2. Ganze Zahlen

Natürliche Zahlen

$$\mathbb{N} := \{0, 1, 2, 3, 4, \dots\}$$

$\gneq 0$

Positive natürliche Zahlen

$$\mathbb{N}^+ := \mathbb{N} \setminus \{0\} = \{1, 2, 3, 4, \dots\}$$

Ganze Zahlen

$$\mathbb{Z} := \{\dots -2, -1, 0, 1, 2, 3, \dots\}$$

$$= \{0, 1, -1, 2, -2, 3, -3, \dots\}$$

Primzahlen $\mathbb{P} \subseteq \mathbb{N}$

$$\mathbb{P} = \{2, 3, 5, 7, 11, 13, 17, \dots\}$$

Primzahlen sind $\neq 1$ und
nur durch 1 und sich selbst teilbar
(ohne Rest).

Für $a \in \mathbb{N}$, $b \in \mathbb{N}^+$ sagen wir

b teilt a ohne Rest, Schreibweise $b|a$

$:\Longleftrightarrow$

Es gibt ein $q \in \mathbb{N}$, so daß

$$a = q \cdot b \quad (\text{auch } a : b = q)$$

Etwa $3|9$, da $9 = 3 \cdot 3$, $3|21$

aber nicht $3|20$, Schreibweise $3 \nmid 20$

$3|0$, $7|0$, aber $0|7$ nicht definiert.

Es gilt nicht $9|3$ aber $9|0$.

Es gilt zwar $3 \nmid 20$, aber

$$20 = 6 \cdot 3 + \underset{\substack{\uparrow \\ \text{Rest}}}{1} \quad (20 : 6 = 3 \text{ Rest } 1)$$

Eine Verallgemeinerung des Teilbarkeitsbegriffs ist Teilbarkeit mit Rest:

Für $a \in \mathbb{N}$, $b \in \mathbb{N}^+$, r mit $0 \leq r \lneqq b$

b teilt a mit Rest r

$:\Longleftrightarrow$

Es gibt ein $q \in \mathbb{N}$, so daß

$$a = b \cdot q + r \quad (\text{auch } a : b = q \text{ Rest } r)$$

5 teilt 21 mit Rest 1

$$21 = 4 \cdot 5 + 1.$$

Beachte obwohl $21 = 4 \cdot 5 + 6$ nicht 5 teilt 21 mit Rest 6. Ebenso $21 = 6 \cdot 5 - 9$, da kein $0 \leq r \lneq 5$.

Zwei Bezeichnungen in diesem Zusammenhang: $a \in \mathbb{N}$, $b \in \mathbb{N}^+$ (Vorsicht nicht bei $a<0$!)

Mod(a,b) ist der Rest r (a%b Java)

Div(a,b) ist der Quotient q. (a/b Java)

Statt r ist der Rest sagt man auch: r ist der Modulus von a bei Division durch b. Das Div(a,b) nennen wir auch Quotient von a und b.

Kann es 2 verschiedene Reste oder Quotienten von a bei Division durch b geben? Eindeutigkeit von Quotient und Rest.

Man beachte, nicht alles ist eindeutig: Sei $a \in \mathbb{N}$. Dann ist die Wurzel von a das $b \in \mathbb{Z}$, so daß $a = b^2$. Dann $4 = 2^2$, $4 = (-2)^2$.

Also wie ist das bei den Resten und Quotienten? Sei $a \in \mathbb{N}$, $b \in \mathbb{N}^+$ und $0 \leq r \lneq b$

$$a = q \cdot b + r.$$

Gibt es einen weiteren Rest, r' und Quotienten q', dann ebenfalls

2.6

$$a = \underbrace{\underset{\in \mathbb{N}}{c'}}\cdot \overset{\in \mathbb{N}}{b} + \underbrace{r'}_{0 \le r' \lneq b.}$$

Also ist

$$c \cdot b + r = c' \cdot b + r' \; (= a)$$

Also ist

$$r - r' = (c - c') \cdot b .$$

Da $0 \le r \lneq b$ und $0 \le r' \lneq b$ ist

$$\underbrace{-(b-1)}_{= -b+1} \le r - r' \le b-1$$

das heißt eben $|r - r'| \le b-1$.

Andererseits ist $c - c' \in \mathbb{Z}$. Also

$(c-c') b$ kann nur folgende Werte

annehmen: $0 \cdot b,\ 1 \cdot b,\ (-1) \cdot b,\ 2 \cdot b,\ (-2) \cdot b, \dots$

Also ist die einzige Möglichkeit die bleibt $q = q'$ und dann muß eben auch $r = r'$ sein.

Also: Eindeutigkeit von $Mod(a,b)$, $Div(a,b)$.

Existenz: Da $b \gneq 0$ ist, ist irgendwann das erste Mal

$$b + b + \dots + b \geq a .$$

Ist $b + \dots + b = a$, dann

$Div(a,b) = \#\, b$'s in der Summe

$Mod(a,b) = 0$.

Ist $b + \dots + b \gneq a$, dann

$Div(a,b) = (\#\, b\text{'s}) - 1$

$Mod(a,b) = a - Div(a,b)$.

Es ist zum Beispiel

$$\mathrm{Mod}(20,8) = 4 \quad \text{und} \quad \mathrm{Mod}(52,8) = 4$$

In diesem Falle wird die Mod-Schreibweise auch eingesetzt. Man schreibt dann

$$20 = 52 \bmod 8 \quad \text{oder} \quad 20 \equiv 52 \bmod 8$$

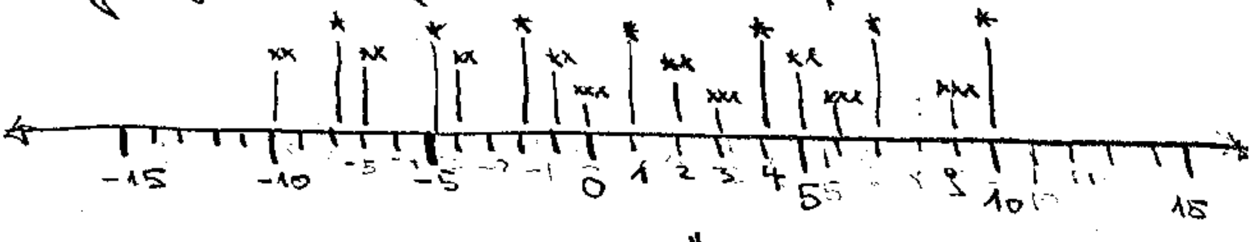
„gleicher Rest modulo 8."

Schließlich ist auch $a < 0$ möglich:

$$\mathrm{Div}(-1,7) = -1 \qquad \mathrm{Mod}(-1,7) = +6$$

$$-1 = (-1)\cdot 7 + 6$$ ← Beachte $0 \le r \le 7$.

Grafisch: Gleichheit mod 3.



* entspricht „$3\cdot\mathbb{Z}+1$"   xx $3\cdot\mathbb{Z}+2$

xxx entspricht $3\cdot\mathbb{Z} = 3\cdot\mathbb{Z}+3$.

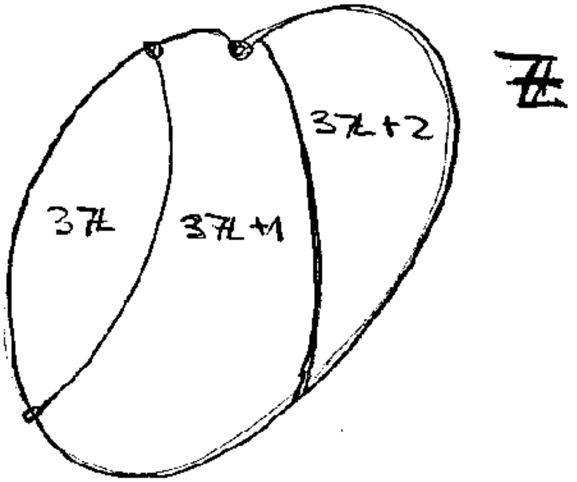
Beachte

$$\mathbb{Z} = 3\cdot\mathbb{Z} \;\dot\cup\; 3\cdot\mathbb{Z}+1 \;\dot\cup\; 3\cdot\mathbb{Z}+2$$

Mengen disjunkt,

d.h. ohne gemeinsame Elemente.

Partition = disjunkte Zerlegung.



Partition entspricht Äquivalenz-relation.

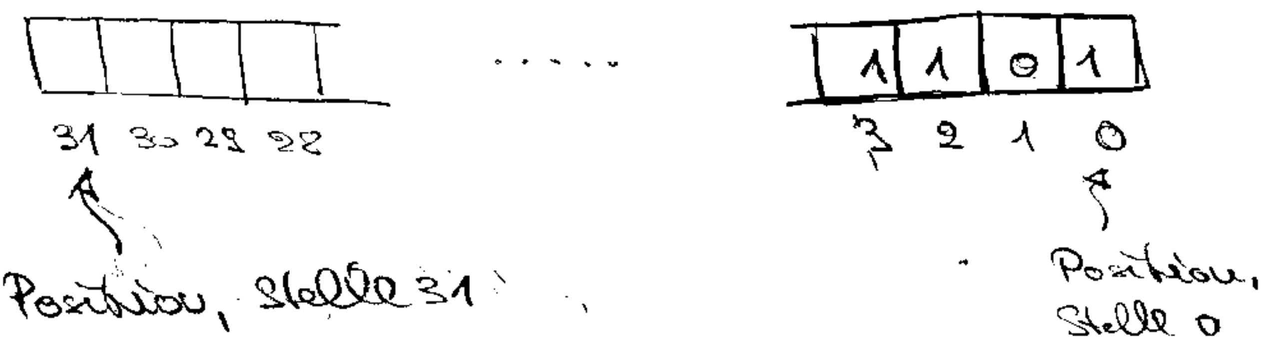
Ausprobieren in Java: $-7 \% 3$, $-7 / -3$

und auch einmal $7 \% -3$. $7 = (-3)\cdot(-2) + 1$

$-6/-3 \qquad -6 = -3\cdot 2 \qquad -7 \% -3 \qquad -7 = -3\cdot 3+2$

Wie werden Zahlen im Rechner dargestellt, d.h. in welcher Form werden sie gespeichert?

Ein Speicherwort des Hauptspeichers



Pro Position wird 1 Bit gespeichert

8 Bits = 1 Byte.

Speicherworte üblicherweise

32 Bits = 4 Bytes

oder neuerdings auch

64 Bits = 8 Bytes.

Zahlen (eigentlich alles) muß letztlich über den Bits 0, 1, also als Folge von Bits dargestellt werden.

