Lösen von (linearen) Rekursionsgleichungen

Zusatzmaterial zur Vorlesung
Informatik 1, WS 2006/07

bei

Prof. Niels Pinkwart
Institut für Informatik
Technische Universität Clausthal

Version vom: 27. Februar 2007

Autor: Robert F. P. Hartmann
gesetzt in $\LaTeX\,2_\varepsilon$ und pdf$\LaTeX$

Blaise Pascal (1623 - 1662) in **Von der Methode der geometrischen, d.h. methodischen und vollkommenen Beweise**[a]:

„Ich will also das, was ein Beweis ist, [. . .] verständlich machen [. . .] Vorher muß ich aber die Idee von einer noch höheren und vollkommeneren Methode geben, wohin die Menschen jedoch niemals gelangen können: denn was über die Geometrie geht, geht über uns hinaus; trotzdem ist es notwendig, etwas davon zu sagen, obgleich es unmöglich zu verwirklichen ist. Diese wahre Methode, welche Beweise in höchster Vollendung führen würde, wenn es möglich wäre, sie zu erreichen, würde in zwei Hauptsachen bestehen:

Einmal, keinen Begriff zu verwenden, dessen Sinn man nicht vorher deutlich erklärt hätte, zum anderen, niemals eine Behauptung aufzustellen, die man nicht aus schon bekannten Wahrheiten bewiesen hätte; d.h. mit einem Wort, alle Begriffe zu definieren und alle Behauptungen zu beweisen.“

[a] Zitat aus Karl Vorländer: **Geschichte der Philosophie, Band 4, Philosophie der Neuzeit**; Rowohlt Taschenbuchverlag Band 261/62, 1966 München

Vorwort:

Im Grundstudium kamen in den Vorlesungen nur die „Ratemethode“ zur Lösung von Rekursionsgleichungen dran. Das Thema wurde dann in der großen Übung [Sta01], die vom Mitarbeiter gehalten wurde, vertieft. In einigen Vorlesungen des Hauptstudiums gab es kurze Abschnitte über Rekursionsgleichungen. [Har05], [Wun01], [Brü03]
Am besten hatte mir damals die Erklärung meines Tutors Steffen Pristerjahn gefallen [Pri02].

Ich hoffe, dass mit meiner Übersicht mindestens genauso gut Arbeiten kann, wie ich mit Steffens Übersicht es konnte.

Habt Ihr einen Fehler gefunden, so schickt mir doch bitte einen Hinweis per E-Mail an: Robert.Hartmann@tu-clausthal.de

Robert Hartmann (27. Februar 2007)

Inhaltsverzeichnis

1 Lineare Rekursionsgleichungen

Zur Berechnung der genauen Komplexität rekursiver Algorithmen werden häufig lineare Rekursionsgleichungen benötigt.

1.1 Definitionen

Definition 1.1 (Menge der polynomialen Funktionen)**.** Die Menge aller polynomialen Funktionen $p(n) = \sum c_s \cdot n^s$ heiße $\mathbb{P}$

Definition 1.2 (Lineare Rekursionsgleichung)**.** Lineare Rekursionsgleichungen sind Gleichungen der Form

$$a_k t_n + a_{k-1} t_{n-1} + \ldots + a_1 t_{n-k-1} + a_1 t_{n-k} = s(n) \tag{1.1}$$

mit *Unbekannten* $t_\mu (\mu \in n-k, \ldots, n)$ und *konstanten Koeffizienten* a_ν $(\nu \in \{0, \ldots, k\})$, wobei $a_0 \neq 0$ und $a_k \neq 0$. $s(n)$ ist eine *Störfunktion.*

Definition 1.3 (Standardform einer polynomialen Störfunktion)**.** Ist $s(n) \in \mathbb{P}$ ein Polynom und ist $s(n) = \sum_{j=0}^{h} (b_j^n \cdot p_j(n))$ und $p_j(n)$ ein Polynom mit dem Grad d_j und b_j sind Konstanten so steht $s(n)$ in Standardform.

Definition 1.4 (Standardform der Rekursionsgleichung mit $s(n) \in \mathbb{P}$)**.** Ist $s(n) \in \mathbb{P}$ und hat die Rekursionsgleichung die Form

$$\sum_{i=0}^{k} a_i \cdot t_{n-k+i} = \sum_{j=0}^{h} (b_j^n \cdot p_j(n)) \tag{1.2}$$

so ist sie in Standardform.

Satz 1.1. *Jede polynomiale Störfunktion lässt sich ohne Probleme in Standardform bringen. (Hier ohne Beweis.)*

1.2 Aufstellen der Rekursionsgleichung

1.2.1 Beispiel: Fibonaccizahlen, Rekursionsgleichung aufstellen

Definition 1.5 (FIBONACCIzahlen, Index; F, $\mathbb{F}$)**.**

α) Die durch

$$F_n = n \text{ für } n = 0, 1 \text{ und } F_{n+1} = F_n + F_{n-1} \text{ für } n \in \mathbb{N} \tag{1.3}$$

(rekursiv) definierte Zahlen heißen FIBONACCIzahlen [1] (Fz) — gelegentlich auch LUCAS [2] -Zahlen, diese jedoch oft für ähnliche Folgen oder Verallgemeinerungen; — früher statt F_n meist u_n; n wird gelegentlich als **Index** von F_n bezeichnet.

