VL-03: Registermaschinen

(Berechenbarkeit und Komplexität, WS 2018)

Gerhard Woeginger

WS 2018, RWTH

Organisatorisches

- Nächste Vorlesung:
  Freitag, November 2, 16:30–18:00 Uhr, Audimax
- Webseite:
  `http://algo.rwth-aachen.de/Lehre/WS1819/BuK.php`

Wiederholung

Wdh.: k-Band- vs 1-Band-TM

Satz

Eine k-Band-TM M mit Zeitbedarf $t(n)$ und Platzbedarf $s(n)$ kann von einer 1-Band-TM M' mit Zeitbedarf $O(t^2(n))$ und Platzbedarf $O(s(n))$ simuliert werden.

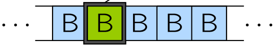
···	B	*	*	*	*	A	r	b	e	i	t	s	b	a	n	d	1	B	B	B	···

···	B	A	r	b	e	i	t	s	b	a	n	d	2	*	*	*	*	B	B	B	···

Simuliert durch

···	B	*	*	*	*	A	r	b	e	i	t	s	b	a	n	d	1	B	B	B	···
···	B	B	B	B	B	B	#	B	B	B	B	B	B	B	B	B	B	B	B	B	···
···	B	A	r	b	e	i	t	s	b	a	n	d	2	*	*	*	*	B	B	B	···
···	B	#	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	···

Wdh.: Gödelnummer $\langle M \rangle$



$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1, B\}$

$Q = \{q_1, q_2, q_3\}$

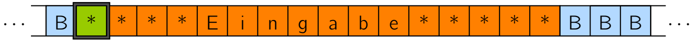
Start q_1 Ende q_2 Blank B

δ	0	1	B
q_1	(q_1, B, R)	(q_3, B, R)	(q_2, B, N)
q_3	$(q_1, 1, R)$	$(q_2, 0, R)$	(q_1, B, L)

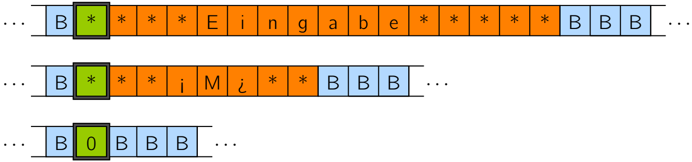
111 0101010001000 11 0100100010001000 11 010001001000100 11
00010101001000 11 000100100101000 11 000100010100010 111

Wdh.: Universelle TM

Simulierte Turingmaschine M



Initialisierung der universellen Maschine U



Wdh.: Universelle TM

Laufzeit der universellen TM

- Bei Eingabe $\langle M \rangle w$ simuliert U die TM M auf Wort w.
- Jeder Schritt von M wird dabei von U in $f(|\langle M \rangle|)$ Zeit simuliert.
- Wenn $|\langle M \rangle|$ als Konstante angesehen wird,
  so simuliert U die TM M mit konstantem Zeit- und Platzverlust.

Wdh.: Die Church-Turing These

Alonzo Church und Alan Turing formulierten in den 1930er Jahren die folgende These:

Church-Turing These

Die Klasse der TM-berechenbaren Funktionen stimmt mit der Klasse der "intuitiv berechenbaren" Funktionen überein.

Sprachliche Übereinkunft für den Rest der Vorlesung:
berechenbare Funktion= TM-berechenbare Funktion= rekursive Funktion
entscheidbare Sprache= TM-entscheidbare Sprache= rekursive Sprache

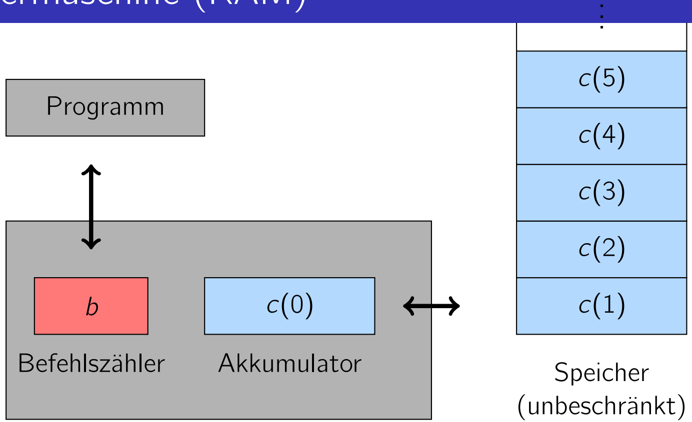
Vorlesung VL-03
Registermaschinen

- Registermaschinen (RAMs)
- Simulation von RAM durch TM
- Simulation von TM durch RAM