Die Darstellung von Zahlen basiert auf dem Dualsystem. Zunächst einige Beispiele dazu:

0 dezimal ist 0 dual.

1 dezimal ist 1 dual.

2 dezimal ist 10 dual $(10)_2$

3 dezimal ist $(11)_2$

10 dezimal ist $(1010)_2$

„Ist im Dualsystem zu sehen"

Das Zehner- oder Dezimalsystem beruht auf folgendem Prinzip:

$$1 = 1 \cdot 10^0.$$

Beachte $10^0 = 1$, sinnvoll da $10^{a-a} = \frac{10^a}{10^a} = 1$.

$$2 = 2 \cdot 10^0, \dots, 9 = 9 \cdot 10^0.$$

$$10 = 1 \cdot 10^1 + 0 \cdot 10^0$$

$$12 = 1 \cdot 10^1 + 2 \cdot 10^0$$

$$100 = 1 \cdot 10^2$$

$$0 = 0 \cdot 10^0.$$

Für $a \in \mathbb{N}$ gilt: Es gibt ein $n \geq 1$ (Anzahl Stellen, # Stellen) und Ziffern $z_0, z_1, \dots, z_{n-1}$ mit $0 \leq z_i \lneq 10$

[illegible] $\neq 0$.

so daß

$$a = z_{m-1} \cdot 10^{m-1} + z_{m-2} \cdot 10^{m-2} + \dots + z_1 \cdot 10^1 + z_0 \cdot 10^0 .$$

Man schreibt dann auch

$$a = \sum_{i=0}^{m-1} z_i \cdot 10^i .$$

$m-1 \geq 0$, da $m \geq 1$ vorausgesetzt ist

(Entspricht der Zahl $(z_{m-1} z_{m-2} \dots z_0)_{10}$.)

„Dezimalsystem"

Wieso ist eigentlich jedes $a \in \mathbb{N}$ wie oben darstellbar? Das läßt sich aus unserer Division mit Rest ableiten: Ist $a \in \mathbb{N}$, dann haben wir Werte z_0 mit $0 \leq z_0 \lneq 10$ und q_1 so daß

$q_0 =$

$$a = q_1 \cdot 10 + z_0$$

Div$(a, 10^0)$ Div$(a, 10)$ Mod$(a, 10)$

Weiter $\leftarrow \mathrm{Div}(a,10^1)$

$a = \mathrm{Div}(a,1)$, $1 = 10^0$

$q_1 = q_2 \cdot 10 + z_1$ $\leftarrow \mathrm{Mod}(\mathrm{Div}(a,10),10)$

und

$\mathrm{Div}(\mathrm{Div}(a,10),10) = \mathrm{Div}(a,100)$. Denn

$q_2 = q_3 \cdot 10 + z_2$

$a = c \cdot 100 + r$

$\vdots$

$a = c_1 \cdot 10 + z_0$

$q_1 = c_2 \cdot 10 + z_1$ < 100

$q_{m-1} = 0 \cdot 10 + z_{m-1}$.

q_m

$\Rightarrow a = c_2 100 + z_1 10 + z_0$

Einsetzen ergibt nun:

$\mathrm{Div}(\mathrm{Div}(a,10),10) = \mathrm{Div}(a,100)$

$$a = q_0 \cdot 10 + z_0$$

$$= (q_1 \cdot 10 + z_1) \cdot 10 + z_0$$

$$= ((q_2 \cdot 10 + z_2) 10 + z_1) 10 + z_0$$

$$= (((q_3 \cdot 10 + z_3) \cdot 10 + z_2) \cdot 10 + z_1) \cdot 10 + z_0$$

$$\vdots$$

$$= z_{m-1} \cdot 10^{m-1} + z_{m-2} \cdot 10^{m-2} + \dots + z_2 \cdot 10^2 + z_1 \cdot 10 + z_0$$

Also: Existenz der fraglichen Darstellung.

Wieso ist die Darstellung im Dezimalsystem eindeutig? Angenommen es wäre für ein $a \in \mathbb{N}^+$

$$a = \sum_{i=0}^{m-1} z_i \cdot 10^i \quad \text{und} \quad a = \sum_{j=0}^{m-1} y_j \cdot 10^j$$

und $0 \leq z_i, y_i \leq 10$. Dann folgt, daß

$$z_0 = y_0$$

ist, da $z_0 = \mathrm{Mod}(a,10)$ und $y_0 = \mathrm{Mod}(a,10)$ und Division mit Rest eindeutig.

Das ist $\mathrm{Div}(a,10)$

Dann ist

$$0 \leq z_i \leq 9$$

$$= \sum_{i=1}^{m-1} z_i \cdot 10^i = \sum_{j=1}^{m-1} y_j \cdot 10^j,$$

darum $q_1 = \sum_{i=1}^{m-1} z_i \cdot 10^{i-1} = \sum_{j=1}^{m-1} y_j \cdot 10^{j-1}$

Dann gilt für z_1 und y_1, daß

$$\mathrm{Mod}(q_1, 10) = z_1 \text{ und } \mathrm{Mod}(q_1', 10) = y_1$$

Also $z_1 = y_1$, dann $z_2 = y_2, \ldots$. Ist etwa $m \gneq n$,

dann haben wir $y_{m-1} = \ldots = y_m = 0$. $\left\{\begin{array}{c}\text{Dann} \\ m-1 \geq n\end{array}\right\}$

Also eigentlich: Eindeutigkeit abgesehen von Nullen vorne.

Statt mit 10 kann man ganz analog mit jedem $b \in \mathbb{N}^+$, $b \geq 2$ verfahren.

Dualsystem $b = 2$. ($b = 1$ geht nicht so, da $1^{z_i} = 1$, und $0 \leq z_i \lneq 1$ $z_i = 0$ erfordert.)

Also jedes $a \in \mathbb{N}$ ist abgesehen von Nullen vorne eindeutig darstellbar als

$$a = \sum_{i=0}^{n-1} z_i \cdot 2^i, \quad n \geq 1 \text{ und } 0 \leq z_i < 2$$

also Ziffern aus 0, 1. Dann auch die Schreibweise $a = (z_{n-1} \dots z_0)_2$.

$$(4)_{10} = 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = (100)_2$$

$z_0 = \mathrm{Mod}(4,2) = 0$,
$q_1 = \mathrm{Div}(4,2) = 2$

$z_1 = \mathrm{Mod}(2,2) = 0$,
$q_2 = \mathrm{Div}(2,2) = 1$

$z_2 = \mathrm{Mod}(1,2) = 1$,
$q_3 = \mathrm{Div}(1,2) = 0$ ← Deshalb Schluß
$q_4 = 0$.

Stelle 3 von hinten, beginnend 4 = 2mal Dividiert den Wert
$\mathrm{Mod}(\mathrm{Div}(\mathrm{Div}(\mathrm{Div}(4,2),2),2),2)$

Was ist die größte Zahl, die man auf n Stellen darstellen kann?

$$(\underbrace{1 \cdots 1}_{n-1})_2 = 1\cdot 2^{n-1} + 2^{n-2} + \dots + 2^0 = 2^n - 1 .$$

Denn es ist:

$$2^0 + 2^1 = 3 = 2^2 - 1$$

$$2^0 + 2^1 + 2^2 = 7 = 2^3 - 1$$

$$2^0 + 2^1 + 2^2 + 2^3 = 16 - 1 .$$

$$\vdots$$

$$\underbrace{2^0 + 2^1 + 2^2 + \dots + 2^{n-2}}_{= 2^{n-1} - 1} + 2^{n-1} = 2^n - 1 .$$

Das zeigt ein Induktionsbeweis.

Es wird gezeigt für alle $n \geq 1$ ist

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1 .$$

Induktionsanfang: $n = 1$

Es ist

$$2^0 = 1 = 2^1 - 1.$$

Induktionsschluß: Angenommen

es gilt die Behauptung für $n-1$.

Also

$$\sum_{i=0}^{n-2} 2^i = 2^{n-1} - 1.$$

Dann gilt

$$\sum_{i=0}^{n-1} 2^i = 2^{n-1} - 1 + 2^{n-1} = 2^n - 1$$

Die kleinste Zahl auf n Stellen, wenn eine 1 ist 2^{n-1}!

Also

$$2^n - 1 \geq 1\,\overbrace{z_{n-2} \cdots z_0}^{n-1 \text{ Nullen}} \geq 2^{n-1}$$

$$2 \cdot 2^{n-1} - 1$$

Ausgerechnet

$2^{n-1} + 2^{n-2} + 2^{n-3} + \dots + 1 + 1$

$= 2^n$

Die Null!

Es gilt noch allgemeiner die Formel für die geometrische Reihe

$$\sum_{i=0}^{n-1} a^i = \underbrace{a^0}_{=1} + a^1 + a^2 + a^3 + \dots + a^{n-1}$$

$$= \frac{a^n - 1}{a - 1} \quad \text{für alle } a \neq 1.$$

Der Beweis ist eine Übungsaufgabe.

Analog im Dezimalsystem: Größte Zahl mit n Stellen

$$(\underset{n-1}{9}\,9 \dots \underset{0}{9})_{10} = 10^{n-1} \cdot 9 + 10^{n-2} \cdot 9 + \dots + 9 \cdot 10 + 9$$

$$= 9 \cdot \sum_{i=0}^{n-1} 10^i$$

$$= 9 \cdot \frac{10^n - 1}{10 - 1} = 10^n - 1.$$

Noch einige Beispiele von Dualzahlen: (2.21)

$(8)_{10} = (1.000)_2$

$(16)_{10} = (1.0000)_2$

$(32)_{10} = (\underbrace{100000}_{2^5})_2$

$(64)_{10} = (1000000)_2$

$(128)_{10} = (10000000)_2$

Eine Stelle mehr führt zur Multiplikation mit 2. Exponentielles Wachstum. kombinatorische Explosion!

$\underbrace{(256)_{10}}_{2^8} = (1\underbrace{00000000}_{8 \text{ Nullen}})_2$

$\underbrace{(512)_{10}}_{2^9} = (1\underbrace{000000000}_{9 \text{ Nullen}})_2$

$\underbrace{(1024)_{10}}_{2^{10}} = (1\underbrace{0000000000}_{10 \text{ Nullen}})_2$

$1 \cdot 2^{10} + 0 \cdot 2^9 + 0 \cdot 2^8 + \ldots + 0 \cdot 2^1 + 0 \cdot 2^0.$

2.22

$$(1000)_{10} \cdot (1000)_{10} = (1.000.000)_{10}$$

also ist

$$2^{10} \cdot 2^{10} \approx (1.000.000)_{10}$$

Andererseits

$$2^{10} \cdot 2^{10} = 2^{10+10} = 2^{20} = (10\ldots\ldots0)_2$$

wobei wir 20 Stellen haben.

