$

# Beweis

**Beweis.** Aus der Definition folgt: Es gibt  $t_a, t_b \in \mathbb{Z}$  mit  $a = t_a \cdot m + (a \bmod m)$  und  $b = t_b \cdot m + (b \bmod m)$ .

Ferner gilt nach Definition für alle  $t, z \in \mathbb{Z}$ :

$$(tm + z) \bmod m = z \bmod m.$$

Also haben wir

$$\begin{aligned} (a + b) \bmod m &= (t_a m + (a \bmod m) + t_b m + (b \bmod m)) \bmod m \\ &= ((t_a + t_b)m + (a \bmod m) + (b \bmod m)) \bmod m \\ &= ((a \bmod m) + (b \bmod m)) \bmod m \end{aligned}$$

und

$$(a \cdot b) \bmod m = ((t_a m + (a \bmod m)) \cdot (t_b m + (b \bmod m))) \bmod m$$

**Lemma.**

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned} (a + b) \bmod m &= \\ &((a \bmod m) + (b \bmod m)) \bmod m \end{aligned}$$

und

$$\begin{aligned} (a \cdot b) \bmod m &= \\ &((a \bmod m) \cdot (b \bmod m)) \bmod m. \end{aligned}$$

# Beweis

**Beweis.** Aus der Definition folgt: Es gibt  $t_a, t_b \in \mathbb{Z}$  mit  $a = t_a \cdot m + (a \bmod m)$  und  $b = t_b \cdot m + (b \bmod m)$ .

Ferner gilt nach Definition für alle  $t, z \in \mathbb{Z}$ :

$$(tm + z) \bmod m = z \bmod m.$$

Also haben wir

$$\begin{aligned} (a + b) \bmod m &= (t_a m + (a \bmod m) + t_b m + (b \bmod m)) \bmod m \\ &= ((t_a + t_b)m + (a \bmod m) + (b \bmod m)) \bmod m \\ &= ((a \bmod m) + (b \bmod m)) \bmod m \end{aligned}$$

und

$$\begin{aligned} (a \cdot b) \bmod m &= ((t_a m + (a \bmod m)) \cdot (t_b m + (b \bmod m))) \bmod m \\ &= (t_a t_b m^2 + t_a m \cdot (b \bmod m) + t_b m \cdot (a \bmod m) \\ &\quad + (a \bmod m) \cdot (b \bmod m)) \bmod m \end{aligned}$$

**Lemma.**

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned} (a + b) \bmod m &= \\ &((a \bmod m) + (b \bmod m)) \bmod m \end{aligned}$$

und

$$\begin{aligned} (a \cdot b) \bmod m &= \\ &((a \bmod m) \cdot (b \bmod m)) \bmod m. \end{aligned}$$

# Beweis

**Beweis.** Aus der Definition folgt: Es gibt  $t_a, t_b \in \mathbb{Z}$  mit  $a = t_a \cdot m + (a \bmod m)$  und  $b = t_b \cdot m + (b \bmod m)$ .

Ferner gilt nach Definition für alle  $t, z \in \mathbb{Z}$ :

$$(tm + z) \bmod m = z \bmod m.$$

Also haben wir

$$\begin{aligned} (a + b) \bmod m &= (t_a m + (a \bmod m) + t_b m + (b \bmod m)) \bmod m \\ &= ((t_a + t_b)m + (a \bmod m) + (b \bmod m)) \bmod m \\ &= ((a \bmod m) + (b \bmod m)) \bmod m \end{aligned}$$

und

$$\begin{aligned} (a \cdot b) \bmod m &= ((t_a m + (a \bmod m)) \cdot (t_b m + (b \bmod m))) \bmod m \\ &= (t_a t_b m^2 + t_a m \cdot (b \bmod m) + t_b m \cdot (a \bmod m) \\ &\quad + (a \bmod m) \cdot (b \bmod m)) \bmod m \\ &= ((a \bmod m) \cdot (b \bmod m)) \bmod m. \quad \square \end{aligned}$$

**Lemma.**

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned} (a + b) \bmod m &= \\ &((a \bmod m) + (b \bmod m)) \bmod m \end{aligned}$$

und

$$\begin{aligned} (a \cdot b) \bmod m &= \\ &((a \bmod m) \cdot (b \bmod m)) \bmod m. \end{aligned}$$

# Anwendung

Rechnen mit großen Zahlen modulo  $m$ .

Wir wollen  $7^{66} \bmod 13$  berechnen.

Dazu berechnen wir zunächst die Reste von  $7^k$  für alle 2-er Potenzen  $k \leq 66$ . Dabei verwenden wir vorheriges Lemma.

$$7^2 \bmod 13 = 49 \bmod 13 = 10$$

Lemma.

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned}(a + b) \bmod m &= \\ ((a \bmod m) + (b \bmod m)) \bmod m\end{aligned}$$

$$\begin{aligned}(a \cdot b) \bmod m &= \\ ((a \bmod m) \cdot (b \bmod m)) \bmod m.\end{aligned}$$

# Anwendung

Rechnen mit großen Zahlen modulo  $m$ .

Wir wollen  $7^{66} \bmod 13$  berechnen.

Dazu berechnen wir zunächst die Reste von  $7^k$  für alle 2-er Potenzen  $k \leq 66$ . Dabei verwenden wir vorheriges Lemma.

$$7^2 \bmod 13 = 49 \bmod 13 = 10$$

$$7^4 \bmod 13 = (7^2 \bmod 13) \cdot (7^2 \bmod 13) \bmod 13 = 10 \cdot 10 \bmod 13 = 9$$

Lemma.

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned}(a + b) \bmod m &= ((a \bmod m) + (b \bmod m)) \\ &\quad \bmod m\end{aligned}$$

$$\begin{aligned}(a \cdot b) \bmod m &= ((a \bmod m) \cdot (b \bmod m)) \\ &\quad \bmod m.\end{aligned}$$

# Anwendung

Rechnen mit großen Zahlen modulo  $m$ .

Wir wollen  $7^{66} \bmod 13$  berechnen.

Dazu berechnen wir zunächst die Reste von  $7^k$  für alle 2-er Potenzen  $k \leq 66$ . Dabei verwenden wir vorheriges Lemma.

$$7^2 \bmod 13 = 49 \bmod 13 = 10$$

$$7^4 \bmod 13 = (7^2 \bmod 13) \cdot (7^2 \bmod 13) \bmod 13 = 10 \cdot 10 \bmod 13 = 9$$

$$7^8 \bmod 13 = (7^4 \bmod 13)^2 \bmod 13 = 9 \cdot 9 \bmod 13 = 81 \bmod 13 = 3$$

Lemma.