- Collatz Problem

Registermaschinen (RAM)

Registermaschine (RAM)



Befehlssatz:

LOAD, STORE, ADD, SUB, MULT, DIV

INDLOAD, INDSTORE, INDADD, INDSUB, INDMULT, INDDIV

CLOAD, CADD, CSUB, CMULT, CDIV

IF c(0)?x THEN GOTO j (mit ? in {=,<,<=,>,>=})

GOTO, END

RAM: Einige ausgewählte Befehle (1)

LOAD i:	$c(0) := c(i)$,	$b := b + 1;$
INDLOAD i:	$c(0) := c(c(i))$,	$b := b + 1;$
CLOAD i:	$c(0) := i$,	$b := b + 1;$
STORE i:	$c(i) := c(0)$,	$b := b + 1;$
INDSTORE i:	$c(c(i)) := c(0)$,	$b := b + 1;$
ADD i:	$c(0) := c(0) + c(i)$,	$b := b + 1;$
CADD i:	$c(0) := c(0) + i$,	$b := b + 1;$
INDADD i:	$c(0) := c(0) + c(c(i))$,	$b := b + 1;$
	$\vdots$	
SUB i:	$c(0) := \max\{0, c(0) - c(i)\}$	$b := b + 1;$
	$\vdots$	
MULT i:	$c(0) := c(0) * c(i)$	$b := b + 1;$
	$\vdots$	

RAM: Einige ausgewählte Befehle (2)

DIV i: $$c(0) := \begin{cases} \lfloor c(0)/c(i) \rfloor & \text{falls } c(i) \neq 0, \\ 0 & \text{sonst} \end{cases}, \quad b := b+1;$$

GOTO j: $$b := j$$

IF $c(0) = x$ GOTO j: $b := j$ falls $c(0) = x$, sonst $b := b+1$;

END.

RAM: Funktionsweise

- Der Speicher der RAM ist unbeschränkt und besteht aus dem Akkumulator $c(0)$ und aus den Registern $c(1), c(2), c(3), \ldots$
- Die Inhalte der Register sind natürliche Zahlen, die beliebig gross werden können.

- Die Eingabe besteht ebenfalls aus natürlichen Zahlen, die zu Beginn in den ersten Registern abgespeichert sind.
- Alle anderen Register sind mit 0 initialisiert.
- Der Befehlszähler startet mit dem Wert 1. Ausgeführt wird jeweils der Befehl in derjenigen Zeile, auf die der Befehlszähler verweist.

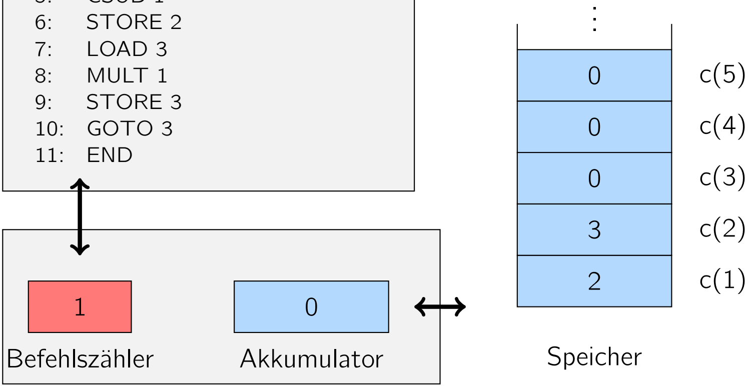
- Die Rechnung stoppt, sobald der Befehl END erreicht ist.
- Die Ausgabe befindet sich nach dem Stoppen in den ersten Registern.