Wieviele Stellen braucht man zur binären Darstellung von $a \in \mathbb{N}$?

0	$(0)_2$	1 Stelle
1	$(1)_2$	1 Stelle,
2	$(10)_2$	2 Stellen
4	$(100)_2$	3 Stellen

$\vdots$

$\vdots$

$2^{10} = 1024 \quad (1\underbrace{0000000000}_{10 \text{ Nullen}})_2$ 11 Stellen

Die 0 ist nicht von dieser Form.

Zur Darstellung von Zweipotenzen

$a = 2^m$ brauchen wir genau $m+1$ Stellen. (gilt auch für $m = 0$.)

Vorne die Eins

m Nullen

Dann ist

$$m = \log_2 a$$

$2^m : 2 = 2^{m-1}$

$2^{m-1} : 2 = 2^{m-2}$

$m+1$ = # Male, die man durch 2 dividieren kann, bis ≤ 2.

Wie bei Nicht-Zweipotenzen?

2	3	4	5 … 7		8
$(10)_2$	$(11)_2$	$(100)_2$	3 Stellen		1000

Es ist $2 \lneq \log_2 5, \log_2 6, \log_2 7 \lneq 3$

etwa $\log_2 5 = 2{,}31\ldots$ oder ähnlich.

Stellen von a

= das kleinste i, so daß

$$\mathrm{Div}(a, 2^i) = 0$$

Dann ist $\lfloor 2{,}31 \rfloor = 2{,}0$, $\lfloor 2{,}0 \rfloor = 2$, $\lfloor 2{,}9 \rfloor = 2$

(Gauß Klammer, floor). Also ist jedes

$a \in \{4, \ldots, 7\}$ mit

$$\lfloor \log_2 a \rfloor + 1$$

Stellen darstellbar. Gilt immer, sogar für 0, wenn $\log_2 0 := 0$ definiert wird.

Beachte $\lceil \log_2 a \rceil$ stimmt bei $a=4$ nicht.

Stelle bis Division durch 2 ≤ 1 ist. Dann ebenso da Div $= 0$

Es ist für $a \geq 1$

$$\lfloor \log_2 a \rfloor + 1 = \lceil \log_2 (a+1) \rceil$$

$\lceil 2{,}7 \rceil = 3$, $\lceil 2{,}1 \rceil = 3$ (ceiling, Decke)

Für $a = 2^m$ Zweierpotenz haben wir $m+1$

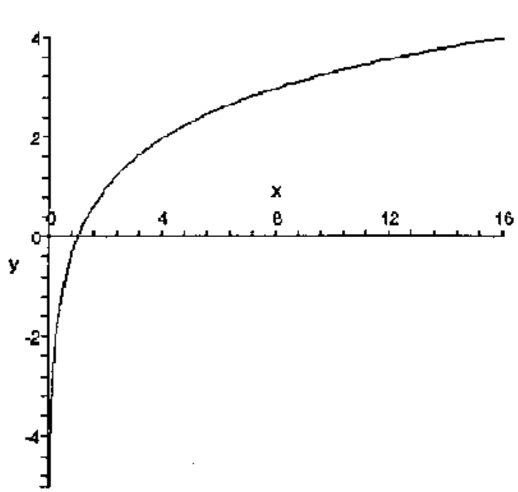
für $a = 2^m + b$ $1 \leq b \leq 2^m$ ist

$2^m + 2 \leq a+1 \leq 2^{m+1}$ also gilt es auch.

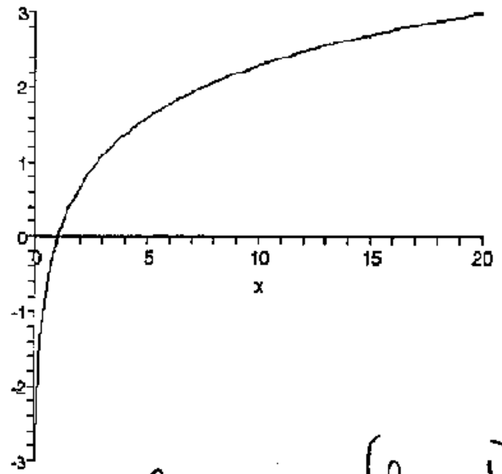
Nicht für $a = 0$ jedenfalls, $\lceil \log_2 1 \rceil = 0$.

Logarithmusfunktionen

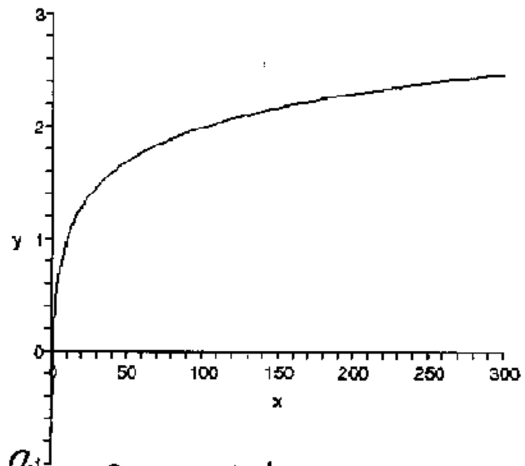
Dyadischer Logarithmus



Natürlicher Logarithmus



Dekadischer Logarithmus



$\log_c a = (\log_c b) \log_b a \quad c, b, a > 0$

$\log_b a = r,s \Leftrightarrow a = \underbrace{b \cdot b \cdot b \cdots b}_{r} \cdot \underbrace{b^s}_{<b}$

Logarithmengesetze:

$\log_a a = 1$ $\qquad$ $\log(ab) = \log a + \log b$ $\qquad$ $\log a^n = n \cdot \log a$

$\log 1 = 0$ $\qquad$ $\log \frac{a}{b} = \log a - \log b \quad (a>0, b>0)$ $\qquad$ $\log \sqrt[n]{a} = \frac{1}{n} \cdot \log a$

Grundrechenarten mit verschieden Basen kurz Beispielen. Dezimal → dual mit Resten. Dual → Dezimal folgendermaßen. (2.26)

$(100000)_2 = (32)_{10}$ wird

systematisch ermittelt durch:

$(100000)_2 : (1010)_2 = (11)_2$ Rest $(10)_2$

$$\begin{array}{r} (100000)_2 \\ -(1010)_2 \\ \hline (01100)_2 \\ -(1010)_2 \\ \hline (0010)_2 \end{array}$$

Das ist Div$((10000)_2, (1010)_2)$

$(11)_2 : (1010)_2 = 0$ Rest $(11)_2$, deshalb Abbruch.

Also ist

$(100000)_2$

$= (11)_2 \cdot (1010)_2 + (10)_2$

$= 3 \cdot 10 + 2$

$= (32)_{10}$.

2.27

$(10000000)_2 : (1010)_2 = (1100)_2$ Rest $(1000)_2$

$$\begin{array}{r}
10000000 \\
-\,1010 \\
\hline
01100 \\
1010 \\
\hline
1000
\end{array}$$

Nun ist $(1100)_2 > (1010)_2$ also

weiterteilen

$(1100)_2 : (1010)_2 = 1$ Rest $(10)_2$

$$\begin{array}{r}
1100 \\
1010 \\
\hline
10
\end{array}$$

Es ist

$(1)_2 : (1010)_2 = 0$ Rest $(1)_2$

$(10)_2$

$(1)_2$ $(1000)_2$

Also

$(10000000)_2 = (128)_{10}$

(2.28)

Multiplikation mit 2 im Dualsystem : Eine Null anhängen.
(Verschieben (shift) nach links.)

$\mathrm{Div}\left((10010)_2, (10)_2\right) = (1001)_2$

$\mathrm{Mod}\left((10011)_2, (10)_2\right) = (1)_2.$

Bei Speicherworten der Länge etwa 4 (4 Bits) können wir so die Zahlen $(0)_{10} = (0000)_2$, $(1)_{10} = (0001)_2, \ldots, (1111)_2 = (15)_{10}$ darstellen. Bei Länge 32 etwa $0, \ldots, \underbrace{(111\ldots1)_2}_{32} = 2^{32}-1 \approx 4.000.000.000$

Größte Zahl

Also $16 = 2^4$
größte Zahl $= 16-1=15$
4 Stellen
alle
freie
Stellen
($2^3 = 8$, von $(1000)_2$ bis $(1111)_2$ von 0 bis 7 sind 8 Werte.

Was ist, wenn die Addition aus diesem Bereich hinausführt?
Wir lassen den Übertrag, der rausfällt weg! Was heißt das dann?

$$(1111)_2 + (0001)_2 = (10000)_2 = (16)_{10},$$

wird dann aber zu Null.

$$(1111)_2 + (0010)_2 = (10001)_2 = (17)_{10},$$

wird aber dann zu 1.

$$(1111)_2 + (1111)_2 = (11110) = (30)_{10}$$

wird zu $(1110)_2 = (14)_{10}$.

Es ist

$$\mathrm{Mod}\left((z_{n-1} \dots z_0)_2, \underbrace{(10000)_2}_{=(16)_{10}}\right) = (z_3 z_2 z_1 z_0)_2$$

und allgemein:

$$\mathrm{Mod}\left((z_{n-1} \dots z_0)_2, 1\underbrace{0 \dots 0}_{i \text{ Nullen}}\right) = (z_{i-1} z_{i-2} \dots z_0)_2$$

Abschneiden der ersten Stelle bei Vieren bedeutet uns rechnen auf den Resten mod 16.

Ebenso bei etwa 32 Positionen: mod 2^{32}. Ist $0 \leq a, b, a+b \lneqq 2^{32}$, dann ist

$$\mathrm{Mod}(a+b, 2^{32}) = a+b$$

Ist also $0 \leq a, b \lneqq 2^{32}$ und $a+b \geq 2^{32}$, dann ist

$$\mathrm{Mod}(a+b, 2^{32}) = \underbrace{a+b-2^{32}}_{\lneqq 2^{32}},$$

da $a+b \lneqq 2^{33}$ ist.

Eine erste Möglichkeit auch negative Zahlen zu bekommen, ist die Einführung eines Vorzeichenbits nach folgendem Muster:

0111	7
⋮	
0001	1
0000	0
1000	-0
1001	-1
⋮	
1111	-7

Das erfordert aber auch einen eigenen Algorithmus ~~zur~~ Subtraktion. Es geht auch einfacher.