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned}(a + b) \bmod m &= \\ ((a \bmod m) + (b \bmod m)) \bmod m\end{aligned}$$

$$\begin{aligned}(a \cdot b) \bmod m &= \\ ((a \bmod m) \cdot (b \bmod m)) \bmod m.\end{aligned}$$

# Anwendung

Rechnen mit großen Zahlen modulo  $m$ .

Wir wollen  $7^{66} \bmod 13$  berechnen.

Dazu berechnen wir zunächst die Reste von  $7^k$  für alle 2-er Potenzen  $k \leq 66$ . Dabei verwenden wir vorheriges Lemma.

$$7^2 \bmod 13 = 49 \bmod 13 = 10$$

$$7^4 \bmod 13 = (7^2 \bmod 13) \cdot (7^2 \bmod 13) \bmod 13 = 10 \cdot 10 \bmod 13 = 9$$

$$7^8 \bmod 13 = (7^4 \bmod 13)^2 \bmod 13 = 9 \cdot 9 \bmod 13 = 81 \bmod 13 = 3$$

$$7^{16} \bmod 13 = 9$$

Lemma.

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned}(a + b) \bmod m &= \\ ((a \bmod m) + (b \bmod m)) \bmod m\end{aligned}$$

$$\begin{aligned}(a \cdot b) \bmod m &= \\ ((a \bmod m) \cdot (b \bmod m)) \bmod m.\end{aligned}$$

# Anwendung

Rechnen mit großen Zahlen modulo  $m$ .

Wir wollen  $7^{66} \bmod 13$  berechnen.

Dazu berechnen wir zunächst die Reste von  $7^k$  für alle 2-er Potenzen  $k \leq 66$ . Dabei verwenden wir vorheriges Lemma.

$$7^2 \bmod 13 = 49 \bmod 13 = 10$$

$$7^4 \bmod 13 = (7^2 \bmod 13) \cdot (7^2 \bmod 13) \bmod 13 = 10 \cdot 10 \bmod 13 = 9$$

$$7^8 \bmod 13 = (7^4 \bmod 13)^2 \bmod 13 = 9 \cdot 9 \bmod 13 = 81 \bmod 13 = 3$$

$$7^{16} \bmod 13 = 9$$

$$7^{32} \bmod 13 = 3$$

$$7^{64} \bmod 13 = 9$$

Lemma.

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned}(a + b) \bmod m &= \\ ((a \bmod m) + (b \bmod m)) \bmod m\end{aligned}$$

$$\begin{aligned}(a \cdot b) \bmod m &= \\ ((a \bmod m) \cdot (b \bmod m)) \bmod m.\end{aligned}$$

# Anwendung

Rechnen mit großen Zahlen modulo  $m$ .

Wir wollen  $7^{66} \bmod 13$  berechnen.

Dazu berechnen wir zunächst die Reste von  $7^k$  für alle 2-er Potenzen  $k \leq 66$ . Dabei verwenden wir vorheriges Lemma.

$$7^2 \bmod 13 = 49 \bmod 13 = 10$$

$$7^4 \bmod 13 = (7^2 \bmod 13) \cdot (7^2 \bmod 13) \bmod 13 = 10 \cdot 10 \bmod 13 = 9$$

$$7^8 \bmod 13 = (7^4 \bmod 13)^2 \bmod 13 = 9 \cdot 9 \bmod 13 = 81 \bmod 13 = 3$$

$$7^{16} \bmod 13 = 9$$

$$7^{32} \bmod 13 = 3$$

$$7^{64} \bmod 13 = 9$$

Es gilt  $66 = 64 + 2$ . Wir wenden das Lemma noch einmal an:

$$\begin{aligned} 7^{66} \bmod 13 &= (7^{64} \cdot 7^2) \bmod 13 = ((7^{64} \bmod 13) \cdot (7^2 \bmod 13)) \bmod 13 \\ &= (9 \cdot 10) \bmod 13 = 12. \end{aligned}$$

**Lemma.**

Für  $a, b, m \in \mathbb{Z}$  mit  $m \geq 2$ :

$$\begin{aligned} (a + b) \bmod m &= ((a \bmod m) + (b \bmod m)) \bmod m \\ (a \cdot b) \bmod m &= ((a \bmod m) \cdot (b \bmod m)) \bmod m. \end{aligned}$$

# Folgerung

**3-Teilbarkeit.** Eine Zahl ist genau dann durch 3 teilbar, wenn ihre Quersumme durch 3 teilbar ist.

# Folgerung

**3-Teilbarkeit.** Eine Zahl ist genau dann durch 3 teilbar, wenn ihre Quersumme durch 3 teilbar ist.

**Beweis.** Aus  $10 \bmod 3 = 1$  folgt, dass  $10^i \bmod 3 = 1$  für alle  $i \in \mathbb{N}$ .

Sei nun  $n \in \mathbb{N}$  gegeben. In Dezimaldarstellung:  $n = a_k a_{k-1} \cdots a_1 a_0$ .

# Folgerung

**3-Teilbarkeit.** Eine Zahl ist genau dann durch 3 teilbar, wenn ihre Quersumme durch 3 teilbar ist.

**Beweis.** Aus  $10 \mod 3 = 1$  folgt, dass  $10^i \mod 3 = 1$  für alle  $i \in \mathbb{N}$ .

Sei nun  $n \in \mathbb{N}$  gegeben. In Dezimaldarstellung:  $n = a_k a_{k-1} \cdots a_1 a_0$ .  
Dann gilt

$$\begin{aligned} (\sum_{i=0}^k a_i \cdot 10^i) \mod 3 &= \left( \sum_{i=0}^k (a_i \mod 3)(10^i \mod 3) \right) \mod 3 \\ &= \left( \sum_{i=0}^k a_i \right) \mod 3 \end{aligned}$$

