

Arithmetik

- Grundlegende Tatsachen über den Ring \mathbb{Z}
- Euklidischer Algorithmus
- Lösung ganzzahliger Gleichungssysteme
- Binärer euklidischer Algorithmus

1

- **grösster gemeinsamer Teiler** für $(a, b) \in \mathbb{Z} \times \mathbb{Z}, (a, b) \neq (0, 0)$ ist $d \in \mathbb{Z}_{>0}$ mit

$$d|a \wedge d|b \wedge (\forall c \in \mathbb{Z}_{>0} : c|a \wedge c|b \Rightarrow c|d)$$

Bezeichnung: $d = \text{ggT}(a, b)$

Tatsache: $d = \text{ggT}(a, b) = \max\{c \in \mathbb{Z}_{>0} : c|a \wedge c|b\}$

Konvention (GA): $\text{ggT}(0, 0) = 0$

- **kleinstes gemeinsames Vielfaches** für $(a, b) \in \mathbb{Z} \times \mathbb{Z}, (a, b) \neq (0, 0)$ ist $m \in \mathbb{Z}_{>0}$ mit

$$a|m \wedge b|m \wedge (\forall c \in \mathbb{Z}_{>0} : a|c \wedge b|c \Rightarrow m|c)$$

Bezeichnung: $m = \text{kgV}(a, b)$

Tatsache: $m = \text{kgV}(a, b) = \min\{c \in \mathbb{Z}_{>0} : a|c \wedge b|c\}$

3

Grundlegende Tatsachen über den Ring \mathbb{Z}

- $\langle \mathbb{Z}; +, \cdot \rangle$ ist ein nullteilerfreier Ring
- Divisionseigenschaft

$$\forall a \in \mathbb{Z}, b \in \mathbb{Z}_{>0} \exists q, r \in \mathbb{Z} : a = b \cdot q + r, 0 \leq r < b$$

- Bezeichnungen

$$q = \lfloor \frac{a}{b} \rfloor = (a \text{ div } b) \quad : \text{Quotient}$$

$$r = a - \lfloor \frac{a}{b} \rfloor \cdot b = (a \text{ mod } b) \quad : \text{Rest}$$

- Teilbarkeit

$$b \text{ teilt } a : b|a \Leftrightarrow (a \text{ mod } |b|) = 0$$

2

- $p \in \mathbb{Z}_{>1}$ ist **Primzahl**, wenn gilt

$$\forall a \in [1..p] : (a|p \Rightarrow a = 1 \vee a = p)$$

- **Euklid:**

$$p \text{ Primzahl} \Rightarrow (\forall a, b \in \mathbb{Z} : p|(a \cdot b) \Rightarrow p|a \vee p|b)$$

- **Euklid:** es gibt unendlich viele Primzahlen

- **Fundamentalsatz der Arithmetik** (GAUSS, GA Theorem 7.21)

Die Zerlegung natürlicher Zahlen in ihre Primteiler (mit Vielfachheiten) ist eindeutig, d.h.

zu jeder natürlichen Zahl $n > 1$ gibt es eindeutig bestimmte Primzahlen $p_1 < p_2 < \dots < p_k$ und Exponenten $e_1, e_2, \dots, e_k \in \mathbb{N}_{>0}$ mit

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

- Die Berechnung dieser Darstellung (Faktorisierung) ist mutmasslich ein algorithmisch sehr aufwendiges Problem.

4

- Sind $a, b \in \mathbb{Z}_{>0}$ mit

$$a = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdots p_k^{\alpha_k}, \quad b = p_1^{\beta_1} \cdot p_2^{\beta_2} \cdots p_k^{\beta_k}$$

mit Primzahlen $p_1 < p_2 < \dots < p_k$ und Exponenten

$\alpha_1, \alpha_2, \dots, \alpha_k, \beta_1, \beta_2, \dots, \beta_k \geq 0$, so gilt

$$\text{ggT}(a, b) = p_1^{\min\{\alpha_1, \beta_1\}} p_2^{\min\{\alpha_2, \beta_2\}} \cdots p_k^{\min\{\alpha_k, \beta_k\}}$$

$$\text{kgV}(a, b) = p_1^{\max\{\alpha_1, \beta_1\}} p_2^{\max\{\alpha_2, \beta_2\}} \cdots p_k^{\max\{\alpha_k, \beta_k\}}$$

ggT und kgV ganzer Zahlen lassen sich also prinzipiell mittels Faktorisierung berechnen — das ist aber keine effiziente Methode!

- $\text{ggT}(a, b) \cdot \text{kgV}(a, b) = a \cdot b$
- Ganze Zahlen a, b heißen **teilerfremd (relativ prim)**, wenn $\text{ggT}(a, b) = 1$.