Das Ziel ist es, die Subtraktion auf die Addition mit Abschneiden der eventuell überzähligen Stelle zurückzuführen. Am Beispiel von $n = 2$, also Zahlen $(x_1x_0)_2$ sieht es so aus:

00	01	10	11
0	1	2	3

Es ist

$$11 + 10 = 100$$

das ergibt dann 00. Interpretieren wir doch einfach 11 als -1.

Dann weiter im

$$11 + 11 = 110$$

ergibt 10. Wie ist 10 zu interpretieren? Als -2.

Die neue Interpretation ist die

2'er Komplement

Darstellung:

00	01	10	11
0	1	-2	-1
	2^1-1	-2^1	-2^1+1

Allgemein auf n Positionen davor:

$$\overbrace{0 \ldots 0}^{n \text{ Nullen}} \quad 0 \ldots 1 \quad \ldots \quad 01 - 1$$

$$0 \quad 1 \quad \ldots \quad 2^{n-1}-1$$

$$10 \ldots 0 \quad 10 \ldots 01 \quad \ldots \quad \overbrace{11 \ldots 1}^{n \text{ Einsen}}$$

$$-2^{n-1} \quad -2^{n-1}+1 \quad \ldots \quad -1$$

Insgesamt haben wir hier 2^n Werte.

Die offizielle Definition der 2'er Kpl. Darstellung auf n Bits ist jetzt:

- Ist $0 \leq a \leq 2^{n-1} - 1$, dann ist das

  2'er Kpl. von a = Binärdstg. von a auf n Bits. (Vorne eine 0)

- Ist $-2^{n-1} \leq a \leq -1$, etwa $a = -a'$, dann

  2'er Kpl. von a = Binärdstg. von $2^n - a' = 2^n + a$. auf n Bits (Vorne eine 1).

Man kann auch alternativ und prägnanter definieren:

Man beachte: $0 \leq a \leq 2^4$, dann $Mod(a, 2^4) = a$
$-2^4 \leq a \leq 0$, dann $Mod(a, 2^4) = 2^4 + a$, $a \leq 0$.

2'er Kpl. von a

= Binärdarstellung auf n Bits von $Mod(a, 2^n)$.

Es ist bekanntlich $Mod(-1, 2^4) = 2^4 - 1$, da

$-2^4 + \underset{\text{Rest}}{(2^4 - 1)} = -1$, $Mod(-2^{4-1}, 2^4) = 2^{4-1}$.

Wir haben also jetzt 2 Möglichkeiten, eine Bitfolge als Zahl zu interpretieren: Als Binärzahl oder als 2'er Kpl. Darstellung. Aus der Bitfolge $z_{n-1} \dots z_0$ ergibt sich die Binärzahl bekanntlich als

$$(z_{n-1} \dots z_0)_2 = \sum_{i=0}^{n-1} 2^i \cdot z_i .$$

Dagegen haben wir beim 2'er Kpl:

$$(z_{n-1} \dots z_0)_{2\text{Kpl}} = \begin{cases} (z_{n-1} \dots z_0)_2 & \text{falls } z_{n-1} = 0 \\ \underbrace{(z_{n-1} \dots z_0)_2 - 2^n}_{\lneq 0.} & \text{falls } z_{n-1} = 1 . \end{cases}$$

Man prüft leicht nach, daß der Prozeß

$a \rightsquigarrow$ 2'er Kpl Dstg. von a, $z_{n-1} \dots z_0$

$\rightsquigarrow (z_{n-1} \dots z_0)_2$ falls $z_{n-1} = 0$ bzw.

$(z_{n-1} \dots z_0)_2 - 2^n$ falls $z_{n-1} = 1$

in jedem Falle wieder a ergibt.

Quintessenz ist nun der folgende Satz, der besagt, daß die normale binäre Addition hinreichend ist, um auf den 2'er Kpl'ten zu rechnen.

Satz

Sei $-2^{n-1} \le a, b \le 2^{n-1} - 1$ und

$$(x_{n-1} \dots x_0)_{2Kpl} = a, \quad (y_{n-1} \dots y_0)_{2Kpl} = b.$$

Sei außerdem

$$(z_n z_{n-1} \dots z_0)_2 = (x_{n-1} \dots x_0)_2 + (y_{n-1} \dots y_0)_2$$

(also eine normale binäre Addition).

(a) Für alle a, b wie oben gilt:

$$\underbrace{\mathrm{Mod}\left((z_n z_{n-1} \dots z_0)_2, 2^n\right)}_{= (z_{n-1} \dots z_0)_2} = \mathrm{Mod}(\underbrace{a+b}_{-2^n \le a+b \le 2^n - 2}, 2^n)$$

(b) [illegible]

(b) Ist nun auch noch $-2^{n-1} \le a+b \le 2^{n-1}-1$, so ist $z_{n-1} \dots z_0$ das 2'er Kpl. von $a+b$, das heißt

$$(z_{n-1} - z_0)_{2\,Kpl} = a+b.$$

(Das heißt, addieren wir die beiden 2'er Kpl't'e einfach binär, bilden dann den Rest modulo 2^n, so haben wir das 2'er Komplement von $a+b$. Das folgende Diagramm „kommutiert" unter den gegebenen Voraussetzungen:

$$\begin{tikzcd}
(a,b) \arrow[r, "\text{Normale Addition}"] \arrow[d] & a+b \in \mathbb{Z} \arrow[d] \\
(2\text{'er Kpl } a,\ 2\text{'er Kpl } b) \arrow[r, "\text{Normale (binäre) Addition modulo } 2^n"'] & 2\text{'er Kpl } a+b
\end{tikzcd}$$

(9.38)

Beweis

(a) Die Reste modulo 2^n bleiben gleich, sofern wir Vielfache (positive oder negative) von 2^n zu einem Wert hinzuaddieren.

(b) Mit (a) gilt, daß

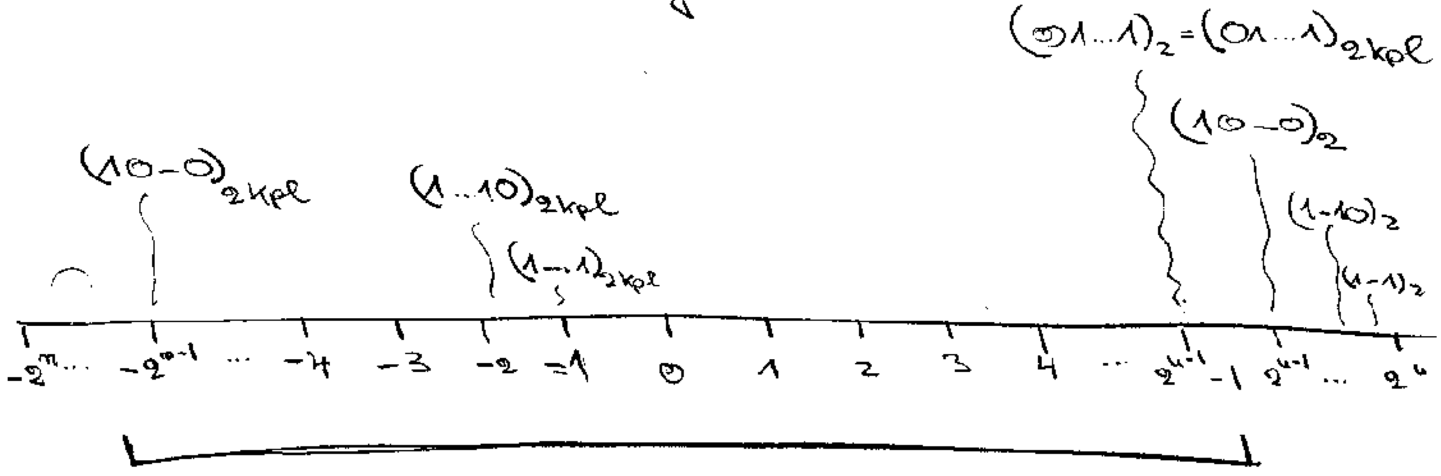
$$\underbrace{(z_{n-1} \dots z_0)_2}_{\lneq 2^n} = \mathrm{Mod}(a+b, 2^n)$$

ist. Da $a+b$ in dem fraglichen Bereich ist, $-2^{n-1} \le a+b \le 2^{n-1}-1$, ist nach der alternativen Definition oben $z_{n-1} \dots z_0$ die 2'er Kpl. Darstellung von $a+b$, also

$$(z_{n-1} \dots z_0)_{2Kpl} = a+b. \qquad \square$$

Hausaufgabe: Aufgaben 9.22, ...

Noch einmal das 2'er Komplement an der Zahlengeraden:



Bereich des 2'er Kpl'ts.

Es ist $-2^{n-1} = -2^n + 2^{n-1}$

also $\mathrm{Mod}(-2^{n-1}, 2^n) = 2^{n-1} = (1..0)_2$

Ebenso $-2^{n-1}+1 = -2^n + (2^{n-1}+1)$

also $\mathrm{Mod}(-2^{n-1}+1, 2^n) =$

$= 2^{n-1}+1 = (10..01)_2$

⋮

Schließlich $-1 = -2^n + (2^n - 1)$

also $\mathrm{Mod}(-1, 2^n) = (1\ldots1)_2$.

Folgenden Trick zur Ermittlung des 2'er Kpl-Darstg auf n Bits sollte man kennen:

Für $0 \le a \le 2^{n-1}-1$ gibt es mittels der Binärdarstellung von a.

Für $-2^{n-1} \le a \le -1$ und $a = -a'$ ist es die Binärdstg. von $2^n + a = 2^n - a'$.

Nun ist

$$2^n - a' \quad 2^n - a' = 2^n - 1 - a' + 1.$$

Ist $(x_{n-1} \dots x_0)_2 = a'$, so ist die Binärdarstellung von $2^n - 1 - a'$ einfach

$$(\bar{x}_{n-1}\, \bar{x}_{n-2} \dots \bar{x}_0)_2 = 2^n - 1 - a'$$

mit $\bar{x}_i = 1 - x_i$ ($\bar{x}_i = 0$ wenn $x_i = 1$, $\bar{x}_i = 1$ wenn $x_i = 0$).

$\overline{x}_{n-1} \dots \overline{x}_0$ ist das 1'er Komplement von $x_{n-1} \dots x_0$, zu dem dann nur noch 1 addiert werden muß, um zum 2'er Komplement zu kommen.