# Folgerung

**3-Teilbarkeit.** Eine Zahl ist genau dann durch 3 teilbar, wenn ihre Quersumme durch 3 teilbar ist.

**Beweis.** Aus  $10 \mod 3 = 1$  folgt, dass  $10^i \mod 3 = 1$  für alle  $i \in \mathbb{N}$ .

Sei nun  $n \in \mathbb{N}$  gegeben. In Dezimaldarstellung:  $n = a_k a_{k-1} \cdots a_1 a_0$ . Dann gilt

$$\begin{aligned} \left( \sum_{i=0}^k a_i \cdot 10^i \right) \mod 3 &= \left( \sum_{i=0}^k (a_i \mod 3)(10^i \mod 3) \right) \mod 3 \\ &= \left( \sum_{i=0}^k a_i \right) \mod 3 \end{aligned}$$

Insbesondere folgt also

$$\sum_{i=0}^k a_i 10^i \equiv 0 \pmod{3} \iff \sum_{i=0}^k a_i \equiv 0 \pmod{3}.$$

# Rechenregeln der modularen Arithmetik

**Lemma.** Für alle  $a, b, c, d, m \in \mathbb{Z}$  mit  $m \geq 2$  gilt: Wenn

$$a \equiv b \pmod{m} \quad \text{und} \quad c \equiv d \pmod{m}$$

dann

$$(a + c) \equiv (b + d) \pmod{m} \quad \text{und} \quad a \cdot c \equiv b \cdot d \pmod{m}.$$

**Beweis.** Aus  $a \equiv b \pmod{m}$  folgt nach Definition:

Es gibt ein  $t \in \mathbb{Z}$  mit  $a - b = tm$ .

Ebenso existiert ein  $t' \in \mathbb{Z}$  mit  $c - d = t'm$ .

# Rechenregeln der modularen Arithmetik

**Lemma.** Für alle  $a, b, c, d, m \in \mathbb{Z}$  mit  $m \geq 2$  gilt: Wenn

$$a \equiv b \pmod{m} \quad \text{und} \quad c \equiv d \pmod{m}$$

dann

$$(a + c) \equiv (b + d) \pmod{m} \quad \text{und} \quad a \cdot c \equiv b \cdot d \pmod{m}.$$

**Beweis.** Aus  $a \equiv b \pmod{m}$  folgt nach Definition:

Es gibt ein  $t \in \mathbb{Z}$  mit  $a - b = tm$ .

Ebenso existiert ein  $t' \in \mathbb{Z}$  mit  $c - d = t'm$ .

Daraus folgt  $(a + c) - (b + d) = (t + t')m$  und somit

$$(a + c) \equiv (b + d) \pmod{m}.$$

# Rechenregeln der modularen Arithmetik

**Lemma.** Für alle  $a, b, c, d, m \in \mathbb{Z}$  mit  $m \geq 2$  gilt: Wenn

$$a \equiv b \pmod{m} \quad \text{und} \quad c \equiv d \pmod{m}$$

dann

$$(a + c) \equiv (b + d) \pmod{m} \quad \text{und} \quad a \cdot c \equiv b \cdot d \pmod{m}.$$

**Beweis.** Aus  $a \equiv b \pmod{m}$  folgt nach Definition:

Es gibt ein  $t \in \mathbb{Z}$  mit  $a - b = tm$ .

Ebenso existiert ein  $t' \in \mathbb{Z}$  mit  $c - d = t'm$ .

Daraus folgt  $(a + c) - (b + d) = (t + t')m$  und somit

$$(a + c) \equiv (b + d) \pmod{m}.$$

Nun zeigen wir die zweite Aussage.

Aus  $c - d = t'm$  folgt  $c = t'm + d$  und aus  $a - b = tm$  folgt  $b = a - tm$ .

# *Rechenregeln der modularen Arithmetik*

**Lemma.** Für alle  $a, b, c, d, m \in \mathbb{Z}$  mit  $m \geq 2$  gilt: Wenn

$$a \equiv b \pmod{m} \quad \text{und} \quad c \equiv d \pmod{m}$$

dann

$$(a + c) \equiv (b + d) \pmod{m} \quad \text{und} \quad a \cdot c \equiv b \cdot d \pmod{m}.$$

**Beweis.** Aus  $a \equiv b \pmod{m}$  folgt nach Definition:

Es gibt ein  $t \in \mathbb{Z}$  mit  $a - b = tm$ .

Ebenso existiert ein  $t' \in \mathbb{Z}$  mit  $c - d = t'm$ .

Daraus folgt  $(a + c) - (b + d) = (t + t')m$  und somit

$$(a + c) \equiv (b + d) \pmod{m}.$$

Nun zeigen wir die zweite Aussage.

Aus  $c - d = t'm$  folgt  $c = t'm + d$  und aus  $a - b = tm$  folgt  $b = a - tm$ .

Also gilt  $(ac - bd) = a(t'm + d) - (a - tm)d = (at' + dt)m$  und somit

$$(a \cdot c) \equiv (b \cdot d) \pmod{m}. \quad \square$$

# Kürzen in der modularen Arithmetik

**Lemma.** Für alle  $a, b, c, m \in \mathbb{Z}$  mit  $m \geq 2$  und  $\text{ggT}(c, m) = 1$  gilt

Wenn  $a \cdot c \equiv b \cdot c \pmod{m}$  dann  $a \equiv b \pmod{m}$ .

# Kürzen in der modularen Arithmetik

**Lemma.** Für alle  $a, b, c, m \in \mathbb{Z}$  mit  $m \geq 2$  und  $\text{ggT}(c, m) = 1$  gilt

Wenn  $a \cdot c \equiv b \cdot c \pmod{m}$  dann  $a \equiv b \pmod{m}$ .

**Beweis.**

Aus  $ac \equiv bc \pmod{m}$  folgt  $m \mid (a - b)c$ .

# Kürzen in der modularen Arithmetik

**Lemma.** Für alle  $a, b, c, m \in \mathbb{Z}$  mit  $m \geq 2$  und  $\text{ggT}(c, m) = 1$  gilt

Wenn  $a \cdot c \equiv b \cdot c \pmod{m}$  dann  $a \equiv b \pmod{m}$ .

**Beweis.**

Aus  $ac \equiv bc \pmod{m}$  folgt  $m \mid (a - b)c$ .

Da  $\text{ggT}(c, m) = 1$ , folgt  $m \mid (a - b)$ .

# Kürzen in der modularen Arithmetik

**Lemma.** Für alle  $a, b, c, m \in \mathbb{Z}$  mit  $m \geq 2$  und  $\text{ggT}(c, m) = 1$  gilt

Wenn  $a \cdot c \equiv b \cdot c \pmod{m}$  dann  $a \equiv b \pmod{m}$ .