5

Schema der Ausführung von Euklids Algorithmus

$$\begin{aligned} a_0 &:= a \\ a_1 &:= b \\ a_0 &= a_1 \cdot q_0 + a_2 && (q_0 \in \mathbb{Z}, 0 < a_2 < a_1) \\ a_1 &= a_2 \cdot q_1 + a_3 && (q_1 \in \mathbb{N}_+, 0 < a_3 < a_2) \\ a_2 &= a_3 \cdot q_2 + a_4 && (q_2 \in \mathbb{N}_+, 0 < a_4 < a_3) \\ &\vdots \\ a_{n-2} &= a_{n-1} \cdot q_{n-2} + a_n && (q_{n-2} \in \mathbb{N}_+, 0 < a_n < a_{n-1}) \\ a_{n-1} &= a_n \cdot q_{n-1} && (q_{n-1} \geq 2, a_{n+1} = 0) \end{aligned}$$

$\langle q_0, q_1, q_2, \dots, q_{n-1} \rangle$: Quotientenfolge
 $\langle a_0, a_1, a_2, \dots, a_n, 0 \rangle$: Restefolge

7

Euklidischer Algorithmus — Grundalgorithmus

Von EUKLID stammt ein effizientes Verfahren zur ggT-Berechnung mittels iterierter Division mit Rest:

```

rekursive Version:
Euclid-rek (int a, int b)
  if b = 0 then
    return(a)
  else
    return(Euclid-rek(b, a mod b))
  end if

```

```

iterative Version:
Euclid-iter (int a, int b)
  alpha := a, beta := b
  while beta != 0 do
    (alpha, beta) := (beta, alpha mod beta)
  end while
  return(alpha)

```

basierend auf

Theorem (GA, Thm. 7.19)

$$\forall a, b \in \mathbb{Z} : \text{ggT}(a, b) = \text{ggT}(b, a \bmod b)$$

6

Beispiel: $a = 57, b = 33$

$$\begin{aligned} 57 &= 33 \cdot 1 + 24 \\ 33 &= 24 \cdot 1 + 9 \\ 24 &= 9 \cdot 2 + 6 \\ 9 &= 6 \cdot 1 + 3 \\ 6 &= 3 \cdot 2 \end{aligned}$$

Quotientenfolge: $\langle q_0, q_1, \dots, q_4 \rangle = \langle 1, 1, 2, 1, 2 \rangle$

Restefolge: $\langle a_0, a_1, a_2, \dots, a_6 \rangle = \langle 57, 33, 24, 9, 6, 3, 0 \rangle$

8

- **Terminierung**

$$a_0 \geq a_1 > a_2 > \dots > a_n > a_{n+1} = 0 \text{ für ein } n \geq 1$$

- **Korrektheit**

$$\begin{aligned} \text{ggT}(a, b) &= \text{ggT}(a_0, a_1) \\ &= \text{ggT}(a_1, a_2) \\ &= \text{ggT}(a_2, a_3) \\ &= \dots = \text{ggT}(a_n, a_{n+1}) = \text{ggT}(a_n, 0) = a_n \end{aligned}$$

9

Eine alternative Betrachtung:

Euklids Algorithmus für $(a, b) = (a_0, a_1)$ erzeugt zu der Folge

$$a_i = a_{i+1} \cdot q_i + a_{i+2} \quad (0 \leq i \leq n)$$

von Divisionsschritten eine Quotientenfolge $\langle q_0, q_1, \dots, q_n \rangle$ mit

$$q_i \geq 1 \quad (0 \leq i < n) \quad \text{und} \quad q_n \geq 2$$

Es gilt

$$a_{i+2} \cdot (q_i + 1) < a_{i+1} \cdot q_i + a_{i+2} = a_i \quad (0 \leq i < n)$$

und somit

$$q_i + 1 < a_i/a_{i+2} \quad (0 \leq i < n) \quad \text{und} \quad q_n = a_n/a_{n+1} .$$

11

Effizienz wird gesichert durch den **Satz von Lamé** (1845):

Wenn die Berechnung von $\text{ggT}(a, b)$ für $a > b > 0$ genau k Aufrufe der Prozedur `Euclid-rek` (bzw. k Schleifendurchläufe der `while`-Schleife in `Euclid-iter`) erfordert, dann gilt $a \geq f_k$ und $b \geq f_{k-1}$, wobei f_i die i -te FIBONACCI-Zahl ist.

Aus den bekannten Aussagen über die FIBONACCI-Zahlen ergibt sich:

Sind $a, b \in \mathbb{N}$ mit $a \geq b$, dann ist die Zahl der Divisionschritte im Euklidischen Algorithmus für $a, b \leq 4.8 \cdot \log_{10}(a) + 2$.

10

Ausgehend von $a = a_0 \geq a_1 = b$ ergibt sich

$$\begin{aligned} 2^{n+1} &\leq q_n \prod_{i=0}^{n-1} (q_i + 1) < \frac{a_n}{a_{n+1}} \prod_{i=0}^{n-1} \frac{a_i}{a_{i+2}} \leq \frac{a_0 \cdot \dots \cdot a_n}{a_2 \cdot \dots \cdot a_{n+1} a_{n+1}} \\ &= \frac{a_0 a_1}{a_{n+1} a_{n+1}} \leq \left(\frac{a}{\text{ggT}(a, b)} \right)^2 \end{aligned}$$

- **Theorem** : Für $a, b \in \mathbb{N}$ benötigt die Berechnung von $\text{ggT}(a, b)$ höchstens

$$\lfloor 2 \cdot \log_2 \max(a, b) \rfloor + 1 \text{ Divisionsschritte}$$

(Linearität in der Problemgröße)

12

Untersuchung im logarithmischen Komplexitätsmodell:

$\ell_\beta(a)$: Grösse (Länge) von a bei der Darstellungen in Basis β

Jeder Divisionschritt $a_i = a_{i+1} \cdot q_i + a_{i+2}$ erfordert $\ell_\beta(a_{i+1}) \cdot \ell_\beta(q_i)$ Operationen in β -Arithmetik.