RAM: Beispielprogramm

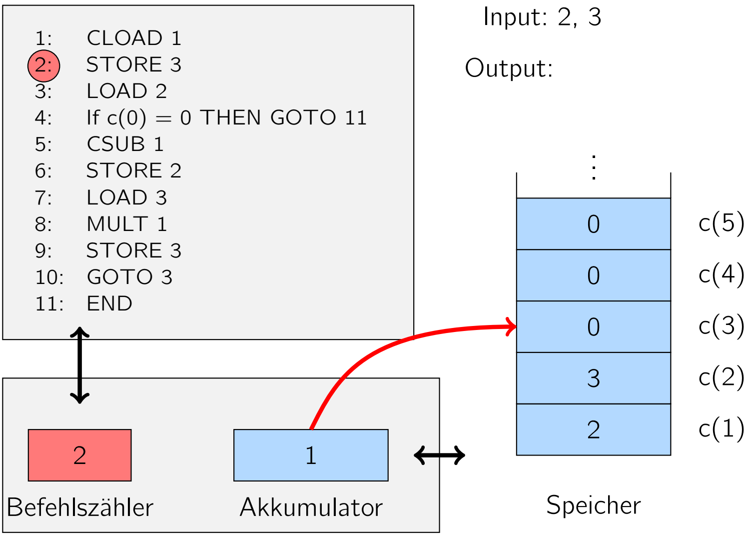
```
1:   CLOAD 1
2:   STORE 3
3:   LOAD 2
4:   If c(0) = 0 THEN GOTO 11
5:   CSUB 1
6:   STORE 2
7:   LOAD 3
8:   MULT 1
9:   STORE 3
10:  GOTO 3
11:  END
```

Input: 2, 3

Output:



RAM: Beispielprogramm



RAM: Anmerkungen

Auf einer RAM können wir alle Befehle realisieren (wie beispielsweise Schleifen und Rekursionen), die wir von höheren Programmiersprachen her gewohnt sind,

Modelle für die Rechenzeit

- Uniformes Kostenmaß: Jeder Schritt zählt als eine Zeiteinheit.
- Logarithmisches Kostenmaß: Die Laufzeitkosten eines Schrittes sind proportional zur binären Länge der Zahlen in den angesprochenen Registern.

Simulation von RAM durch TM

Simulation RAM durch TM

Satz

Für jede im logarithmischen Kostenmass $t(n)$-zeitbeschränkte RAM R gibt es ein Polynom q und eine $q(n + t(n))$-zeitbeschränkte TM M, die R simuliert.

Im Beweis können wir *o.B.d.A.* für die Simulation eine 2-Band-TM statt einer 1-Band-TM verwenden. Warum?

Vorbemerkungen (1): Polynome

- Seien $\alpha, \beta, \gamma \in \mathbb{N}$ geeignet gewählte Konstanten.
- Wir werden zeigen: Die Laufzeit der Simulation der RAM mit Laufzeitschranke $t(n)$ durch eine 2-Band-TM ist nach oben beschränkt durch $t'(n) = \alpha(n + t(n))^\beta$.
- Die 2-Band-TM mit Laufzeitschranke $t'(n)$ kann nun wiederum mit quadratischem Zeitverlust durch eine (1-Band-)TM simuliert werden, also mit einer Laufzeitschranke der Form $t''(n) = \gamma(t'(n))^2$.
- Für die Simulation der RAM auf der (1-Band-)TM ergibt sich somit eine Laufzeitschranke von

$$t''(n) \;=\; \gamma(t'(n))^2 \;=\; \gamma\left(\alpha(n + t(n))^\beta\right)^2 \;=\; \gamma\alpha^2 \cdot (n + t(n))^{2\beta}.$$

- Diese Laufzeitschranke ist polynomiell in $n + t(n)$, weil sowohl der Term $\gamma\alpha^2$ als auch der Term 2β konstant sind.

Vorbemerkungen (2): Polynome

Beobachtung

Die Klasse der Polynome ist unter Hintereinanderausführung abgeschlossen.

Mit anderen Worten:
Wenn sowohl die Abbildung $x \mapsto p(x)$ als auch die Abbildung $x \mapsto q(x)$ Polynome sind, dann ist auch die Abbildung $x \mapsto q(p(x))$ ein Polynom.

Deshalb können wir eine *konstante Anzahl* von Simulationen, deren Zeitverlust jeweils polynomiell nach oben beschränkt ist, ineinander schachteln und erhalten dadurch wiederum eine Simulation mit polynomiell beschränktem Zeitverlust.