**Beweis.**

Aus  $ac \equiv bc \pmod{m}$  folgt  $m \mid (a - b)c$ .

Da  $\text{ggT}(c, m) = 1$ , folgt  $m \mid (a - b)$ .

Also gilt  $a \equiv b \pmod{m}$ . □

## 9.4 Beispielanwendungen der modularen Arithmetik

## Zufallszahlengeneratoren

**Generieren von Zufallszahlen.** Oft wird eine Sequenz zufälliger Zahlen benötigt, etwa für Testdatensätze.

Programmiersprachen bieten solche Befehle, z.B. `nrand48()` in C+. Gute Zufallszahlengeneratoren zu bauen ist ziemlich komplex.

# Zufallszahlengeneratoren

**Generieren von Zufallszahlen.** Oft wird eine Sequenz zufälliger Zahlen benötigt, etwa für Testdatensätze.

Programmiersprachen bieten solche Befehle, z.B. `nrand48()` in C+. Gute Zufallszahlengeneratoren zu bauen ist ziemlich komplex.

## Lineare Kongruenzmethode.

Ein einfaches, in der Praxis bewährtes Verfahren ist die lineare Kongruenzmethode. Gegeben sind feste Werte  $a$ ,  $c$ , und  $m$ .

Man wählt einen beliebigen Startwert  $x_0$  und berechnet iterativ:

$$x_{n+1} = ax_n + c \pmod{m}.$$

### Beispiel.

Setze  $m = 5$ ,  $c = 3$  und  $a = 6$ .

Dann erhalten wir:

$$x_0 = 0$$

$$x_1 = 6 * 0 + 3 = 3$$

$$x_2 = 6 * 3 + 3 = 21 \pmod{5} = 1$$

$$x_3 = 6 * 1 + 3 = 9 \pmod{5} = 4$$

$$x_4 = 6 * 4 + 3 = 27 \pmod{5} = 2$$

$$x_5 = 6 * 2 + 3 = 15 \pmod{5} = 0$$

# Zufallszahlengeneratoren

## Lineare Kongruenzmethode.

Ein einfaches, in der Praxis bewährtes Verfahren ist die lineare Kongruenzmethode. Gegeben sind feste Werte  $a$ ,  $c$ , und  $m$ .

Man wählt einen beliebigen Startwert  $x_0$  und berechnet iterativ:

$$x_{n+1} = ax_n + c \bmod m.$$

**Qualität des Verfahrens.** Die Qualität hängt von der Verteilung der Werte  $x_i$  ab, was von der Wahl der  $a, c, m$  abhängt.

Ein mögliches Kriterium ist, dass alle Zahlen  $0, \dots, m - 1$  erreicht werden, bevor ein  $x_i$  wiederholt wird.

Man kann zeigen, dass das genau dann passiert, wenn

- $\text{ggT}(c, m) = 1$  gilt, jede Primzahl, die  $m$  teilt, auch  $a - 1$  teilt und
- wenn  $4|m$ , dann ist auch  $a - 1$  ein Vielfaches von 4.

# International Standard Book Number (ISBN)

**ISBN Nummer.** 10- oder (seit 2007) 13-stellige Zahl.

ISBN-10 besteht aus **4** Segmenten, durch “-“ getrennt.

1. Länder- bzw. Sprachcode (0 für engl., 3 für deutschspr.)
2. Verlagsnummer
3. Laufende Nummer innerhalb des Verlagsprogramms
4. Prüfziffer

# International Standard Book Number (ISBN)

**ISBN Nummer.** 10- oder (seit 2007) 13-stellige Zahl.

ISBN-10 besteht aus **4** Segmenten, durch “-“ getrennt.

1. Länder- bzw. Sprachcode (0 für engl., 3 für deutschspr.)
2. Verlagsnummer
3. Laufende Nummer innerhalb des Verlagsprogramms
4. Prüfziffer

**Prüfziffer** Die Prüfziffer wird aus anderen Ziffern wie folgt berechnet.

Sind die ersten 9 Ziffern  $a_1 \dots a_9$ , dann ist die Prüfziffer

$$\left( \sum_{i=1}^9 i \cdot a_i \right) \bmod 11.$$

# International Standard Book Number (ISBN)

**ISBN Nummer.** 10- oder (seit 2007) 13-stellige Zahl.

ISBN-10 besteht aus **4** Segmenten, durch “-“ getrennt.

1. Länder- bzw. Sprachcode (0 für engl., 3 für deutschspr.)
2. Verlagsnummer
3. Laufende Nummer innerhalb des Verlagsprogramms
4. Prüfziffer

**Prüfziffer** Die Prüfziffer wird aus anderen Ziffern wie folgt berechnet.

Sind die ersten 9 Ziffern  $a_1 \dots a_9$ , dann ist die Prüfziffer

$$\left( \sum_{i=1}^9 i \cdot a_i \right) \bmod 11.$$

**Beispiel.**

Das Buch *Diskrete Strukturen* von Angelika Steger hat die Nummer

3-540-46660-6.

$$\begin{aligned}
 & 1 \cdot 3 + 2 \cdot 5 + 3 \cdot 4 + 4 \cdot 0 + \\
 & 5 \cdot 4 + 6 \cdot 6 + 7 \cdot 6 + 8 \cdot 6 + 9 \cdot 0 \\
 & = 171 = 6 \bmod 11
 \end{aligned}$$

# International Standard Book Number (ISBN)

**ISBN Nummer.** 10- oder (seit 2007) 13-stellige Zahl.

ISBN-10 besteht aus **4** Segmenten, durch “-“ getrennt.

1. Länder- bzw. Sprachcode (0 für engl., 3 für deutschspr.)
2. Verlagsnummer
3. Laufende Nummer innerhalb des Verlagsprogramms
4. Prüfziffer

**Prüfziffer** Die Prüfziffer wird aus anderen Ziffern wie folgt berechnet.

Sind die ersten 9 Ziffern  $a_1 \dots a_9$ , dann ist die Prüfziffer

$$\left( \sum_{i=1}^9 i \cdot a_i \right) \bmod 11.$$

**Eigenschaften.** Wenn man sich an genau einer Stelle vertippt, ergibt sich eine Zahl, für die die Prüfziffer falsch ist.