Gesamtaufwand, gemessen in Basis- β -Operationen für Euklids Algorithmus (a, b) :

$$\sum_{i=0}^n \ell_\beta(a_{i+1}) \cdot \ell_\beta(q_i)$$

und dies ist, unter der Annahme $a \geq b$, beschränkt durch

$$\begin{aligned} \ell_\beta(a_1) \cdot \left(\sum_{i=0}^{n-1} \ell_\beta(q_i + 1) + \ell_\beta(q_n) \right) &\sim \ell_\beta(b) \cdot \ell_\beta(q_n) \cdot \prod_{i=0}^{n-1} (q_i + 1) \\ &\leq 2 \ell_\beta(b) \cdot (\ell_\beta(a) - \ell_\beta(\text{ggT}(a, b)) + 1) \end{aligned}$$

13

Die algebraische Sicht:

- Für $a, b \in \mathbb{Z}$ ist

$$H_{a,b} = \{ a \cdot u + b \cdot v ; u, v \in \mathbb{Z} \}$$

eine Untergruppe (sogar ein "Ideal") von \mathbb{Z} ,

d.h. $x, y \in H_{a,b} \Rightarrow x - y \in H_{a,b}$ (und $u \cdot x \in H_{a,b}$ für alle $u \in \mathbb{Z}$)

- Divisionseigenschaft von $\mathbb{Z} \Rightarrow$ ist $H \neq \{0\}$ eine Untergruppe von \mathbb{Z} , so gilt

$$H = k \cdot \mathbb{Z} = \{ k \cdot u ; u \in \mathbb{Z} \}$$

wobei $k = \min\{h \in H ; h > 0\}$

- Für $a, b \in \mathbb{Z}$ mit $(a, b) \neq (0, 0)$ gilt $H_{a,b} = k \cdot \mathbb{Z}$ mit $k = a \cdot s + b \cdot t$ und es ist $k = \text{ggT}(a, b)$, denn
 - $a \in H_{a,b} \Rightarrow k | a$ und $b \in H_{a,b} \Rightarrow k | b$
 - $c | a \wedge c | b \Rightarrow c | (a \cdot s + b \cdot t) = k$
- Das liefert noch kein algorithmisches Verfahren zur Berechnung von (s, t)

15

Euklidischer Algorithmus — erweiterte Form

Eine fundamental wichtige Eigenschaft des ggT für ganze Zahlen ist:

der $\text{ggT}(a, b)$ lässt sich als Linearkombination von a und b mit ganzzahligen Koeffizienten darstellen (BÉZOUT-Beziehung)

$$\forall a, b \in \mathbb{Z} \exists s, t \in \mathbb{Z} : a \cdot s + b \cdot t = \text{ggT}(a, b)$$

Tatsächlich ist $\text{ggT}(a, b)$ die kleinste positive Zahl $\in \mathbb{Z}$, die sich als Linearkombination von a und b darstellen lässt.

14

EXTENDED_EUCLID (int a , int b)

```
{ /* returns (d, s, t), where d = gcd(a, b) = as + bt */
  if (b == 0) return (a, 1, 0);
  else
  {
    (d', s', t') = Extended_Euclid(b, a mod b);
    s = t';
    t = s' - t' * (a div b);
    return (d', s, t);
  }
}
```

16

Iterative Version des erweiterten Euklidischen Algorithmus:

seien $a \geq b > 0$

für $i = 0, 1, 2, \dots$ seien $\alpha^{(i)} = \langle s_i, t_i, a_i \rangle \in \mathbb{Z}^3$ und $q_i \in \mathbb{N}_+$ definiert durch

$$\alpha^{(0)} := \langle 1, 0, a \rangle$$

$$\alpha^{(1)} := \langle 0, 1, b \rangle$$

$$i := 1$$

while $a_i \neq 0$ **do**

$$q_{i-1} := \lfloor a_{i-1}/a_i \rfloor$$

$$\alpha^{(i+1)} := \alpha^{(i-1)} - q_{i-1} \cdot \alpha^{(i)}$$

$$i := i + 1$$

end while

17

Die Eigenschaften des erweiterten Euklidischen Algorithmus ergeben sich leicht aus einer Darstellung in Matrixschreibweise

$$\alpha = \langle \alpha_1, \alpha_2, \alpha_3 \rangle := \langle 1, 0, a \rangle$$