Beweis (1)

Beweis des Satzes

- Wir verwenden eine 2-Band-TM, die die RAM schrittweise simuliert. Auf Band 1 werden die einzelnen Befehle simuliert, und auf Band 2 wird der Inhalt aller verwendeten Register abgespeichert.
- Das RAM-Programm P bestehe aus p Programmzeilen.
- Für jede Programmzeile schreiben wir ein TM-Unterprogramm. Es sei M_i das Unterprogramm für Programmzeile i, $1 \leq i \leq p$.
- Ausserdem spezifizieren wir ein Unterprogramm M_0 für die Initialisierung der TM, und ein Unterprogramm M_{p+1} für die Aufbereitung der Ausgabe des Ergebnisses.

Beweis (2)

Abspeichern der RAM-Konfiguration auf der TM:

- Den Befehlszähler kann die TM im Zustand abspeichern, da die Länge des RAM-Programms konstant ist.
- Die Registerinhalte werden wie folgt auf Band 2 abgespeichert:

$$\#\#0\#\,\mathrm{bin}(c(0))\#\#\,\mathrm{bin}(i_1)\#\,\mathrm{bin}(c(i_1))\#\#\ldots$$

$$\ldots\#\#\,\mathrm{bin}(i_m)\#\,\mathrm{bin}(c(i_m))\#\#\#,$$

wobei $0, i_1, \ldots, i_m$ die Indizes der benutzten Register sind.

Beobachtung

Der Platzbedarf auf Band 2 ist durch $O(n + t(n))$ beschränkt, weil die RAM für jedes neue Bit, das sie erzeugt, mindestens eine Zeiteinheit benötigt.

Beweis (3)

Rechenschritt für Rechenschritt simuliert die TM nun die Konfigurationsveränderungen der RAM.

Dazu ruft die TM jeweils das im Programmzähler b spezifizierte Unterprogramm M_b auf.

Das Unterprogramm M_b

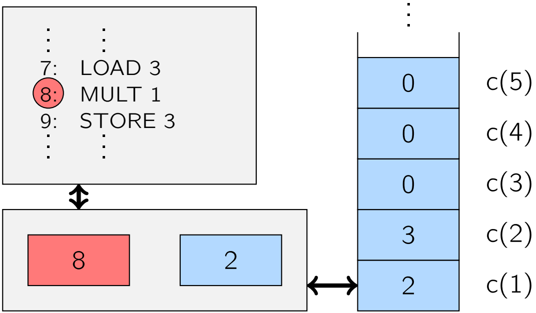
- kopiert den Inhalt der in Programmzeile b angesprochenen Register auf Band 1,
- führt die notwendigen Operationen auf diesen Registerinhalten durch,
- kopiert dann das Ergebnis in das in Zeile b angegebene Register auf Band 2 zurück, und
- aktualisiert zuletzt den Programmzähler b.

Beweis (4)

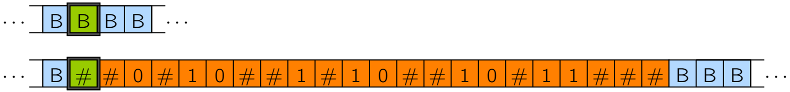
Laufzeitanalyse:

- Die Initialisierung erfordert Zeit $O(n)$.
- Alle Unterprogramme haben eine Laufzeit, die polynomiell in der Länge des aktuellen Wortes auf Band 2 beschränkt ist.
  Also: Eine Laufzeit polynomiell in $n + t(n)$.
- Somit ist auch die Gesamtlaufzeit der Simulation polynomiell in $n + t(n)$ beschränkt.
- Ende des Beweises.

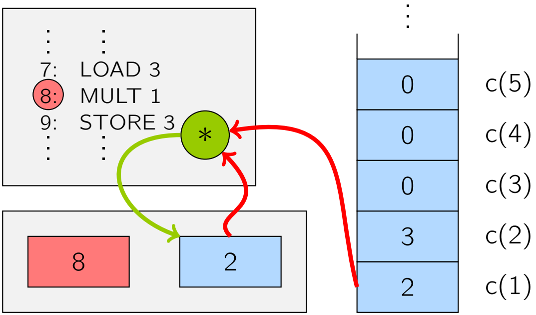
Simulierte Registermaschine M



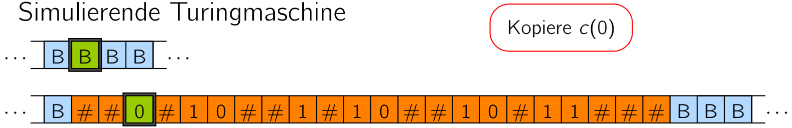
Simulierende Turingmaschine



Simulierte Registermaschine M



Simulierende Turingmaschine



Simulation von TM durch RAM

Simulation TM durch RAM

Satz

Jede $t(n)$-zeitbeschränkte TM kann durch eine RAM simuliert werden, die zeitbeschränkt ist durch