ISBN-10 hat noch die Eigenschaft, bei Transpositionen nebeneinander liegender Zahlen auch eine falsche Prüfziffer zu generieren.

**Beispiel.**

Das Buch *Diskrete Strukturen* von Angelika Steger hat die Nummer

3-540-46660-6.

$$\begin{aligned}
 & 1 \cdot 3 + 2 \cdot 5 + 3 \cdot 4 + 4 \cdot 0 + \\
 & 5 \cdot 4 + 6 \cdot 6 + 7 \cdot 6 + 8 \cdot 6 + 9 \cdot 0 \\
 & = 171 = 6 \bmod 11
 \end{aligned}$$

## 9.5 Der Euklidische Algorithmus

# Berechnen des ggT

Der ggT. Für spätere Anwendungen brauchen wir einen guten Weg, den ggT zweier Zahlen auszurechnen.

Beispiel. Sei  $n = 24948$  und  $m = 8712$ .

$$\begin{aligned} 24948 &= 2^2 \cdot 3^4 \cdot 7 \cdot 11 \\ 8712 &= 2^2 \cdot 3^4 \cdot 7 \cdot 11 \end{aligned}$$

Also ist der  $\text{ggT}(n, m) = 2^2 \cdot 3^2 \cdot 11 = 396$

# Berechnen des ggT

**Der ggT.** Für spätere Anwendungen brauchen wir einen guten Weg, den ggT zweier Zahlen auszurechnen.

Beispiel. Sei  $n = 24948$  und  $m = 8712$ .

$$\begin{aligned} 24948 &= 2^2 \cdot 3^4 \cdot 7 \cdot 11 \\ 8712 &= 2^2 \cdot 3^4 \cdot 7 \cdot 11 \end{aligned}$$

Also ist der  $\text{ggT}(n, m) = 2^2 \cdot 3^2 \cdot 11 = 396$

**Problem.** Für große Zahlen ist das Berechnen der Primfaktorzerlegung recht aufwendig.

Besser ist daher der Euklidische Algorithmus.

# Der Algorithmus von Euklid

Algorithmus: Euklidischer Algorithmus.

**Eingabe:** Zahlen  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:** ggT( $m, n$ ).

Algorithmus Euklid( $m, n$ )

Wenn  $m \mid n$  dann

gib  $m$  zurück

sonst

gibt Euklid( $n \bmod m, m$ ) zurück.

# Beispiel

Beispiel.

Algorithmus: Euklidischer Algorithmus.

**Eingabe:** Zahlen  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:** ggT( $m, n$ ).

Algorithmus Euklid( $m, n$ )

Wenn  $m \mid n$  dann

gib  $m$  zurück

sonst

gibt Euklid( $n \bmod m, m$ ) zurück.

# Der Algorithmus von Euklid

Algorithmus: Euklidischer Algorithmus.

**Eingabe:** Zahlen  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:** ggT( $m, n$ ).

Algorithmus Euklid( $m, n$ )

Wenn  $m \mid n$  dann

gib  $m$  zurück

sonst

gibt Euklid( $n \bmod m, m$ ) zurück.

# Der Algorithmus von Euklid

Algorithmus: Euklidischer Algorithmus.

**Eingabe:** Zahlen  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:** ggT( $m, n$ ).

Algorithmus Euklid( $m, n$ )

Wenn  $m \mid n$  dann

gib  $m$  zurück

sonst

gibt Euklid( $n \bmod m, m$ ) zurück.

**Lemma.** Sind  $m, n \in \mathbb{N}$  mit  $m \leq n$  und  $m \nmid n$ , so gilt

$$\text{ggT}(m, n) = \text{ggT}(n \bmod m, m).$$

# Beweis

Beweis.

Wir müssen zeigen, dass jeder Teiler von  $m$  und  $n$  auch ein Teiler von  $n \bmod m$  und  $m$  ist und umgekehrt.

**Lemma.**

Sind  $m, n \in \mathbb{N}$  mit  $m \leq n$  und  $m \nmid n$ , so gilt

$$\text{ggT}(m, n) = \text{ggT}(n \bmod m, m).$$

# Beweis

Beweis.

Wir müssen zeigen, dass jeder Teiler von  $m$  und  $n$  auch ein Teiler von  $n \bmod m$  und  $m$  ist und umgekehrt.

Sei also  $d \in \mathbb{N}$  mit  $d | n$  und  $d | m$ .

**Lemma.**

Sind  $m, n \in \mathbb{N}$  mit  $m \leq n$  und  $m \nmid n$ , so gilt

$$\text{ggT}(m, n) = \text{ggT}(n \bmod m, m).$$

# Beweis

Beweis.

Wir müssen zeigen, dass jeder Teiler von  $m$  und  $n$  auch ein Teiler von  $n \bmod m$  und  $m$  ist und umgekehrt.

Sei also  $d \in \mathbb{N}$  mit  $d | n$  und  $d | m$ .

Da  $n \bmod m = n - \ell \cdot m$  für ein geeignetes  $\ell \in \mathbb{N}$ , teilt  $d$  auch  $n \bmod m$ .

**Lemma.**

Sind  $m, n \in \mathbb{N}$  mit  $m \leq n$  und  $m \nmid n$ , so gilt

$$\text{ggT}(m, n) = \text{ggT}(n \bmod m, m).$$

# Beweis

Beweis.

Wir müssen zeigen, dass jeder Teiler von  $m$  und  $n$  auch ein Teiler von  $n \bmod m$  und  $m$  ist und umgekehrt.

Sei also  $d \in \mathbb{N}$  mit  $d | n$  und  $d | m$ .

Da  $n \bmod m = n - \ell \cdot m$  für ein geeignetes  $\ell \in \mathbb{N}$ , teilt  $d$  auch  $n \bmod m$ .

Umgekehrt, sei  $d \in \mathbb{N}$  mit  $d | (n \bmod m)$  und  $d | m$ .

Lemma.

Sind  $m, n \in \mathbb{N}$  mit  $m \leq n$  und  $m \nmid n$ , so gilt

$$\text{ggT}(m, n) = \text{ggT}(n \bmod m, m).$$

# Beweis

Beweis.

Wir müssen zeigen, dass jeder Teiler von  $m$  und  $n$  auch ein Teiler von  $n \bmod m$  und  $m$  ist und umgekehrt.