$$\beta = \langle \beta_1, \beta_2, \beta_3 \rangle := \langle 0, 1, b \rangle$$

while $\beta_3 \neq 0$ **do**

$$q := \alpha_3 \operatorname{div} \beta_3$$

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} := \begin{pmatrix} 0 & 1 \\ 1 & -q \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

end while

$$\langle s, t, d \rangle := \langle \alpha_1, \alpha_2, \alpha_3 \rangle$$

19

Beispiel $a = 57, b = 33$

$$\alpha^{(0)} = \langle 1, 0, 57 \rangle$$

$$\alpha^{(1)} = \langle 0, 1, 33 \rangle$$

$$q_0 = \lfloor 57/33 \rfloor = 1$$

$$\alpha^{(2)} = \alpha^{(0)} - q_0 \cdot \alpha^{(1)} = \langle 1, -1, 24 \rangle$$

$$q_1 = \lfloor 33/24 \rfloor = 1$$

$$\alpha^{(3)} = \alpha^{(1)} - q_1 \cdot \alpha^{(2)} = \langle -1, 2, 9 \rangle$$

$$q_2 = \lfloor 24/9 \rfloor = 2$$

$$\alpha^{(4)} = \alpha^{(2)} - q_2 \cdot \alpha^{(3)} = \langle 3, -5, 6 \rangle$$

$$q_3 = \lfloor 9/6 \rfloor = 1$$

$$\alpha^{(5)} = \alpha^{(3)} - q_3 \cdot \alpha^{(4)} = \langle -4, 7, 3 \rangle$$

$$q_4 = \lfloor 6/3 \rfloor = 2$$

$$\alpha^{(6)} = \alpha^{(4)} - q_4 \cdot \alpha^{(5)} = \langle 11, -19, 0 \rangle$$

BÉZOUT-Beziehung

$$57 \cdot (-4) + 33 \cdot 7 = 3 = \operatorname{ggT}(57, 33)$$

Notabene

$$57 \cdot 11 - 33 \cdot (-19) = 0, \quad 11 = \frac{33}{\operatorname{ggT}(57, 33)}, \quad 19 = \frac{57}{\operatorname{ggT}(57, 33)}$$

18

Es ist

$$\begin{pmatrix} \alpha^{(i)} \\ \alpha^{(i+1)} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_{i-1} \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & -q_0 \end{pmatrix} \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \end{pmatrix}$$

für $0 \leq i \leq n$;

daher ist

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \begin{pmatrix} a \\ b \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

eine Schleifeninvariante.

Mit $\alpha^{(i)} = \langle s_i, t_i, a_i \rangle$ ist

$$a \cdot s_i + b \cdot t_i = a_i \quad \text{für } 0 \leq i \leq n + 1$$

20

Daraus ergibt sich die BÉZOUT-Beziehung:

$$a \cdot s_n + b \cdot t_n = a_n = \text{ggT}(a, b)$$

sowie

$$a \cdot s_{n+1} + b \cdot t_{n+1} = a_{n+1} = 0$$

mit

$$s_{n+1} = (-1)^{n+1} \frac{b}{\text{ggT}(a, b)} \quad \text{und} \quad t_{n+1} = (-1)^n \frac{a}{\text{ggT}(a, b)}$$

21

Beispiel $a = 57, b = 33$:

Quotientenfolge : $\langle q_0, \dots, q_4 \rangle = \langle 1, 1, 2, 1, 2 \rangle$

$$\begin{aligned} \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} &= \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix} \\ \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} &= \begin{pmatrix} -1 & 2 \\ 3 & -5 \end{pmatrix} \\ \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} &= \begin{pmatrix} 3 & -5 \\ -4 & 7 \end{pmatrix} \\ \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} &= \begin{pmatrix} -4 & 7 \\ 11 & -19 \end{pmatrix} \end{aligned}$$

23

Beweis (Induktion über Anzahl der Divisionen im EA)

beachte: ist $\langle q_0, q_1, \dots, q_{n-1} \rangle$ die Quotientenfolge für EA(a_0, a_1), so ist

$\langle q_1, q_2, \dots, q_{n-1} \rangle$ die Quotientenfolge für EA(a_1, a_2), wobei

$\text{ggT}(a_0, a_1) = \text{ggT}(a_1, a_2)$, daher

$$\begin{aligned} \begin{pmatrix} s_n & t_n \\ s_{n+1} & t_{n+1} \end{pmatrix} &= \begin{pmatrix} 0 & 1 \\ 1 & -q_n \end{pmatrix} \dots \begin{pmatrix} 0 & 1 \\ 1 & -q_1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -q_0 \end{pmatrix} \\ &= \begin{pmatrix} s'_{n-1} & t'_{n-1} \\ s'_n & t'_n \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -q_0 \end{pmatrix} = \begin{pmatrix} s'_{n-1} & t'_{n-1} \\ \frac{(-1)^n a_2}{\text{ggT}(a_1, a_2)} & \frac{(-1)^{n+1} a_1}{\text{ggT}(a_1, a_2)} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -q_0 \end{pmatrix} \\ &= \begin{pmatrix} t'_{n-1} & s'_{n-1} - q_0 t'_{n-1} \\ \frac{(-1)^{n+1} a_1}{\text{ggT}(a_0, a_1)} & \frac{(-1)^n a_2}{\text{ggT}(a_0, a_1)} - q_0 \frac{(-1)^{n+1} a_1}{\text{ggT}(a_0, a_1)} \end{pmatrix} \\ &= \begin{pmatrix} t'_{n-1} & s'_{n-1} - q_0 t'_{n-1} \\ \frac{(-1)^{n+1} a_1}{\text{ggT}(a_0, a_1)} & \frac{(-1)^n a_0}{\text{ggT}(a_0, a_1)} \end{pmatrix} \end{aligned}$$