- $O(t(n) + n)$ *im uniformen Kostenmass und*
- $O((t(n) + n) \cdot \log(t(n) + n))$ *im logarithmischen Kostenmass.*

Beweis (1)

- O.B.d.A. nehmen wir an, dass es sich um eine TM mit einseitig beschränktem Band handelt, dessen Zellen mit $0, 1, 2, 3, \ldots$ durchnummeriert sind.
  (Übungsaufgabe)
- Die Zustände und Zeichen werden ebenfalls durchnummeriert und mit ihren Nummern identifiziert, so dass sie in den Registern abgespeichert werden können.
- Register 1 speichert den Index der Kopfposition.
- Register 2 speichert den aktuellen Zustand.
- Die Register $3, 4, 5, 6, \ldots$ speichern die Inhalte der besuchten Bandpositionen $0, 1, 2, 3, \ldots$

Beweis (2)

Die TM wird nun Schritt für Schritt durch die RAM simuliert.

Auswahl des richtigen TM-Übergangs

Die RAM verwendet eine zweistufige if-Abfrage:

- Auf einer ersten Stufe von $|Q|$ vielen if-Abfragen wird der aktuelle Zustand selektiert.
- Für jeden möglichen Zustand gibt es dann eine zweite Stufe von $|\Gamma|$ vielen if-Abfragen, die das gelesene Zeichen selektieren.

Durchführung des TM-Übergangs

Je nach Ausgang der if-Abfragen aktualisiert die RAM

- den TM-Zustand in Register 2,
- die TM-Bandinschrift in Register $c(1)$ und
- die TM-Bandposition in Register 1.

Simulierte Turingmaschine M

B	1	1	2	1	B	B	B	…

δ	1	2	B
q_1			
q_2			
q_3		$(q_2, 1, R)$	

q_3

Simulierende Registermaschine

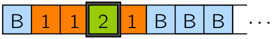
```
 6:  LOAD 2
 7:  IF c(0) = 1 THEN GOTO 23
 8:  IF c(0) = 2 THEN GOTO 32
 9:  IF c(0) = 3 THEN GOTO 41

41:  INDLOAD 1
42:  IF c(0) = 1 THEN GOTO 77
43:  IF c(0) = 2 THEN GOTO 85
```

6	0

⋮

1	c(6)	
2	c(5)	
1	c(4)	
1	c(3)	
3	c(2)	Zustand
5	c(1)	Kopfposition

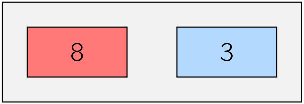
Simulierte Turingmaschine M



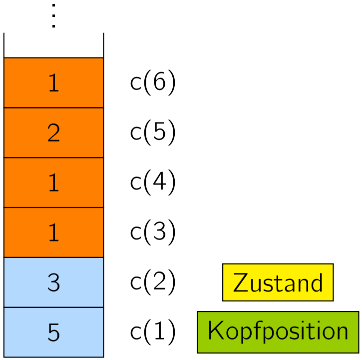
δ	1	2	B
q_1			
q_2			
q_3		$(q_2, 1, R)$	

q_3

```
6:  LOAD 2
7:  IF c(0) = 1 THEN GOTO 23
8:  IF c(0) = 2 THEN GOTO 32
9:  IF c(0) = 3 THEN GOTO 41

41: INDLOAD 1
42: IF c(0) = 1 THEN GOTO 77
43: IF c(0) = 2 THEN GOTO 85
```



Simulierende Registermaschine



Beweis (3a)

Laufzeitanalyse im uniformen Kostenmodell:

- Die Initialisierung kann in Zeit $O(n)$ durchgeführt werden.
- Die Simulation jedes einzelnen TM-Schrittes hat konstante Laufzeit.
- Insgesamt ist die Simulationszeit somit $O(n + t(n))$.