Betrachtung der Java Programme ZweiKplTest.java und ZweibyteTest.java.

Wir haben in Java 4 verschiedene Typen für ganze Zahlen. Arbeiten mit der 2'er Kpl. Darstg.

byte   1 Byte = 8 Bit   von -128 bis 127
($2^8 = 256$)

short   2 Byte = 16 Bit   von -32768 bis 32767
($2^{16} = 65536$)

int 32 Bit von -2^{31} bis $2^{31}-1$
($2^{31} \approx 2$ Milliarden)

long 64 Bits von -2^{63} bis $2^{63}-1$.

Literalkonstanten ausprobieren

LitKonst.java

Der Compiler macht folgendes:

1. Parameter rechts long. Ergebnis von arithmetischer Operation ist long. Gibt Compilerfehler bei links int.

2. Kein Parameter long, dann int. Egal ob Typen byte oder so.

Die linke Seite muß dann int sein. Oder explizite Verkleinerung des Bereiches rechts.

Wie in (byte)(a+b), selbst nötig, wenn a; b byte sind, da dann a+b als int gesehen wird.

Implizite Typanpassung nur dann, wenn kleinerer Bereich in den größeren: Rechts int links long geht automatisch.

Das Programm PRIM in PRIM.java.

Korrektheitsbeweis: Wird die Schleife mit dem return statement verlassen $\Rightarrow$ c_{start} keine Primzahl. Wir überlegen uns: Wird die Schleife nicht mit dem return statement verlassen $\Rightarrow$ c_{start} Primzahl.

Sei wieder

d_l = Wert von d nach l'tem Lauf,

$l \geq 1$, sofern er stattfindet. Außerdem

$d_0 = 2$.

(2.44)

Erste Invariante: → (2.44 a)

$$d_l = l + 2$$

(gilt nach dem l'ten Lauf (sofern er stattfindet)).

Sieht man durch Inspektion des Rumpfes.

Zweite Invariante:

$2, 3, 4, \ldots, d_l - 1$ teilen q nicht,

gilt nach dem l'ten Lauf (sofern er stattfindet). Leichte Inspektion.

Die Schleife läuft bis $d_l \cdot d_l \geq q$ ist. Da $d_l = l+2$ hält sie also in jedem Falle an. Ist l so, daß $d_l \cdot d_l \geq q$ ist, gilt

$2, 3, \ldots, d_l - 1$ teilen q nicht.

Da $d_l \geq \sqrt{q}$ ist deshalb q Primzahl, denn wir haben keinen Teiler unter $2, \ldots, \sqrt{q}$

falls c Quadratzahl und keinen unter $2, \ldots, \lfloor\sqrt{c}\rfloor$ falls c keine Quadratzahl. Die Anzahl der Durchläufe ist $\sqrt{c} - 2$ falls c Quadratzahl, $\lceil\sqrt{c}\rceil - 2$ falls c keine Quadratzahl.

Etwa: $c = 5$, dann $d_1 = 3$ Schluß und $\lceil\sqrt{5}\rceil = 3$. $c = 7$, dann $d_1 = 3$ Schluß.
$c = 11$, dann $d_1 = 3$, $d_2 = 4$ Schluß.

2.45

$2^{15} = 1024 \cdot 32$

Table 1

USEFUL PRIME NUMBERS

N	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
2^{15}	19	49	51	55	61	75	81	115	121	135
2^{16}	15	17	39	57	87	89	99	113	117	123
2^{17}	1	9	13	31	49	61	63	85	91	99
2^{18}	5	11	17	23	33	35	41	65	75	93
2^{19}	1	19	27	31	45	57	67	69	85	87
2^{20}	3	5	17	27	59	69	129	143	153	185
2^{21}	9	19	21	55	61	69	105	111	121	129
2^{22}	3	17	27	33	57	87	105	113	117	123
2^{23}	15	21	27	37	61	69	135	147	157	159
2^{24}	3	17	33	63	75	77	89	95	117	167
2^{25}	39	49	61	85	91	115	141	159	165	183
2^{26}	5	27	45	87	101	107	111	117	125	135
2^{27}	39	79	111	115	135	187	199	219	231	235
2^{28}	57	89	95	119	125	143	165	183	213	273
2^{29}	3	33	43	63	73	75	93	99	121	133
2^{30}	35	41	83	101	105	107	135	153	161	173
2^{31}	1	19	61	69	85	99	105	151	159	171
2^{32}	5	17	65	99	107	135	153	185	209	267
2^{33}	9	25	49	79	105	285	301	303	321	355
2^{34}	41	77	113	131	143	165	185	207	227	281
2^{35}	31	49	61	69	79	121	141	247	309	325
2^{36}	5	17	23	65	117	137	159	173	189	233
2^{37}	25	31	45	69	123	141	199	201	351	375
2^{38}	45	87	107	131	153	185	191	227	231	257
2^{39}	7	19	67	91	135	165	219	231	241	301
2^{40}	87	167	195	203	213	285	293	299	389	437
2^{41}	21	31	55	63	73	75	91	111	133	139
2^{42}	11	17	33	53	65	143	161	165	215	227
2^{43}	57	67	117	175	255	267	291	309	319	369
2^{44}	17	117	119	129	143	149	287	327	359	377
2^{45}	55	69	81	93	121	133	139	159	193	229
2^{46}	21	57	63	77	167	197	237	287	305	311
2^{47}	115	127	147	279	297	339	435	541	619	649
2^{48}	59	65	89	93	147	165	189	233	243	257
2^{59}	55	99	225	427	517	607	649	687	861	871
2^{60}	93	107	173	179	257	279	369	395	399	453
2^{63}	25	165	259	301	375	387	391	409	457	471
2^{64}	59	83	95	179	189	257	279	323	353	363
10^{6}	17	21	39	41	47	69	83	93	117	137
10^{7}	9	27	29	57	63	69	71	93	99	111
10^{8}	11	29	41	59	69	153	161	173	179	213
10^{9}	63	71	107	117	203	239	243	249	261	267
10^{10}	33	57	71	119	149	167	183	213	219	231
10^{11}	23	53	57	93	129	149	167	171	179	231
10^{12}	11	39	41	63	101	123	137	143	153	233
10^{16}	63	83	113	149	183	191	329	357	359	369

2^{15}

32768
− 19
32749

999983

10 000 000 000

9 999 999 967

The ten largest primes less than N are $N - a_1, \ldots, N - a_{10}$.

10.000.000.000.000.000
− 63
9 999 999 999 999 937

Zunächst einmal noch zwei grundsätzliche Dinge zu Primzahlen:
Es ist einer der frühesten Beweise der Mathematik (von Euklid, ca 300 v Christus), daß es unendlich viele Primzahlen gibt, also

$$|\underbrace{\mathbb{P}}_{=\{2,3,5,7,\dots\}}| = \infty$$

Das sieht man recht leicht, indem man zu jeder endlichen Menge von Primzahlen eine neue Primzahl angibt, die nicht zu dieser Menge gehört:

Die Konstruktion am Beispiel:

Menge $M = \{2, 3, 7\}$, dann

betrachte zunächst $b = 2 \cdot 3 \cdot 7 + 1 = 43$

Dann gilt zunächst einmal

$$2 \nmid b, \; 3 \nmid b, \; 7 \nmid b.$$

Also keine der Zahlen der Menge teilt b.

b braucht im allgemeinen keine

Primzahl zu sein, hat aber

einen kleinsten Teiler $\neq 1$.

Immer sogar $q_1 \cdots q_k + 1$ wird nicht von q_j geteilt!

Dieser muß Primzahl sein.

Dieser kleinste Teiler ist,

wie gesagt, nicht in unserer Menge

und wir haben eine weitere Primzahl.

(2.48)

Analog eben für beliebige endliche Menge von Primzahlen $p_1, \dots, p_m$.

Es ist nicht direkt $p_1 \cdots p_n + 1$ Primzahl, wenn $p_1, \dots, p_n$ die ersten Primzahlen sind:

$2 \cdot 3 + 1 = 7$, $2 \cdot 3 \cdot 5 + 1 = 31$, $2 \cdot 3 \cdot 5 \cdot 7 + 1 = 211$

$2 \cdot 3 \cdot 5 \cdot 7 \cdot 9 + 1 = 1891$ keine Primzahl.

Ebenso klappt es nicht bei

$2 \cdot 3 + 1 = 7$ $2 \cdot 3 \cdot 7 + 1 = 43$,

$2 \cdot 3 \cdot 7 \cdot 43 + 1 = 1807$, ist auch keine Primzahl mehr.

Merken wir aber: Der kleinste Teiler ≥ 2 einer Zahl muß eine Primzahl sein!

Primzahlen beziehen ihre Bedeutung (auch) aus dem Fundamentalsatz der Arithmetik:

Jede Zahl $a \in \mathbb{N}$, $a \gneq 1$ läßt sich schreiben als

$$a = p_1^{e_1} \cdots p_m^{e_m}$$

$p_1, \ldots, p_m$ sind paarweise verschiedene Primz.; $e_i \geq 1$.

Diese Darstellung ist im wesentlichen eindeutig. „Bis auf Reihenfolge"

Etwa $a = 7 \; (= 7^1)$. $a = 8 = 2^3$,

$15 = 3 \cdot 5 = 5 \cdot 3$, $30 = 3 \cdot 2 \cdot 5 = 2 \cdot 3 \cdot 5 \ldots$

$24 = 3 \cdot 2^3 = 2^3 \cdot 3$. (Brüdern Bundschuh, Zahlentheorie)

Wozu kann man eine solche Faktorisierung verwenden? Man kann zum Beispiel die ganzzahligen Teiler von n ablesen: Sei

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

dann gilt

$m \mid n$ gdw. $m = p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k}$,

wobei $f_i \leq e_i$ für alle i.

„$\Leftarrow$" gilt offensichtlich.

„$\Rightarrow$" Sei $m \mid n$, dann $n = q \cdot m$,

falls zum Beispiel $m = p_1^{f_1} m'$,

$f_1 > e_1$, dann

$$n = p_1^{f_1} m' \cdot q .$$

also hätten wir eine zweite

2.51

Primfaktorzerlegung im Widerspruch zur Eindeutigkeit. Ebenso, falls $m = p \cdot m'$ und $p \neq p_i$ für alle i.

Primzahlen $q_1 \le q_2 \le q_3 \le \dots \le q_k$ und $a = q_1 \cdot q_2 \cdot q_3 \cdots q_k$.

Also: Wie kann man jetzt die Primfaktorzerlegung von einer Zahl a ermitteln?