22

EUKLIDIS Algorithmus (in der erweiterten Form von BÉZOUT) behandelt ein fundamentales Problem:

Lösen linearer Gleichungen in \mathbb{Z}

denn es gilt für $a, b, c \in \mathbb{Z}$ mit $(a, b) \neq (0, 0)$:

$$\exists u, v \in \mathbb{Z} : a \cdot u + b \cdot v = c \Leftrightarrow \text{ggT}(a, b) \mid c$$

und wenn diese Bedingung erfüllt ist, ist die Menge der ganzzahligen Lösungen der Geradengleichung

$$a \cdot x + b \cdot y = c$$

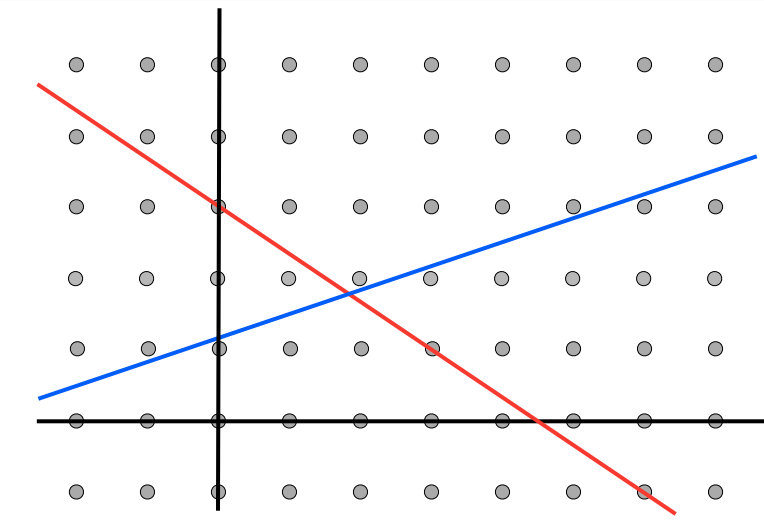
gegeben durch

$$(x, y) = \frac{c}{d} \cdot (s, t) + \frac{k}{d} \cdot (-b, a) \quad (k \in \mathbb{Z})$$

wobei s, t Bézout-Koeffizienten für (a, b) und $d = \text{ggT}(a, b)$ sind, d.h.,

$$a \cdot s + b \cdot t = d$$

24



25

Beispiel

- $57 \cdot x + 33 \cdot y = -9$

Lösungsmenge:

$$\begin{aligned} (x, y) &= \frac{-9}{3} \cdot (-4, 7) + \frac{k}{3} \cdot (-33, 57) \\ &= (12 - 11 \cdot k, -21 + 19 \cdot k) \quad (k \in \mathbb{Z}) \end{aligned}$$

- $57 \cdot x + 33 \cdot y = -8$

Lösungsmenge: \emptyset

Wichtige Bemerkung:

man kann diese Idee ausbauen zu einem Verfahren, um die Menge der ganzzahligen Lösungen von linearen Gleichungssysteme mit ganzzahligen Koeffizienten zu berechnen.

27

Zum Beweis:

- es gilt

$$H_{a,b} = \{ a \cdot u + b \cdot v ; u, v \in \mathbb{Z} \} = \text{ggT}(a, b) \cdot \mathbb{Z}$$

daher ist $a \cdot x + b \cdot y = c$ in \mathbb{Z} lösbar $\Leftrightarrow c \in H_{a,b}$ (d.h. $\text{ggT}(a, b) \mid c$)

- falls Lösbarkeit gegeben und $d = \text{ggT}(a, b)$

$$a \cdot x + b \cdot y = c \Leftrightarrow \frac{a}{d} \cdot x + \frac{b}{d} \cdot y = \frac{c}{d}$$

wobei nun $\text{ggT}(a/d, b/d) = 1$, d.h. es genügt, den Fall $\text{ggT}(a, b) = 1$ zu betrachten

- mit BÉZOUT-Koeffizienten s, t , d.h. $a \cdot s + b \cdot t = 1$ gilt

$$a \cdot (c \cdot s) + b \cdot (c \cdot t) = c$$

d.h. $(x_0, y_0) = (c \cdot s, c \cdot t)$ ist spezielle Lösung von $a \cdot x + b \cdot y = c$

- $(x, y) = (x_0 + X, y_0 + Y)$ ist genau dann Lösung von $a \cdot x + b \cdot y = c$, wenn (X, Y) Lösung von $a \cdot X + b \cdot Y = 0$
- die Lösungen von $a \cdot X + b \cdot Y = 0$ sind genau die $(k \cdot b, -k \cdot a)$ mit $k \in \mathbb{Z}$ — hierfür benötigt man $\text{ggT}(a, b) = 1$