Beweis (3b)

Laufzeitanalyse im logarithmischen Kostenmodell:

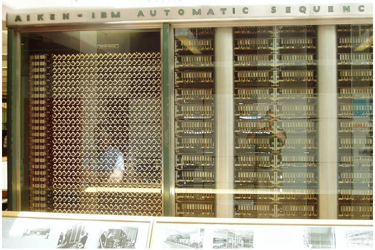
- Die in den Registern gespeicherten Zahlen repräsentieren Zustände, Zeichen und Bandpositionen.
- Zustände und Zeichen haben eine konstante Kodierungslänge.
- Die Bandpositionen, die während der Simulation angesprochen werden, sind durch $\max\{n, t(n)\} \leq n + t(n)$ beschränkt. Die Kodierungslänge dieser Positionen ist also $O(\log(t(n) + n))$.
- Damit kann die Simulation jedes einzelnen TM-Schrittes in Zeit $O(\log(t(n) + n))$ durchgeführt werden.
- Insgesamt ergibt sich somit eine Simulationszeit von $O((t(n) + n)\log(t(n) + n))$.

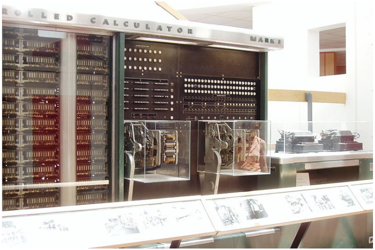
$\square$

Zusammenfassung

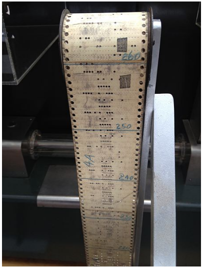
- Die Mehrband-TM kann mit quadratischem Zeitverlust durch eine (1-Band-)TM simuliert werden.
- TM und RAM (im logarithmischen Kostenmodell) können sich gegenseitig mit polynomiellem Zeitverlust simulieren.
- Wenn es also nur um Fragen der Berechenbarkeit von Problemen (oder um ihre Lösbarkeit in polynomieller Zeit) geht, so können wir wahlweise auf die TM, die Mehrband-TM oder die RAM zurückgreifen.

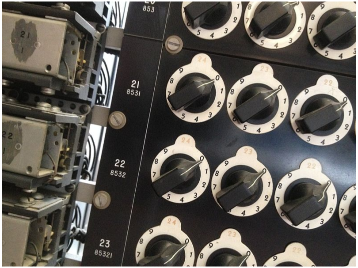
Harvard Mark I (1944)

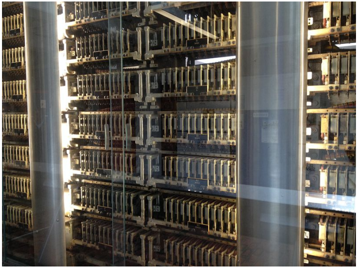




Harvard Mark I (1944)



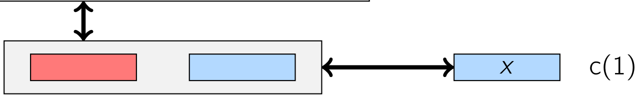




Das Collatz Problem

```
 1:  LOAD 1
 2:  IF c(0) > 1 THEN GOTO 4
 3:  END
 4:  CADD 1
 5:  CDIV 2
 6:  CMULT 2
 7:  SUB 1
 8:  IF c(0) > 0 THEN GOTO 13
 9:  LOAD 1
10:  CDIV 2
11:  STORE 1
12:  GOTO 1
13:  LOAD 1
14:  CMULT 3
15:  CADD 1
16:  STORE 1
17:  GOTO 1
```

$$x \leftarrow \begin{cases} x/2 & \text{wenn } x \text{ gerade} \\ 3x+1 & \text{wenn } x \text{ ungerade} \end{cases}$$



Das Collatz Problem (1)

$$x \leftarrow \begin{cases} x/2 & \text{wenn } x \text{ gerade} \\ 3x+1 & \text{wenn } x \text{ ungerade} \end{cases}$$

Mit dieser Iterationsgleichung erhält man z.B. die Zahlenfolgen

- 1, 4, 2, 1, ...
- 2, 1, 4, 2, 1,...
- 3, 10, 5, 16, 8, 4, 2, 1, ...
- 5, 16, 8, 4, 2, 1, 4, 2, 1
- 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26,
  13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1

Offenes Problem

Hält die obige Registermaschine auf allen Eingaben?

Das Collatz Problem (2)

Statistik der Zahlenfolgenlängen bei Eingaben bis zu 100 Millionen: