

A-5 Modulare Arithmetik

Satz A.49 (Teilung mit Rest)

In $\mathcal{M} = \mathbb{Z}$ gibt es für jedes Paar $a, m \in \mathcal{M}$ mit $m > 0$ genau ein Paar $q, r \in \mathcal{M}$, so dass gilt:

$$a = qm + r \quad \wedge \quad 0 \leq r < m > 0.$$

Dabei wird r Rest genannt, q ist der Quotient.



Definition A.50 (Modulbezeichnung, Teilbarkeit, Primzahl)

(i) Da der Rest r oft wichtiger ist als der Quotient q schreibt man

$$r = r_m(a) = a \bmod m$$

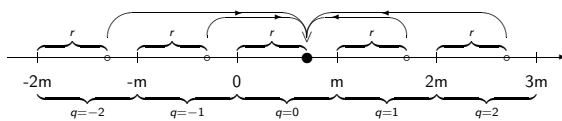
(sprich: r gleich "a modulo m").

(ii) Offenbar gilt $r_m(a) = 0$ genau dann wenn a durch m teilbar ist. Dann schreibt man $m|a$ (sprich "m teilt a").

(iii) Folgt aus $m|a$ immer $m \in \{1, a\}$ und ist $a \neq 1$, so heißt a **Primzahl**.



Zahlengerade



Beispiel A.51

$$7 \bmod 3 = r_3(7) = 1 \quad (q = 2)$$

Beispiel A.52

$$-13 \bmod 5 = r_5(-13) = 2 \quad (q = -3)$$

Bemerkung:

In der Programmiersprache C wird mod durch das Prozentzeichen % definiert:

$$a \% m = a - m(a/m) \quad \text{für } a \geq 0 < m.$$

Da das Vorzeichen des Restes für negative a abhängig von der Implementation (also dem verwendeten Compiler) ist, gilt obige Gleichung nicht unbedingt.

Es erfolgt aber immer eine Rundung in Richtung Null:

$$a/m = -(-a/m) \quad \text{für } a < 0 < m.$$



Satz A.53 (Modulare Arithmetik)

In $\mathbb{Z}_m \simeq \{0, 1, \dots, m-1\}$ wird durch

$$a +_m b := (a + b) \bmod m \equiv r_m(a + b)$$

und

$$a *_m b := (a * b) \bmod m \equiv r_m(a * b)$$

eine kommutative Ringstruktur definiert.

Bemerkung

Bei komplexeren Ausdrücken ohne Division kann erst in \mathbb{Z} gerechnet und nur zum Schluß auf $[0, m-1]$ modularisiert werden.



Satz A.54 (Fermat(1640))

Falls m prim gilt für alle $0 \leq a < m$

$$a^m = a \bmod m.$$

Ist m kein Teiler von a , gilt $a^{m-1} \equiv 1 \pmod{p}$.

Korollar A.55

\mathbb{Z}_m ist genau dann ein Integritätsbereich und damit nach Satz A.30 ein Körper, wenn m eine Primzahl ist. Dann gilt für alle $a \in \mathbb{Z}_m$

$$a^{-1} = a^{m-2}.$$

Beispiel A.56

In $\mathbb{Z}_5 = \{\bar{0}, \bar{1}, \bar{2}, \bar{3}, \bar{4}\}$ gilt:

$$\begin{aligned} \bar{1}^{-1} &= \bar{1} * \bar{1} * \bar{1} = 1 \bmod 5 = \bar{1} & \implies & \bar{1} * \bar{1} = 1 \bmod 5 = \bar{1} \\ \bar{2}^{-1} &= \bar{2} * \bar{2} * \bar{2} = 8 \bmod 5 = \bar{3} & \implies & \bar{2} * \bar{3} = 6 \bmod 5 = \bar{1} \\ \bar{3}^{-1} &= \bar{3} * \bar{3} * \bar{3} = 27 \bmod 5 = \bar{2} & \implies & \bar{3} * \bar{2} = 6 \bmod 5 = \bar{1} \\ \bar{4}^{-1} &= \bar{4} * \bar{4} * \bar{4} = 64 \bmod 5 = \bar{4} & \implies & \bar{3} * \bar{4} = 16 \bmod 5 = \bar{1} \end{aligned}$$