Ist

$$\begin{array}{ccccccccccc} & & 3 & & 5 & & 7 & & 8 & & \\ 2 = p_0 & < & p_1 & < & p_2 & < & p_3 & < & p_4 & < \dots < & p_s \end{array}$$

die Folge aller(!) Primzahlen $\le \sqrt{a}$, dann:

1. Teile a durch p_0 solange es geht
2. Durch p_1 ....

$\vdots$

Hier stellt sich folgendes Problem:

Wo bekommt man die Folge

$$p_0 < p_1 < \dots < p_s$$

her? Tatsächlich braucht

man sie gar nicht! Nehmen

wir stattdessen einfach

$$2 < 3 < 4 < 5 < 6 < 7 < \dots \leq a.$$

1. Teile a durch 2 solange es geht.
Die Vielfachen von 2 teilen danach
a sowieso nicht mehr.

2. Teile a durch 3 solange es geht.
Vielfache von 3 teilen nicht mehr.

⋮

a Einlesen

$d = 2$

while $d \leq a$ do

{ while ($a \% d == 0$) //Eliminiert
{ //Teiler d.

d hinschreiben

$a = \mathrm{div}(a, d)$

}

$d = d{+}{+}$ // Nächstes d verwenden.

}

Zur inneren Schleife: Seien

a_0, d_0 = die Werte vor der Schleife,

sei für $l \geq 1$

a_l = Wert nach l-tem Durchlauf.

sei jetzt $l \geq 1$ so daß einer mindestens l (Läufe) haben. Dann gilt folgende Schleifeninvariante:

Aufgabe ist $\underbrace{d, d, \ldots, d}_{l\text{-mal}}$

$a_l = \mathrm{Div}(a, d^l)$ und a_l ist ganzzahlig.

Anzahl Durchläufe = Anzahl Male, so daß d_0 das a_0 teilt.

Für $l = 1$ gilt die Invariante, gilt sie für l, dann für $l+1$.

Wegen der Bedingung $a \% d == 0$ gilt am Ende der inneren Schleife, daß d das a_l nicht mehr teilt (Schreibweise $d \nmid a_l$, denn das bedeutet gerade nicht $a \% d == 0$).

Damit ist für die innere Schleife klar. Bei Start mit a_0, d_0 endet sie so, daß

$$\underbrace{d_0, \ldots, d_0}_{\ell\text{-mal}}$$

hingeschrieben sind und

$$a_\ell = a_0/(d_0)^\ell \text{ und } d \nmid a_\ell .$$

Korrektheit des Gesamtprogramms.

a_0 = der Wert des eingelesenen a

$d_0 = 2$

und sei für $k \geq 1$, so daß $\geq k$ Läufe der äußeren Schleife stattfinden.

a_k = der Wert von a nach k Läufen

d_k = der Wert von d nach k Läufen.

Sei weiterhin $m \geq 0$ (hängt von a_0 und k ab, Schreibweise $m = m(a_0, k)$) die Anzahl der ausgegebenen Zahlen nach k Läufen und sei

$q_1, \ldots, q_m$

die Folge dieser Zahlen.

Schleifeninvariante ist:

$q_1, \dots, q_m$ sind Primzahlen,

$a_k = a_0 / c_1 \cdot \dots \cdot q_m$ ist ganzzahlig,

a_k hat keine (Prim-)Faktoren $\lneq d_k$;
$d_k = 2 + k$.

Ist das wirklich eine Invariante?

$k = 1$. Dann ist

$$q_1, \dots, q_m = 2, \dots, 2$$

$$a_1 = a_0 / c_1 \cdot \dots \cdot c_m \quad \text{(} q_1 \cdots q_0 = 1 \text{, das leere Produkt")}$$

und $2 \nmid a_1$.

Alles wegen der Eigenschaften der inneren Schleife! Außerdem ist $d_1 = 3 = 2 + k$.

Die Invariante gilt tatsächlich.

gelte sie noch k Durchläufen
und finde ein $k+1$'ter Lauf statt.

Also ist $d_k = 2+k$.

1. Fall d_k keine Primzahl.

Die innere Schleife wird nicht betreten, da a_k keine (Prim-)Faktoren $\leqq d_k$ hat und d_k keine Primzahl ist.

Sonst hätten d_k ja Primfaktoren $\leqq d_k$, die dann a_k teilen würden.

Es ist dann

$$d_{k+1} = d_k + 1 = 2 + k + 1$$

und Invariante gilt.

2. Fall d_k Primzahl

2a. Fall $d_k \nmid a_k$ Invariante gilt.

2.b. Fall $d_k \mid a_k$, dann wie oben bei $k=1$ und die Invariante gilt.

Die Anzahl der Durchläufe der äußeren Schleife ist endlich, wegen d++ und da a_k nicht größer wird.

Ist nun die äußere Schleife zu Ende, also $d_k \geq a_k$, dann gilt immer noch die Invariante. Da a_k keinen Primfaktor $\lneq d_k$ hat, muß (!) $a_k = 1$ und somit ist $c_1, \ldots, c_m$ dann tatsächlich die Primfaktorenzerlegung von a_0.

Einige Beispiele:

k	a_k	d_k	Ausgabe	
0	2	2	–	Äußere Schleife.
1	1	3	2	Schluß.

Ausgabe im ersten Lauf.

2.59

n	a_n	d_n	Ausgabe
0	3	2	–
1	~~3~~ 7	~~3~~ 1	–
2	1	4	3 ← Ausgabe im 2. Lauf.

Schluß

n	a_n	d_n	Ausgabe
0	8	2	
1	1	3	2, 2, 2

Schluß.

n	a_n	d_n	Ausgabe
0	10	2	
1	5	3	2
2	5	4	–
3	5	5	–
4	5	6	5

Wo geht nicht etwa bis 11 oder ähnliches.

(2.6.0)

Ist aber p eine Primzahl, dann

k	a_k	d_k	Ausgabe
0	p	2	–
1	p	3	–
2	p	4	–
	$\vdots$		
$p-2$	p	p	–
$p-1$	1	$p+1$	p

Schluß.

$d_0 = 2$

Anzahl Durchläufe der äußeren Schleife mit $k=0, \ldots, k = p-2+1$ wobei p der größte Primteiler von a_0 ist. Dann ist $d_{p-2+1} = p+1$! und $a_{p-2+1} = 1$. Beachte auch

2.61

noch einmal die Bedingung

unserer Invariante : a_k hat keinen (!)

(Prim-)Teiler $\lneq d_k$!

Kann man dieses Programm

noch verbessern?

1. Weniger d's generieren, nämlich

welche die sowieso keine Primteiler induzieren.

2. Sind wir bei einem k mit

$d_k \cdot d_k \gneq a_k$ angekommen, ist

a_k Primzahl und kann

als solches ausgegeben werden

(unsere Invariante gilt ja immer noch).

(2-62)

Also eine Verbesserung:

⋮

while $d \cdot d \leq a$

{

while

{

}

}

a als letzten Primteiler ausgeben.

Am Ende ist das a_k Primzahl, da $d_k \cdot d_k \gneq a$ und a keinen Primteiler $\lneq d_k$ enthält nach der Invariante.

Jetzt das folgende Problem:

Größter gemeinsamer Teiler.

Sind $g, h \in \mathbb{Z}$, dann ist

$$ggT(g,h) =$$

$$:= \max \underbrace{\{ b \in \mathbb{N}^+ \mid b \mid g \text{ und } b \mid h \}}_{M :=}.$$

Etwa $g = 10$, $h = -8$, dann

$M = \{1, 2\}$, $ggT(10,8) = 2$.

Dagegen $g = 12$, $h = -8$, dann

$M = \{1, 2, 4\}$, $ggT(12,-8) = 4$.

Es ist $ggT(0, h) = h$, sofern $h \neq 0$.

M = die Menge der Teiler von h

Ist $h \lneqq 0$, dann

$$ggT(0,h) = \underbrace{-h}_{\gneqq 0}$$

Wäre ist $ggT(0,0)$, dann $M = \mathbb{Z}$,
deshalb $ggT(0,0)$ nicht definiert.

Es ist für g,h

$$ggT(-g,h) = ggT(-g,-h)$$
$$= ggT(g,-h) = ggT(g,h).$$

Es kommt also nicht aufs Vorzeichen an.

Wie können wir den $ggT(g,h)$ ermitteln? Dazu zunächst einmal die Menge der

Teiler einer Zahl (bereits gezeigt):

Ist $g \geq 1$ und Primfaktorzerlegung

$$g = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_k^{e_k}$$

dann

$b \mid g \iff b = p_1^{f_1} p_2^{f_2} \ldots p_k^{f_k}$ und

für alle i ist $f_i \leq e_i$.

Habe nun auch $h \geq 1$ die Zerlegung

$$h = q_1^{a_1} \ldots q_l^{a_l} \quad \text{und } a_i \geq 0.$$

Seien $s_1, \ldots, s_m$, $m \leq l, k$ die in g und h vorkommenden Primteiler, dann ist

$$M = \{ s_1^{b_1} s_2^{b_2} \cdot \ldots \cdot s_m^{b_m} \mid b_i \leq \text{Exponent von } s_i \text{ in } g \text{ und in } h \}.$$

Damit ist $ggT(g,h)$ das größte Element in T und das ist

Max $M = s_1^{c_1} \cdots s_m^{c_m}$, wobei

$c_i = \min\{\text{Exp. von } s_i \text{ in } g, \text{Exp von } s_i \text{ in } h\}$.

Hier sieht man auch, daß alle gemeinsamen Teiler $ggT(g,h)$ teilen!

Etwa:

$ggT(30, 8) = 2^1$, denn $30 = 2 \cdot 5 \cdot 3$, $8 = 2^3$.

Hätten wir die Primfaktoren und ihre Exponenten zur Verfügung ließe sich $ggT(g,h)$ leicht ermitteln.

Zur Sammlung der Primfaktoren: Datenstrukturen, etwa Mengen, Folgen.