26

- (1) $57x + 33y = -9$ die zu lösende Gleichung
- (2) $x + y = z$ Division mit $33 = \min\{57, 33\}$, neue Variable z
- (3) $24x + 33z = -9$ (1) - $33 \cdot$ (2)
- (4) $x + z = u$ Division mit $24 = \min\{24, 33\}$, neue Variable u
- (5) $24u + 9z = -9$ (3) - $24 \cdot$ (4)
- (6) $2u + z = v$ Division mit $9 = \min\{24, 9\}$, neue Variable v
- (7) $6u + 9v = -9$ (5) - $9 \cdot$ (6)
- (8) $u + v = w$ Division mit $6 = \min\{6, 9\}$, neue Variable w
- (9) $6w + 3v = -9$ (7) - $6 \cdot$ (8)
- (10) $2w + v = -3$ Division mit $\min\{3, 6\} = 3$
- (11) $v = -3 - 2w$ aus (10)
- (12) $u = 3w + 3$ aus (11) und (8)
- (13) $z = -9 - 8w$ aus (12) und (6)
- (14) $x = 11w + 12$ aus (13) und (4)
- (15) $y = -21 - 19w$ aus (14) und (2)

28

Das Verfahren funktioniert auch für Gleichungen mit mehreren Variablen

- (1) $8x - 7y - 5z = 2$ zu lösende Gleichung
- (2) $x - y - z = u$ Division mit 5, neue Variable u
- (3) $3x - 2y + 5u = 2$ (1) $- 5 \cdot$ (2)
- (4) $x - y + 2u = v$ Division mit 2, neue Variable v
- (5) $x + 2v + u = 2$ (3) $- 2 \cdot$ (4)
- (6) $x = 2 - u - 2v$ aus (5)
- (7) $y = 2 + u - 3v$ aus (6) und (4)
- (8) $z = -3u + v$ aus (7) und (2)

Die ganzzahlige Lösungsmenge der Gleichung $8x - 7y - 5z = 2$ ist also

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -1 & -2 \\ 1 & -3 \\ -3 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix} \quad u, v \in \mathbb{Z}$$

Die ganzzahlige Lösungsmenge der Gleichung $57x + 33y = -9$ ist also

$$(x, y) = (12, -21) + w \cdot (11, -19) \quad \text{mit } w \in \mathbb{Z}$$

Zur Lösung linearer Gleichungen über \mathbb{Z}

Gesucht ganzzahlige Lösungsmenge von

$$(*) \quad a_1 \cdot x_1 + a_2 \cdot x_2 + \dots + a_n \cdot x_n = b$$

wobei $a_1, \dots, a_n, b \in \mathbb{Z}$, nicht alle $a_i = 0$

– ohne Einschränkung sei $a_1 > 0$ und $a_1 = \min_{1 \leq i \leq n} \{|a_i|; a_i \neq 0\}$

– Fall 1: $a_1 = 1$

dann ist die allgemeine Lösung von (*) gegeben durch

$$(b - a_2 \cdot x_2 - \dots - a_n \cdot x_n, x_2, \dots, x_n) \quad \text{mit } (x_2, \dots, x_n) \in \mathbb{Z}^{n-1}$$

– Fall 2a: $a_1 > 1$, alle a_i durch a_1 teilbar

dann ist (*) lösbar $\Leftrightarrow a_1 \mid b$ und Lösung ergibt sich aus

$$\frac{a_1}{a_1} \cdot x_1 + \frac{a_2}{a_1} \cdot x_2 + \dots + \frac{a_n}{a_1} \cdot x_n = \frac{b}{a_1}$$

→ Fall 1.

– Fall 2a: $a_1 > 1$, nicht alle a_i durch a_1 teilbar

Bezeichnungen (\div = ganzzahlige Division)

$$\begin{aligned} \mathbf{a} &= \langle a_1, a_2, \dots, a_n \rangle \\ \mathbf{x} &= \langle x_1, x_2, \dots, x_n \rangle \\ (\mathbf{a} \div a_1) &= \langle \underbrace{a_1 \div a_1}_{=1}, a_2 \div a_1, \dots, a_n \div a_1 \rangle \\ (\mathbf{a} \bmod a_1) &= \langle \underbrace{a_1 \bmod a_1}_{=0}, \underbrace{a_2 \bmod a_1, \dots, a_n \bmod a_1}_{\text{nicht alle } =0} \rangle \end{aligned}$$

Zu lösende Gleichung (\bullet = Skalarprodukt)

$$(*) \quad \mathbf{a} \bullet \mathbf{x} = b$$

Divisionsbeziehung

$$\mathbf{a} = a_1 \cdot (\mathbf{a} \div a_1) + (\mathbf{a} \bmod a_1)$$

33

Das Verfahren lässt sich für die Bestimmung der ganzzahligen Lösungen ganzzahliger Gleichungssysteme verwenden.