Multiplikation in \mathbb{Z}_5

*	$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$
$\bar{0}$	$\bar{0}$	$\bar{0}$	$\bar{0}$	$\bar{0}$	$\bar{0}$
$\bar{1}$	$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$
$\bar{2}$	$\bar{0}$	$\bar{2}$	$\bar{4}$	$\bar{1}$	$\bar{3}$
$\bar{3}$	$\bar{0}$	$\bar{3}$	$\bar{1}$	$\bar{4}$	$\bar{2}$
$\bar{4}$	$\bar{0}$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$



Addition und Subtraktion in \mathbb{Z}_5

Subtraktion

$+$ \ $-$	$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$
$\bar{0}$	$\bar{0}$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$
$\bar{1}$	$\bar{1}$	$\bar{2}$	$\bar{4}$	$\bar{3}$	$\bar{2}$
$\bar{2}$	$\bar{2}$	$\bar{3}$	$\bar{4}$	$\bar{4}$	$\bar{3}$
$\bar{3}$	$\bar{3}$	$\bar{4}$	$\bar{0}$	$\bar{1}$	$\bar{4}$
$\bar{4}$	$\bar{4}$	$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$

Addition



Division in \mathbb{Z}_5

/	$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	$\bar{4}$
$\bar{0}$	—	$\bar{0}$	$\bar{0}$	$\bar{0}$	$\bar{0}$
$\bar{1}$	—	$\bar{1}$	$\bar{3}$	$\bar{2}$	$\bar{4}$
$\bar{2}$	—	$\bar{2}$	$\bar{1}$	$\bar{4}$	$\bar{3}$
$\bar{3}$	—	$\bar{3}$	$\bar{4}$	$\bar{1}$	$\bar{2}$
$\bar{4}$	—	$\bar{4}$	$\bar{2}$	$\bar{3}$	$\bar{1}$

◀ ▶ ↺ ↻ 🔍

Anwendung: Hashing

Motivation:

Angenommen eine Firma hat allen ihren Angestellten eine 10 stellige Personalnummer k zugeordnet. Sie erwartet aber nie mehr als $m = 1000$ Angestellte zu haben und hat deshalb eine Registratur mit 1000 durchnummerierten Ablagen angelegt. Um schnell auf diese zugreifen zu können, sucht sie für $n = 10^{10}$ eine sogenannte **Hashfunktion**

$$h : \{1, 2, \dots, n\} \longrightarrow \{0, 1, 2, \dots, m-1\},$$

so daß möglichst alle zu irgendeinem Zeitpunkt tatsächlich vorhandenen Personalnummern k einen "eigenen" Funktionswert $h(k)$ haben. Da die Menge \mathcal{K} der vorhandene k aus datenschutzrechtlichen Gründen nie bekannt ist und sich zudem durch Personalfluktuatun ändern kann, lässt sich für a priori gewählte h eine Kollision

$$h(k') = h(k) \quad \text{mit} \quad k \neq k' \quad \text{und} \quad k, k' \leq n$$

nicht immer verhindern.

◀ ▶ ↺ ↻ 🔍

Fortsetzung: Hashing

Das gilt auch für die einfache **Hashfunktion**

$$h(k) = k \bmod p \quad \text{mit} \quad p \geq m,$$

wobei p häufig als Primzahl gewählt wird.

Um für ein k mit einer bereits durch ein k' belegten Speicheradresse $h(k) = h(k')$ ein freies Ablagefach zu finden, wird in der Nähe von $h(k)$ **sondiert**.

Beim **quadratischen Sondieren** durchsucht man die Adressen

$$(h(k) + i^2) \bmod p \quad \text{und} \quad (h(k) - i^2) \bmod p$$

für $i = 1, 2, \dots, (p-1)/2$, bis freies Fach gefunden wurde.

Setzt man voraus, daß mindestens ein freies Fach vorhanden ist, garantiert der folgende Satz den Erfolg des **quadratischen Sondierens**.

◀ ▶ ↺ ↻ 🔍

Satz A.57 (siehe Hartmann 4.24)

Ist p eine Primzahl mit $p \bmod 4 = 3$ so gilt:

$$\{\pm i^2 \bmod p : i = 1, 2, \dots, (p-1)/2\} = \{1, 2, \dots, p-1\}$$

Mit anderen Worten: Alle Adressen werden durchsucht.

Beispiel A.58 ($p = 11$)

i	1	2	3	4	5
$i^2 \bmod 11$	1	4	9	5	3
$-i^2 \bmod 11$	10	7	2	6	8

◀ ▶ ↺ ↻ 🔍