Wir machen es zunächst einmal ohne Datenstrukturen in ggTPrim.java

Tatsächlich gibt es bessere Verfahren, den $ggT(g,h)$ zu ermitteln (nach Euklid, ca 300 v. Chr.). Zunächst gilt für $g \gneqq 0$

$$ggT(g,g) = ggT(g,g) = g. \quad \left\{ \begin{array}{l} \text{auch} \\ ggT(0,g) = g \end{array} \right. .$$

Außerdem gilt für $g \neq h$, etwa $g \gneqq h \geq 1$, daß mit

$M_1 = \{ b \mid b \mid g \text{ und } b \mid h \}$

$M_2 = \{ b \mid b \mid (g-h) \text{ und } b \mid h \}$

Statt dem g in M_1

gilt

$$M_1 = M_2 .$$

Denn gilt $b \in M_1$, dann

$$g = b \cdot q_1, \quad \text{und} \quad h = b \cdot q_2, \quad q_1, q_2 \geq 1$$

dann

$$g - h = b \cdot q_1 - b \cdot q_2 = b \cdot (q_1 - q_2)$$

2.68

Also: $b \mid g-h$ und $b \mid h$.

Andererseits ist für $b \in M_2$

$$g-h = b \cdot r_1 \quad \text{und} \quad h = b \cdot r_2$$

also

$$g = b \cdot r_1 + \overbrace{b \cdot r_2}^{h=} = b(r_1 + r_2)$$

Also $b \in M_1$. Damit ist $M_1 = M_2$.

Beim Übergang von g_0, h_0, $g_0 \geq h_0 \geq 1$ auf $g_1 = g_0 - h_0$, $h_1 = h_0$ ist die Menge der gemeinsamen Teiler von g_0 und h_0 und g_1 und h_1 eine Invariante.

Unser Euklidischer Algorithmus geht dann so:

	g	h
	10	3
$10-3=$	7	3
$7-3=$	4	3
$4-3=$	1	3
	1	2 ($=3-1$)
	1	1 ($=2-1$)
	1	0

und $ggT(10,3) = ggT(1,1) = 1$.

Eingabe von $g, h \geq 1$,

≥ 1.

while $g \,!= h$ // Für $g \neq h$

{

if $g > h$

{

$g = g - h$;

break; // geht zum Ende

} // der Schleife.

$h = h - g$; // hier ist $h > g$.

}

Ausgabe von g oder h.

Die Sprunganweisungen:
break = geht zum Ende des aktuellen Blocks, beendet aktuelle Schleife
continue = beendet den aktuellen Lauf (nicht die ganze Schleife)
return = beendet Methode

Verifikation:

g_0 = Wert von g am Anfang.

h_0 = Wert von h am Anfang.

Sei

$M_0 = \{ b \mid b \mid g_0 , b \mid h_0 \}$.

Finde für $l \geq 1$ der l'te Lauf statt, dann

$$g_l,\ h_l,\ M_l \text{ nach } l\text{'tem Lauf.}$$

Zur # Läufe: Diese ist endlich, denn $g_0, h_0 \geq 1$ und $0 \lneqq g_{l+1} \lneqq g_l$ oder $0 \lneqq h_{l+1} \lneqq h_l$. Also $2 \leq g_{l+1} + h_{l+1} \lneqq g_l + h_l$. Also irgendwann ist Schluß.
Beachte nicht unbedingt $|g_{l+1} - h_{l+1}| \lneqq |g_l - h_l|$.

Invariante: Findet ein l'ter Lauf statt, dann

$$M_0 = M_l$$

und $g_l \geq 1$ und $h_l \geq 1$.

Inspektion des Schleifenrumpfes und Überlegung oben.

Quintessenz: Ist nach l Läufen das Ende erreicht, so ist $g_l = h_l$, wg. Bedingung.

Dann ist

$$M_l = \{ b \mid b \mid \overset{=h_l}{g_l} \} = M_0$$

wegen der Invariante. Also

$$\mathrm{Max}\, M_l = g_l = \mathrm{ggT}(g_0, h_0).$$

Weitere Beschleunigung unter

Einsatz von

Div(g,h), eigentliche Mod(g,h)

Übung: Programmieren Eukl. java

Für $g,h \geq 1$ ist das kleinste gemeinsame Vielfache definiert durch

$$\mathrm{kgV}(g,h) = \min\{b \mid g \mid b \text{ und } h \mid b\}.$$

Dann $\mathrm{kgV}(g,h) \leq g \cdot h$. Sind $p_1, \ldots, p_s$ alle Primfaktoren, die in g oder h vorkommen (mit Einsen ergänzt) und ist

$$g = p_1^{e_1} \cdots p_s^{e_s}, \quad e_s \geq 0$$

$$h = p_1^{a_1} \cdots p_s^{a_s}, \quad a_s \geq 0,$$

dann ist

$$\mathrm{kgV}(g,h) = p_1^{c_1} \cdots p_s^{c_s}, \text{ wobei}$$

$f_i = \max\{e_i, a_i\}$

Folgt mit der Eindeutigkeit der Primfaktorzerlegung von $\mathrm{kgV}(g,h)$ (Übungsaufgabe).

Andererseits ist

$$\mathrm{ggT}(g,h) = p_1^{d_1} \cdots p_s^{d_s}, \quad d_i \geq 0$$

und $d_i = \min\{a_i, e_i\}$. Nun sieht man leicht, daß

$$g \cdot h = \mathrm{ggT}(g,h) \cdot \mathrm{kgV}(g,h)$$

denn

$$\underbrace{a_i + e_i}_{\text{zu } g\cdot h} = \underbrace{\max\{a_i, e_i\}}_{\text{zu kgV}} + \underbrace{\min\{a_i, e_i\}}_{\text{zu ggT.}}$$

Programm für $\mathrm{kgV}(g,h)$ in ggTkgV.java.

Als nächstes kleines Problem behandeln wir das Problem der

ganzzahligen Wurzel.

Für $n \in \mathbb{N}$ ist $a = gW(n)$ eindeutig definiert durch

$$a^2 \leq n \lneqq (a+1)^2 .$$

In anderen Worten, ist

$$M = \{ b \in \mathbb{N} \mid b^2 \leq n \},$$

so ist $gW(n) = \max M$. Also zum Beispiel ist

$$3 = gW(9) = gW(10) = gW(11) \dots = gW(15)$$

$$4 = gW(16) = \dots = gW(24)$$

$$5 = gW(25) = \dots = gW(36) .$$

Anders geschrieben $gW(n) = \lfloor \sqrt{n} \rfloor$

Ganzzahliger Anteil

2.75

wobei $\sqrt{n} \in \mathbb{R}^+$ genommen wird.

Ein einfaches Programm:

n wird eingelesen

$z = 0$

while $z \cdot z \leq n$

{

$z = z+1$

}

Ausgabe z.

n	z
0	0
	1 Ausgabe 1

n	z
3	0
	1
	2 Ausgabe 2

n	z
4	0
	1
	2
	3 Ausgabe 3

Also nicht ganz richtig.

stattdessen

n wird eingelesen

$z = 0$

while $(z+1)\cdot(z+1) \leq n$

{

$z = z + 1$

}

Ausgabe z

n	z
0	0

Ausgabe 0

n	z
1	0
	1

Ausgabe 1

n	z
2	0
	1

Ausgabe 1, da $2 \cdot 2 \not\geq 2$.

m z

4 0

1

2

Ausgabe 2, da $3 \cdot 3 \gneq 4$.

Läufe: Beschränkt durch $\lfloor \sqrt{n} \rfloor$.

Alternativ: $n_0 - z_l^2 \geq 0$ ist invariant und

$n_0 - z_{l+1}^2 \lneq n_0 - z_l^2$.

Invariante: Nach dem l'ten Lauf, sofern er stattfindet, ist

$$z_l^2 \leq n_0 .$$

gilt für $l = 1$ und gilt sie für l, dann für $l+1$.

Quintessenz: Ist die Schleife nach dem l'ten Lauf zu Ende, so ist

$(z_l + 1)^2 \gneq n_0$ wg. Bedingung

und

$z_e^2 \leq n_0$ wg. Invariante.

Also ist $z_e = $ glw(n).

Falls kein Lauf ist $n = 0$ und $z = 0$.
Damit ist die Verifikation des
Programms erbracht.

Das Programm steht in glwEin.java

Zur Effizienzverbesserung ein
anderer Ansatz: Man könnte das
z um $d \geq 1$ hochzählen, dann
aber könnte man die glw(n)
verfehlen. Deshalb arbeiten
wir mit zwei Variablen,
a und b, so daß

immer die Invariante gilt

$a_\ell^2 \leq n$ und $n \lneq b_\ell^2$, und $0 \leq a_\ell \lneq b_\ell$

das heißt

$0 \leq a_\ell \leq \text{Wurzel}(n) \lneq b_\ell$.

Also Wurzel(n) liegt in dem „halboffenen Intervall" „$[a_\ell, b_\ell)$"

und $a_\ell \lneq b_\ell$. Beachte hier $[a_\ell, b_\ell) = \{a_\ell, \dots, b_\ell - 1\}$.

Beachte hier:

$$\underbrace{\left|\{a_\ell, \dots, b_\ell - 1\}\right|}_{\#\text{Elemente}} = b_\ell - a_\ell .$$

Am Anfang ist

$a_0 = 0$ und $b_0 = n + 1$.

Ziel ist es in jeder Runde die Menge in der man sucht, $[a_\ell, b_\ell)$ um $d = d_\ell$ Elemente zu verkleinern. Wie eben in

$$a_{\ell+1} = a_\ell + d \;,\; b_{\ell+1} = b_\ell \quad \text{oder aber}$$

$$a_{\ell+1} = a_\ell \;,\; b_{\ell+1} = b_\ell - d$$

und die Invariante

$$qw(m) \in [a_{\ell+1}, b_{\ell+1})$$

bleibe erhalten.

Ist irgendwann $b_\ell = a_\ell + 1$, dann ist $a_\ell = qw(m)$, denn dann ist

$$a_\ell^2 \leq m \text{ und } (a_\ell + 1)^2 > m.$$

wegen der Invariante.

Wie können uns ein gutes (d.h. möglichst großes) d finden?

Beachten wir, $d=1$ geht immer, solange $|[a_\ell, b_\ell)| = b_\ell - a_\ell \geq 2$ ist.