Sei also  $d \in \mathbb{N}$  mit  $d | n$  und  $d | m$ .

Da  $n \bmod m = n - \ell \cdot m$  für ein geeignetes  $\ell \in \mathbb{N}$ , teilt  $d$  auch  $n \bmod m$ .

Umgekehrt, sei  $d \in \mathbb{N}$  mit  $d | (n \bmod m)$  und  $d | m$ .

Da wiederum  $n = \ell \cdot m + (n \bmod m)$ , teilt  $d$  auch  $n$ .

**Lemma.**

Sind  $m, n \in \mathbb{N}$  mit  $m \leq n$  und  $m \nmid n$ , so gilt

$$\text{ggT}(m, n) = \text{ggT}(n \bmod m, m).$$

## Beweis

Beweis.

Wir müssen zeigen, dass jeder Teiler von  $m$  und  $n$  auch ein Teiler von  $n \bmod m$  und  $m$  ist und umgekehrt.

Sei also  $d \in \mathbb{N}$  mit  $d | n$  und  $d | m$ .

Da  $n \bmod m = n - \ell \cdot m$  für ein geeignetes  $\ell \in \mathbb{N}$ , teilt  $d$  auch  $n \bmod m$ .

Umgekehrt, sei  $d \in \mathbb{N}$  mit  $d | (n \bmod m)$  und  $d | m$ .

Da wiederum  $n = \ell \cdot m + (n \bmod m)$ , teilt  $d$  auch  $n$ .

Insbesondere gilt also  $\text{ggT}(m, n) | \text{ggT}(n \bmod m, m)$  und  $\text{ggT}(n \bmod m, m) | \text{ggT}(m, n)$ . Daraus folgt die Gleichheit.  $\square$

Lemma.

Sind  $m, n \in \mathbb{N}$  mit  $m \leq n$  und  $m \nmid n$ , so gilt

$$\text{ggT}(m, n) = \text{ggT}(n \bmod m, m).$$

# Erweiterter Euklidischer Algorithmus

## Erweiterter Euklidischer Algorithmus

**Eingabe:**  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:**  $x, y \in \mathbb{Z}$  mit  $\text{ggT}(m, n) = mx + ny$ .

Algorithmus Erw-Euklid( $m, n$ )

Wenn  $m \mid n$  dann

gib  $(1, 0)$  zurück

sonst

sei  $(x', y') = \text{Erw-Euklid}(n \bmod m, m)$ .

sei  $x := y' - x' \cdot \lfloor \frac{n}{m} \rfloor$

$y := x'$

gib  $(x, y)$  zurück.

### Satz.

Für alle  $m, n \in \mathbb{N}$  mit  $m \leq n$  gibt es  $x, y \in \mathbb{Z}$  mit

$$\text{ggT}(m, n) = mx + ny.$$

Erw-Euklid liefert auf Eingabe  $(m, n)$  solche Zahlen zurück.

# Beweis

**Beweis.** Zeigen nur zweite Aussage, aus der die erste folgt.

Der Beweis per Induktion über Zahl der rekursiven Aufrufe.

## Satz.

Für alle  $m, n \in \mathbb{N}$  mit  $m \leq n$  gibt es  $x, y \in \mathbb{Z}$  mit

$$\text{ggT}(m, n) = mx + ny.$$

Erw-Euklid liefert auf Eingabe  $(m, n)$  solche Zahlen zurück.

## Erweiterter Euklidischer Algorithmus

**Eingabe:**  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:**  $x, y \in \mathbb{Z}$  mit  
 $\text{ggT}(m, n) = mx + ny$ .

Algorithmus Erw-Euklid( $m, n$ )

Wenn  $m | n$  dann

gib  $(1, 0)$  zurück

sonst

sei  $(x', y') =$   
 Erw-Euklid( $n \bmod m, m$ ).

sei  $x := y' - x' \cdot \lfloor \frac{n}{m} \rfloor$

$y := x'$

gib  $(x, y)$  zurück.

# Beweis

**Beweis.** Zeigen nur zweite Aussage, aus der die erste folgt.

Der Beweis per Induktion über Zahl der rekursiven Aufrufe.

**Klar.** Algorithmus terminiert, da die Summe  $m + n$  in jedem Schritt kleiner wird.

## Satz.

Für alle  $m, n \in \mathbb{N}$  mit  $m \leq n$  gibt es  $x, y \in \mathbb{Z}$  mit

$$\text{ggT}(m, n) = mx + ny.$$

Erw-Euklid liefert auf Eingabe  $(m, n)$  solche Zahlen zurück.

## Erweiterter Euklidischer Algorithmus

**Eingabe:**  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:**  $x, y \in \mathbb{Z}$  mit  
 $\text{ggT}(m, n) = mx + ny$ .

Algorithmus Erw-Euklid( $m, n$ )

Wenn  $m | n$  dann

gib  $(1, 0)$  zurück

sonst

sei  $(x', y') =$   
 Erw-Euklid( $n \bmod m, m$ ).

sei  $x := y' - x' \cdot \lfloor \frac{n}{m} \rfloor$

$y := x'$

gib  $(x, y)$  zurück.

# Beweis

**Beweis.** Zeigen nur zweite Aussage, aus der die erste folgt.

Der Beweis per Induktion über Zahl der rekursiven Aufrufe.

**Klar.** Algorithmus terminiert, da die Summe  $m + n$  in jedem Schritt kleiner wird.

Für  $m = 0$ , ist der Algorithmus offensichtlich korrekt.

## Satz.

Für alle  $m, n \in \mathbb{N}$  mit  $m \leq n$  gibt es  $x, y \in \mathbb{Z}$  mit

$$\text{ggT}(m, n) = mx + ny.$$

Erw-Euklid liefert auf Eingabe  $(m, n)$  solche Zahlen zurück.

## Erweiterter Euklidischer Algorithmus

**Eingabe:**  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:**  $x, y \in \mathbb{Z}$  mit  
 $\text{ggT}(m, n) = mx + ny$ .

Algorithmus Erw-Euklid( $m, n$ )

Wenn  $m | n$  dann

gib  $(1, 0)$  zurück

sonst

sei  $(x', y') =$   
 Erw-Euklid( $n \bmod m, m$ ).

sei  $x := y' - x' \cdot \lfloor \frac{n}{m} \rfloor$

$y := x'$

gib  $(x, y)$  zurück.

# Beweis

**Beweis.** Zeigen nur zweite Aussage, aus der die erste folgt.

Der Beweis per Induktion über Zahl der rekursiven Aufrufe.

**Klar.** Algorithmus terminiert, da die Summe  $m + n$  in jedem Schritt kleiner wird.

Für  $m = 0$ , ist der Algorithmus offensichtlich korrekt.

Gelte nun  $m > 0$ .

## Satz.

Für alle  $m, n \in \mathbb{N}$  mit  $m \leq n$  gibt es  $x, y \in \mathbb{Z}$  mit

$$\text{ggT}(m, n) = mx + ny.$$

Erw-Euklid liefert auf Eingabe  $(m, n)$  solche Zahlen zurück.

## Erweiterter Euklidischer Algorithmus

**Eingabe:**  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:**  $x, y \in \mathbb{Z}$  mit  
 $\text{ggT}(m, n) = mx + ny$ .

Algorithmus Erw-Euklid( $m, n$ )

Wenn  $m | n$  dann

gib  $(1, 0)$  zurück

sonst

sei  $(x', y') =$   
 Erw-Euklid( $n \bmod m, m$ ).

sei  $x := y' - x' \cdot \lfloor \frac{n}{m} \rfloor$

$y := x'$

gib  $(x, y)$  zurück.

# Beweis

**Beweis.** Zeigen nur zweite Aussage, aus der die erste folgt.

Der Beweis per Induktion über Zahl der rekursiven Aufrufe.

**Klar.** Algorithmus terminiert, da die Summe  $m + n$  in jedem Schritt kleiner wird.

Für  $m = 0$ , ist der Algorithmus offensichtlich korrekt.

Gelte nun  $m > 0$ .

Wende Induktionsannahme auf  $m' := (n \bmod m)$  und  $n' := m$  an.

## Satz.

Für alle  $m, n \in \mathbb{N}$  mit  $m \leq n$  gibt es  $x, y \in \mathbb{Z}$  mit

$$\text{ggT}(m, n) = mx + ny.$$

Erw-Euklid liefert auf Eingabe  $(m, n)$  solche Zahlen zurück.

## Erweiterter Euklidischer Algorithmus

**Eingabe:**  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:**  $x, y \in \mathbb{Z}$  mit  
 $\text{ggT}(m, n) = mx + ny$ .

Algorithmus Erw-Euklid( $m, n$ )

Wenn  $m | n$  dann

gib  $(1, 0)$  zurück

sonst

sei  $(x', y') =$   
 Erw-Euklid( $n \bmod m, m$ ).

sei  $x := y' - x' \cdot \lfloor \frac{n}{m} \rfloor$

$y := x'$

gib  $(x, y)$  zurück.

# Beweis

**Beweis.** Zeigen nur zweite Aussage, aus der die erste folgt.

Der Beweis per Induktion über Zahl der rekursiven Aufrufe.

**Klar.** Algorithmus terminiert, da die Summe  $m + n$  in jedem Schritt kleiner wird.

Für  $m = 0$ , ist der Algorithmus offensichtlich korrekt.

Gelte nun  $m > 0$ .

Wende Induktionsannahme auf  $m' := (n \bmod m)$  und  $n' := m$  an.

Sei  $(x', y') := \text{Erw-Euklid}(m', n')$ , d.h.

$$\text{ggT}(n \bmod m, m) = x' \cdot (n \bmod m) + y' \cdot m.$$

## Satz.

Für alle  $m, n \in \mathbb{N}$  mit  $m \leq n$  gibt es  $x, y \in \mathbb{Z}$  mit

$$\text{ggT}(m, n) = mx + ny.$$

Erw-Euklid liefert auf Eingabe  $(m, n)$  solche Zahlen zurück.

## Erweiterter Euklidischer Algorithmus

**Eingabe:**  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:**  $x, y \in \mathbb{Z}$  mit  
 $\text{ggT}(m, n) = mx + ny$ .

Algorithmus Erw-Euklid( $m, n$ )

Wenn  $m | n$  dann  
gib  $(1, 0)$  zurück

sonst

sei  $(x', y') =$   
 $\text{Erw-Euklid}(n \bmod m, m)$ .

sei  $x := y' - x' \cdot \lfloor \frac{n}{m} \rfloor$

$y := x'$

gib  $(x, y)$  zurück.

# Beweis

**Beweis.** Zeigen nur zweite Aussage, aus der die erste folgt.

Der Beweis per Induktion über Zahl der rekursiven Aufrufe.

**Klar.** Algorithmus terminiert, da die Summe  $m + n$  in jedem Schritt kleiner wird.

Für  $m = 0$ , ist der Algorithmus offensichtlich korrekt.

Gelte nun  $m > 0$ .

Wende Induktionsannahme auf  $m' := (n \bmod m)$  und  $n' := m$  an.

Sei  $(x', y') := \text{Erw-Euklid}(m', n')$ , d.h.

$$\text{ggT}(n \bmod m, m) = x' \cdot (n \bmod m) + y' \cdot m.$$

Nach vorherigem Lemma gilt  $\text{ggT}(n \bmod m, m) = \text{ggT}(m, n)$ .

## Satz.

Für alle  $m, n \in \mathbb{N}$  mit  $m \leq n$  gibt es  $x, y \in \mathbb{Z}$  mit

$$\text{ggT}(m, n) = mx + ny.$$

Erw-Euklid liefert auf Eingabe  $(m, n)$  solche Zahlen zurück.

## Erweiterter Euklidischer Algorithmus

**Eingabe:**  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:**  $x, y \in \mathbb{Z}$  mit  
 $\text{ggT}(m, n) = mx + ny$ .

Algorithmus Erw-Euklid( $m, n$ )

Wenn  $m | n$  dann  
gib  $(1, 0)$  zurück

sonst

sei  $(x', y') =$   
 $\text{Erw-Euklid}(n \bmod m, m)$ .

sei  $x := y' - x' \cdot \lfloor \frac{n}{m} \rfloor$

$y := x'$

gib  $(x, y)$  zurück.

# Beweis

**Beweis.** Zeigen nur zweite Aussage, aus der die erste folgt.

Der Beweis per Induktion über Zahl der rekursiven Aufrufe.

**Klar.** Algorithmus terminiert, da die Summe  $m + n$  in jedem Schritt kleiner wird.

Für  $m = 0$ , ist der Algorithmus offensichtlich korrekt.