Beispiel (vgl. KNUTH, TAOCP, Abschnitt 4.5.2)

1. zu lösendes System

$$\begin{aligned} 10w + 3x + 3y + 8z &= 1 \\ 6w - 7x - 5z &= 2 \end{aligned}$$

2. Elimination von y mittels der ersten Gleichung und Einführung einer neuen Variablen t_1

$$\begin{aligned} 3w + x + y + 2z &= t_1 \\ w + 3t_1 + 2z &= 1 \end{aligned}$$

3. Elimination von w aus zweiter Gleichung

$$\begin{aligned} 6(1 - 3t_1 - 2z) - 7x - 5z &= 2 \\ 18t_1 + 7x + 17z &= 4 \end{aligned}$$

35

Daher

$$\mathbf{a} \bullet \mathbf{x} = a_1 \cdot \underbrace{(\mathbf{a} \div a_1) \bullet \mathbf{x}}_y + (\mathbf{a} \bmod a_1) \bullet \mathbf{x}$$

Betrachte y als neue Variable mit

$$(**) \quad y = x_1 + \lfloor \frac{a_2}{a_1} \rfloor \cdot x_2 + \dots + \lfloor \frac{a_n}{a_1} \rfloor \cdot x_n$$

Mit

$$\begin{aligned} \tilde{\mathbf{a}} &= \langle a_1, a_2 \bmod a_1, \dots, a_n \bmod a_1 \rangle \\ \tilde{\mathbf{x}} &= \langle y, x_2, \dots, x_n \rangle \end{aligned}$$

gilt: die Lösungen $\langle x_1, x_2, \dots, x_n \rangle$ von $(*)$ entsprechen ein-eindeutig den Lösungen $\langle y, x_2, \dots, x_n \rangle$ von

$$\tilde{\mathbf{a}} \bullet \tilde{\mathbf{x}} = b$$

vermöge $(**)$.

Wichtig (für die Terminierung):

$$\min\{a_1, a_2 \bmod a_1, \dots, a_n \bmod a_1\} < a_1$$

34

4. Elimination von x durch Einführung einer neuen Variablen t_2

$$\begin{aligned} 2t_1 + x + 2z &= t_2 \\ 4t_1 + 7t_2 + 3z &= 4 \end{aligned}$$

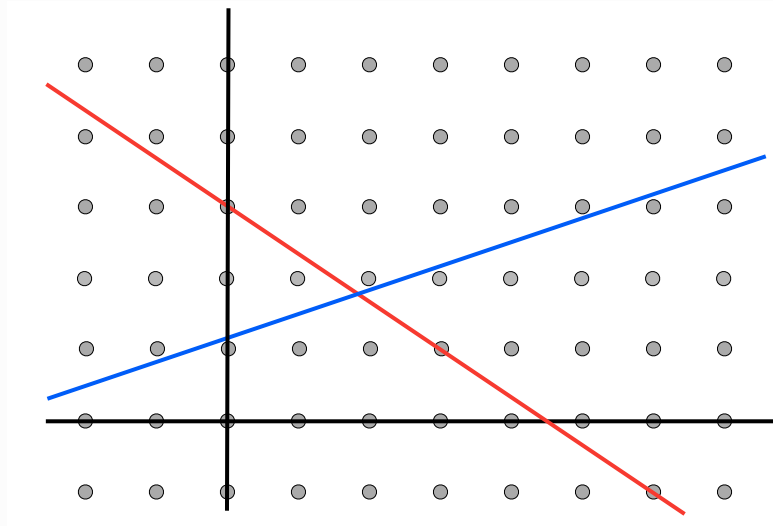
5. Elimination von z durch Einführung einer neuen Variablen t_3

$$\begin{aligned} t_1 + 2t_2 + z &= t_3 \\ t_1 + t_2 + 3t_3 &= 4 \end{aligned}$$

6. Auflösen nach t_2 und Rückwärtseinsetzen liefert

$$\begin{aligned} w &= 17 - 5t_1 - 14t_3 \\ x &= 20 - 5t_1 - 17t_3 \\ y &= -55 + 19t_1 + 45t_3 \\ z &= -8 + t_1 + 7t_3 \end{aligned}$$

36



37

```

BINARY_EUCLID (int a, int b)
{ /* returns gcd(a, b) provided that a, b ∈ ℕ */
  int d = 1;
  while ((a mod 2 == 0) && (b mod 2 == 0))
  {
    a = a/2; b = b/2; d = 2 · d;
  }
  while (a ≠ 0)
  {
    /* a or b is odd */
    while (a mod 2 == 0) a = a/2;
    while (b mod 2 == 0) b = b/2;
    /* a and b are odd */
    if (a < b) swap(a, b)
    a = a - b;
  }
  return (b · d);
}

```

39

Der binäre Euklidische Algorithmus

macht sich die Binärdarstellung der Zahlen zu Nutze und realisiert die Berechnung des ggT mittels Subtraktion und Division durch Zweierpotenzen (= shift).