Ist nun das d so, daß

$$a_\ell + d \leq b_\ell - d,$$

dann gilt

$$(a_\ell + d)^2 \not\geq m \Rightarrow (b_\ell - d)^2 \not\geq m$$

(springen uns links zu weit, dann geht es rechts)

und auch

$$(b_\ell - d)^2 \leq m \Rightarrow (a_\ell + d)^2 \leq m$$

(rechts zu weit, dann links)

Bei d mit $a_\ell + d \le b_\ell - d$ wird (2.29) die Invariante erhalten. Das impliziert, bei d mit

$$2d \le b_\ell - a_\ell = |[a_\ell, b_\ell)|$$

eine Möglichkeit die Invariante:

$$a_{\ell+1} = a_\ell + d, \; b_{\ell+1} = b_\ell \text{ oder}$$

$$a_{\ell+1} = a_\ell, \; b_{\ell+1} = b_\ell - d \text{ usw.}$$

Das größte d, das es tut, ist

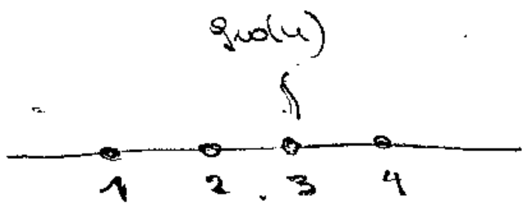
$$d_\ell = \mathrm{Div}(\underbrace{b_\ell - a_\ell}_{\ge 2}, 2) \ge 1$$

folgt $b_\ell - a_\ell \ge 2$.

Beachten wir noch einmal,

ist $a_\ell = 1$, $b_\ell = 4$, dann

$d_\ell = 1$ nach obiger Formel.



Ist etwa $gw(u) = 3$, dann wird das Suchintervall so verkleinert:

a	b	d	
1	4	1	a_e, b_e, d_e
			Ein Lauf.
2	4	1	
			Ein Lauf
3	4		

Schluß

$gw(u) \in [3,4)$ also $gw(u) = 3$

Ist $gw(u) = 2$, dann

a	b	d
1	4	1
2	4	1
2	3	

Beachte hier:

$1 + 1 \leq gw(u)$

und

$4 - 3 \geq gw(u)$,

deshalb 2

Möglichkeiten.

oder auch

a	b	d
1	4	1
1	3	1
2	3	

Man beachte auch, daß die Wahl von $d = \left\lceil \left(\frac{b_e - a_e}{2} \right) \right\rceil$ die Invariante nicht zwingend erhält: $gw(u) = 2$ und

a	b	d
1	4	2
1	2	

geht nicht da $2 \notin [1,2)$

a	b	d
1	4	2
3	4	

geht erst recht nicht.

Das gilt ebenso bei größeren Intervallen, wenn das gesuchte Element $gw(m) = \left\lfloor \frac{b-a}{2} \right\rfloor$ ist. (Übungsaufgabe)

Damit haben wir folgendes Programm zu Binf.java. (Es ist ein Beispiel der binären Suche.

Eingabe von n;

$a = 0$

$b = n+1$

```
while a+1 ≨ b       // Vielleicht besser
{                   //   a+1 ≠ b. Beachte
   d = (b-a)/2;     // a+1 ≨ b ist
   if (a+d)^2 ≤ n   // gleichbedeutend zu
   {                //   a ≤ b-2 oder
      a = a+d       //   2 ≤ b-a.
   }
   if (b-d)^2 ≩ n   // Beachte, daß beide
                    // Fälle gleichzeitig
   {                // wahr werden können
      b = b-d       // a=1, b=6, d=2
   }                // und Wu(n) = 3.
}

Ausgabe von a.      // Hier ist a+1 = b
                    // und damit nur noch
                    // a im Suchintervall.
```

86

Invariante: $\mathrm{Iwo}(m) \in [a_e, b_e)$.

$b_{e+1} - a_{e+1} \lneq b_e - a_e$ also

endlich viele Läufe.

Das war ja die Invariante!

Abbruch bei $a_e + 1 = b_e$ also

$b_e = a_e + 1$, da $\mathrm{Iwo}(u) \in [a_e, b_e)$.

Dieses Beispiel zeigt deutlich

den Nutzen der Invarianten.

Nun mit dem vagen Prinzip:

1. Rechne die Mitte aus.

2. Ist $(\text{Mitte})^2$ größer als m

suche links weiter.

3. Ist $(\text{Mitte})^2$ kleiner als m

suche rechts weiter

ist es nicht ganz leicht, auf den

Algorithmus korrekt zu kommen.

Als letztes Beispiel die Exponentiation. Da a^b schnell zu groß wird, schauen wir uns einmal $0 \leq \mathrm{Mod}(a^b, c) \lneq c$ an.

Zunächst einmal ist

$$\mathrm{Mod}(g \cdot h, c)$$

$$= \mathrm{Mod}(g, c) \odot_c \mathrm{Mod}(h, c),$$

wobei $g \odot_c h = \mathrm{Mod}(g \cdot h, c)$.

Denn ist

$$g = q_1 \cdot c + \overbrace{r_1}^{\mathrm{Mod}(g,c)}, \quad h = q_2 \cdot c + \overbrace{r_2}^{\mathrm{Mod}(h,c)},$$

dann ergibt sich

$g \cdot h =$

$= (q_1 c + r_1) \cdot (q_2 \cdot c + r_2)$

$= q_1 \cdot c (q_2 c + r_2) + r_1 \cdot q_2 \cdot c + r_1 \cdot r_2$

$= q' \cdot c + r_1 \cdot r_2$ Mod $(g \cdot h, c)$

$= q' \cdot c + q'' \cdot c + r'$

Beidemal das c Mod $(r_1 \cdot r_2, c)$

wobei $r_1 \cdot r_2 = q'' \cdot c + r'$

Dann Mod (a^b, c) nach folgendem

Prinzip: $a_1 = a \% c$,

$a_2 = (a \cdot a_0) \% c$, $a_3 = a \cdot a_1 \% c$,

$a_4 = a \cdot a_2 \% c, \ldots, a_b = a \cdot a_{b-1} \% c$.

zu EinfModExp.java. Es gilt effizientes

Dazu stellen wir uns b in Binärdarstellung vor

$$b = \sum_{i=0}^{n-1} z_i \cdot 2^i = (z_{n-1} \ldots z_0)_2 .$$

Dann (wir vergessen jetzt einmal das $-\% q$)

$$a^b = a^{\sum z_i 2^i}$$

$$= a^{z_{n-1} 2^{n-1}} \cdot a^{z_{n-2} \cdot 2^{n-2}} \cdots a^{z_1 2^1} \cdot a^{z_0 \cdot 2^0}$$

und beobachten, daß

$$a^{(2^j)} = a^{(2^{j-1}) \cdot 2}$$

$$= \left(a^{(2^{j-1})}\right)^2 = a^{(2^{j-1})} \cdot a^{(2^{j-1})}$$

ist. Wir bekommen also 2^{j-1} Faktoren a mit einer Multiplikation hin.

9.30

Vorgehensweise: Generieren

$a, a^2, a^4, a^8, \dots, a^{2^{n-1}}$.

Wenn $s_i = 1$ multiplizieren

mit $a^{(2^i)}$ zu dem Teilergebnis.

Selbstverständlich alles modulo.

```
ahilf = 1;
while b ≠ 0
{
    while (b%2 == 0)
    {
        a = (a * a) % q;
        b = b/2
    }
    ahilf = (ahilf * a) % q
    b--
}
```

Ausgabe von ahilf.

Hier die Invarianten zu finden ist nicht ganz so offensichtlich.

Innere Schleife

$a^{(0)} \geq 1$, $b^{(0)} \geq 1$, $a^{(l)}, b^{(l)}$ wie gehabt.

Invariante ist etwa (alles modulo q):

$$\left(a^{(l)}\right)^{b^{(l)}} = \left(a^{(0)}\right)^{b^{(0)}}.$$

denn $\left(a^{(l+1)}\right)^{b^{(l+1)}} = \left(a^{(l)} \cdot a^{(l)}\right)^{b^{(l)}/2} = \left(a^{(l)}\right)^{b^{(l)}} \underset{\text{Inv. für } l}{=} \left(a^{(0)}\right)^{b^{(0)}}$

Inspektion der inneren Schleife

Äußere Schleife

$a_0 \geq 1$, $b_0 \geq 1$, a_l, b_l, hilf_l wie gehabt, aber bezogen auf die äußere Schleife:

2.92

Die Invariante lautet hier:
Am Ende des l'ten Laufs der äußeren Schleife gilt

$$ahilf_l \cdot a_l^{b_l} = a_0^{b_0}.$$

Bedeutet eben ahilf$_l$ enthält " $a_0^{b_0} / a_l^{b_l}$ ".

$l = 1$, dann z. Nach innerer Schleife haben wir das a_1 der äußeren Schleife und das $b_1 + 1$ (!) der äußeren Schleife. Es ist wegen Invariante innerer Schleife

$$a_1^{b_1+1} = a_0^{b_0}$$

Die Anfangswerte der inneren Schleife sind die Gesamtanfangswerte a_0, b_0.

Dann weiter.

$$ahilf_1 = a_1, \text{ da } ahilf_0 = 1.$$

Dann

$$\underbrace{ahilf_1 \cdot a_1^{b_1}}_{= a_1^{b_1+1}} = a_0^{b_0}, \text{ Invariante gilt.}$$

Wegen $a_1^{b_1+1} = a_0^{b_0}$,

(2.33)

Induktionsschluß: Invariante für l,

$$\text{ahilf}_l \cdot a_l^{b_l} = a_0^{b_0}.$$

Wir betreten die innere Schleife mit

a_l, b_l (statt a_0, b_0!)

Am Ende der inneren Schleife haben wir bereits das a_{l+1} und das $b_{l+1}+1$ (!).

Wegen Inv. innerer Schleife ist

$$a_l^{b_l} = a_{l+1}^{b_{l+1}+1}$$

Das sind ja die Eingangswerte der inneren Schleife.

Dann gilt

$$\text{ahilf}_{l+1} = a_{l+1}^{b_{l+1}} =$$

Wegen der Zuweisung!

$$= \text{ahilf}_l \cdot a_{l+1} \cdot a_{l+1}^{b_{l+1}}$$

$$= \text{ahilf}_l \cdot a_{l+1}^{b_{l+1}+1}$$

Inv. innere Schleife, 2.34

$$a_l^{b_l} = a_{l+1}^{b_{l+1}+1}$$

$$= \mathrm{ahilfe}_l \cdot a_l^{b_l}$$

$$= a_0^{b_0}.$$

Quintessenz: Ist $b_l = 0$, das muß am Ende der Schleife sein, dann ist $\mathrm{ahilfe}_l = a_0^{b_0}$.

Bemerkung zur Multiplikation im 2'er Komplement.

2'er Kpl. von a = Binärdarstellg. von $\mathrm{Mod}(a, 2^n)$.

Es ist

$$\mathrm{Mod}(a \cdot b, 2^n) = \mathrm{Mod}(a, 2^n) \odot_{2^n} \mathrm{Mod}(b, 2^n).$$

Können wie üblich auf den 2'er Kpl'ten multiplizieren, ebenso wie addieren!