Gelte nun  $m > 0$ .

Wende Induktionsannahme auf  $m' := (n \bmod m)$  und  $n' := m$  an.

Sei  $(x', y') := \text{Erw-Euklid}(m', n')$ , d.h.

$$\text{ggT}(n \bmod m, m) = x' \cdot (n \bmod m) + y' \cdot m.$$

Nach vorherigem Lemma gilt  $\text{ggT}(n \bmod m, m) = \text{ggT}(m, n)$ .

Weiterhin gilt  $(n \bmod m) = n - m \lfloor \frac{n}{m} \rfloor$ .

## Satz.

Für alle  $m, n \in \mathbb{N}$  mit  $m \leq n$  gibt es  $x, y \in \mathbb{Z}$  mit

$$\text{ggT}(m, n) = mx + ny.$$

Erw-Euklid liefert auf Eingabe  $(m, n)$  solche Zahlen zurück.

## Erweiterter Euklidischer Algorithmus

**Eingabe:**  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:**  $x, y \in \mathbb{Z}$  mit  
 $\text{ggT}(m, n) = mx + ny$ .

Algorithmus Erw-Euklid( $m, n$ )

Wenn  $m | n$  dann  
gib  $(1, 0)$  zurück

sonst

sei  $(x', y') =$   
 $\text{Erw-Euklid}(n \bmod m, m)$ .

sei  $x := y' - x' \cdot \lfloor \frac{n}{m} \rfloor$

$y := x'$

gib  $(x, y)$  zurück.

# Beweis

**Beweis.** Zeigen nur zweite Aussage, aus der die erste folgt.

Der Beweis per Induktion über Zahl der rekursiven Aufrufe.

**Klar.** Algorithmus terminiert, da die Summe  $m + n$  in jedem Schritt kleiner wird.

Für  $m = 0$ , ist der Algorithmus offensichtlich korrekt.

Gelte nun  $m > 0$ .

Wende Induktionsannahme auf  $m' := (n \bmod m)$  und  $n' := m$  an.

Sei  $(x', y') := \text{Erw-Euklid}(m', n')$ , d.h.

$$\text{ggT}(n \bmod m, m) = x' \cdot (n \bmod m) + y' \cdot m.$$

Nach vorherigem Lemma gilt  $\text{ggT}(n \bmod m, m) = \text{ggT}(m, n)$ .

Weiterhin gilt  $(n \bmod m) = n - m \lfloor \frac{n}{m} \rfloor$ .

$$\begin{aligned}\text{ggT}(m, n) &= x' \cdot (n \bmod m) + y' \cdot m \\ &= x' \cdot \left(n - m \lfloor \frac{n}{m} \rfloor\right) + y' \cdot m \\ &= x' \cdot n - x' \cdot m \lfloor \frac{n}{m} \rfloor + y' \cdot m \\ &= m \cdot \left(y' - x' \cdot \lfloor \frac{n}{m} \rfloor\right) + x' \cdot n.\end{aligned}$$

Es gilt also

## Satz.

Für alle  $m, n \in \mathbb{N}$  mit  $m \leq n$  gibt es  $x, y \in \mathbb{Z}$  mit

$$\text{ggT}(m, n) = mx + ny.$$

Erw-Euklid liefert auf Eingabe  $(m, n)$  solche Zahlen zurück.

## Erweiterter Euklidischer Algorithmus

**Eingabe:**  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:**  $x, y \in \mathbb{Z}$  mit  
 $\text{ggT}(m, n) = mx + ny$ .

Algorithmus Erw-Euklid( $m, n$ )

Wenn  $m | n$  dann  
gib  $(1, 0)$  zurück

sonst

sei  $(x', y') =$   
 $\text{Erw-Euklid}(n \bmod m, m)$ .

sei  $x := y' - x' \cdot \lfloor \frac{n}{m} \rfloor$

$y := x'$

gib  $(x, y)$  zurück.

# Beweis

**Beweis.** Zeigen nur zweite Aussage, aus der die erste folgt.

Der Beweis per Induktion über Zahl der rekursiven Aufrufe.

**Klar.** Algorithmus terminiert, da die Summe  $m + n$  in jedem Schritt kleiner wird.

Für  $m = 0$ , ist der Algorithmus offensichtlich korrekt.

Gelte nun  $m > 0$ .

Wende Induktionsannahme auf  $m' := (n \bmod m)$  und  $n' := m$  an.

Sei  $(x', y') := \text{Erw-Euklid}(m', n')$ , d.h.

$$\text{ggT}(n \bmod m, m) = x' \cdot (n \bmod m) + y' \cdot m.$$

Nach vorherigem Lemma gilt  $\text{ggT}(n \bmod m, m) = \text{ggT}(m, n)$ .

Weiterhin gilt  $(n \bmod m) = n - m \lfloor \frac{n}{m} \rfloor$ .

$$\begin{aligned}\text{ggT}(m, n) &= x' \cdot (n \bmod m) + y' \cdot m \\ &= x' \cdot \left(n - m \lfloor \frac{n}{m} \rfloor\right) + y' \cdot m \\ &= x' \cdot n - x' \cdot m \lfloor \frac{n}{m} \rfloor + y' \cdot m \\ &= m \cdot \left(y' - x' \cdot \lfloor \frac{n}{m} \rfloor\right) + x' \cdot n.\end{aligned}$$

Es gilt also

## Satz.

Für alle  $m, n \in \mathbb{N}$  mit  $m \leq n$  gibt es  $x, y \in \mathbb{Z}$  mit

$$\text{ggT}(m, n) = mx + ny.$$

Erw-Euklid liefert auf Eingabe  $(m, n)$  solche Zahlen zurück.

## Erweiterter Euklidischer Algorithmus

**Eingabe:**  $m, n \in \mathbb{N}$  mit  $m \leq n$ .

**Ausgabe:**  $x, y \in \mathbb{Z}$  mit  
 $\text{ggT}(m, n) = mx + ny$ .

Algorithmus Erw-Euklid( $m, n$ )

Wenn  $m | n$  dann  
gib  $(1, 0)$  zurück

sonst

sei  $(x', y') =$   
 $\text{Erw-Euklid}(n \bmod m, m)$ .

sei  $x := y' - x' \cdot \lfloor \frac{n}{m} \rfloor$

$y := x'$

gib  $(x, y)$  zurück.