Er benützt die (offensichtlichen) Beziehungen

$$\begin{aligned}
 \text{ggT}(2a, 2b) &= 2 \cdot \text{ggT}(a, b) && (a, b \in \mathbb{N}) \\
 \text{ggT}(2a, b) &= \text{ggT}(a, b) && (a, b \in \mathbb{N}, b \text{ ungerade}) \\
 \text{ggT}(a, b) &= \text{ggT}(a - b, b) && (a, b \in \mathbb{N}, a \geq b, \text{ beide ungerade})
 \end{aligned}$$

38

Beispiel: $a = 40902, b = 24140$

a	b	\rightarrow	a	b	
40902	24140	\rightarrow	20451	12770	$\Rightarrow d = 2$
20451	12770	\rightarrow	20451	6035	
14416	6035	\rightarrow	901	6035	
5134	901	\rightarrow	2567	901	
1666	901	\rightarrow	833	901	
68	833	\rightarrow	17	833	
816	17	\rightarrow	51	17	
34	17	\rightarrow	17	17	
0	17	\rightarrow			

$$\Rightarrow \text{ggT}(40902, 24140) = 2 \cdot 17 = 34$$

40

- Termination und Korrektheit von `BINARY_EUCLID` ergeben sich aus den drei verwendeten Gleichungen für den ggT
- Bei jedem Durchlauf der äusseren while-Schleife wird mindestens eines der beiden Argumente halbiert $\Rightarrow O(\log(a+b))$ Schleifendurchläufe
- Jeder Schleifendurchlauf erfordert Operationen (Vergleiche, Zuweisungen, Divisionen durch 2, Subtraktion), die sich insgesamt in $O(\log(a+b))$ Bit-Operationen ausführen lassen
- `BINARY_EUCLID` berechnet den ggT von natürlichen Zahlen mit $O(\log^2(a+b))$ Bit-Operationen (HEUN, *Grundlegende Algorithmen*, Abschnitt 7.1; siehe auch KNUTH, *The Art of Computer Programming*, vol. 2, *Seminumerical Algorithms*, Abschnitt 4.5.2, und BACH/SHALLIT, *Algorithmic Number Theory*, vol.1 *Efficient Algorithms*, Kapitel 4)

41

- die wesentliche Änderung gegenüber der iterativen Version der Euklidischen Algorithmus
 - an Stelle der exakten Quotienten $\lfloor d'/d \rfloor$ wird mit den angenäherten Quotienten

$$c = 2^{\max(\ell(d') - \ell(d) - 1, 0)}$$

gerechnet (\rightarrow shift-Operationen). Dabei gilt

$$\frac{1}{4} \cdot \lfloor d'/d \rfloor \leq 2^{\max(\ell(d') - \ell(d) - 1, 0)} \leq \lfloor d'/d \rfloor$$

- Konsequenz für die Bit-Komplexität: auch `BINARY_EXTENDED_EUCLID` verwendet nur $O(\log^2(a+b))$ Bit-Operationen.

43

```

BINARY_EXTENDED_EUCLID (int a, int b)
{ /* returns (d, s, t), where d = gcd(a, b) = a · s + b · t */
  (d, s, t) = (a, 1, 0);
  (d', s', t') = (b, 0, 1);
  while (d' ≠ 0)
  {
    if (d' < d) swap((d, s, t), (d', s', t'))
    c = 1 << max(ℓ(d') - ℓ(d) - 1, 0); /*c = 2max(ℓ(d')-ℓ(d)-1,0) */
    (d', s', t') = (d', s', t') - c(d, s, t);
  }
  return (d, s, t);
}

```

42

a	1023	$\ell(a) = 10$	1	1	1	1	1	1	1	1	1	1	1
b	15	$\ell(b) = 4$	0	0	0	0	0	0	1	1	1	1	
$x = b * 2^{\ell(a) - \ell(b) - 1}$	480		0	1	1	1	1	0	0	0	0	0	
$c = a - x$	543	$\ell(c) = 10$	1	0	0	0	0	1	1	1	1	1	
$y = b * 2^{\ell(c) - \ell(b) - 1}$	480		0	1	1	1	1	0	0	0	0	0	
$d = c - y$	63	$\ell(d) = 6$	0	0	0	0	1	1	1	1	1	1	
$z = b * 2^{\ell(d) - \ell(b) - 1}$	30		0	0	0	0	0	1	1	1	1	0	
$e = d - z$	33	$\ell(e) = 6$	0	0	0	0	1	0	0	0	0	1	
$u = b * 2^{\ell(e) - \ell(b) - 1}$	30		0	0	0	0	0	1	1	1	1	0	
$f = e - u$	3	$\ell(f) = 2$	0	0	0	0	0	0	0	0	1	1	
$v = f * 2^{\ell(b) - \ell(f) - 1}$	6		0	0	0	0	0	0	0	1	1	0	
$g = b - v$	9	$\ell(g) = 4$	0	0	0	0	0	0	1	0	0	1	
$w = f * 2^{\ell(g) - \ell(f) - 1}$	6		0	0	0	0	0	0	0	1	1	0	
$h = g - w$	3	$\ell(h) = 2$	0	0	0	0	0	0	0	0	1	1	

44