β) Die Funktion F mit $F(n) \eqqcolon F_n (n \in \mathbb{N}_0)$ werde **Fibonaccifunktion** oder **-folge** (Ff) genannt, also

$$F = (F_n)_{n \in \mathbb{N}_0},$$

und es sei

$$\mathbb{F} = F\mathbb{N}_0$$

Beispiel 1.1. Um die *Zeitkomplexität* eines Programms zubestimmen werden im wesentlichen die Anzahl der Rechenoperationen gezählt, die der entsprechende Algorithmus vorschreibt. Geht man davon aus, dass jede Operation gleich viele Zeiteinheiten verbraucht, verwendet man das sogenannte *uniforme Zeitkriterium*. Notiert man das Zeitverhalten eines rekursiven Algorithmus, einer rekursiv definierten Funktion oder Methode, auf, erhält man eine Rekursionsgleichung. Im Folgenden wird das Zeitverhalten des Algorithmus der $n - ten$ FIBONACCIzahl betrachtet.

---

[1] LEONARDO PISANO FIBONACCI [LEONARDO VON PISA, genannt FIBONACCI] gelegentlich auch LEONARDO BIGOLLO [Reisender] (*≈ 1170 Pisa?, †≈ 1250 Pisa?), lange Zeit in Bugia (daher kommt das Lehnwort „Bougieren"!) in Algerien [heute Bejaïa, röm. Saldae] und auf Reisen; bedeutender Mathematiker: Zahlentheorie, Geometrie, Algebra, Dezimalsystem (Liber abbaci 1202 - dazu HEINZ LÜNEBURG: Leonardi Pisani Liber Abbaci oder Lesevergnügen eines Mathematikers, Mannheim et al. 1993 2.Auflage -; siehe Bemerkung 1b).

Jedoch schon bei indischen Mathematikern: GOPALA (≈ 1135) und 1150 bei ACHARYA HEMCHANDRA (*1089 Dhandhuka [≈ 50 km südwestlich von Ahmadabad, die größte Stadt von Gujarat], †1173 Gujarat); dieser Gelehrte zählte die Möglichkeiten eine Zeile der Länge n (kurze Silben) in kurze und lange (2 kurze Silben) aufzuteilen - , aber Bildungsgesetz bereits im 7. Jahrhundert von Indern untersucht.

Die Fibonaccizahlen $13, 21, 34$ und 55 sind bereits realisiert im Theater von EPIDAUROS ($\approx -300, \approx -170$): BENNO ARTMANN: „Euclid - The Creation of Mathematics" Springer 1999, S. 239 f.

[2] FRANCOIS EDOUARD ANATOLE LUCAS (*4.4.1842 Amiens, †3.10.1891 Paris [an Wundrose (Erysipel)]), Mathelehrer in Paris am Lycée Saint Souis und am Lycée Charlemagne: Zahlentheorie (Primalitätstest für MERSENNEsche Zahlen), Unterhaltungsmathematik (Turm von Hanoi)

Algorithmus 1. *Zur Berechnung der $n-ten$ Fibonacci*zahl:*

```
fib(n)
{
    case(n)
  {
    0: return 1;
    1: return 1;
    default: return fib(n-1) + fib(n-2);
  }
 }
```

Man benötigt in den Fällen $n = 0$ und $n = 1$ zwei bzw. drei Operationen für Vergleiche und Rückgabe.
Wenn t_n die Komplexität von $fib(n)$ angibt, dann gilt für $n \geq 3$

$$t_n = t_{n-1} + t_{n-2} + 3$$

Insgesamt also

$$t_n = t_{n-1} + t_{n-2} + 3 \text{ mit } t_0 = 2 \text{ und } t_1 = 3$$

2 Homogene Rekursionsgleichungen

Homogene Rekursionsgleichungen stellen den einfachsten Fall dar, auf den man stossen kann. Sie lassen sich mit Hilfe des *charakteristischen Polynoms* leicht lösen.

Definition 2.1 (Homogene Rekursionsgleichung)**.** Eine homogene Rekursionsgleichung ist eine Rekursionsgleichung der Form

$$\sum_{i=0}^{k} a_i \cdot t_{n-k+i} = a_k t_n + a_{k-1} t_{n-1} + \ldots + a_1 t_{n-k+1} + a_0 t_{n-k} = 0 \tag{2.1}$$

Definition 2.2 (Charakteristisches Polynom)**.** Das charakteristische Polynom der homogenen Rekursionsgleichung 2.1 ist definiert als

$$P(x) = a_k x^k + a_{k-1} x^{k-1} + \ldots + a_1 x^1 + a_0 x^0 = \sum_{\lambda=0}^{k} a_\lambda \cdot a^\lambda \tag{2.2}$$

Häufig wird $P(x)$ auch mit $\chi(x)$ bezeichnet.

Auf Grundlage der Nullstellen dieses Polynoms kann man nun die Lösung bestimmen. Das Verfahren wird dazu im Folgenden beschrieben.

2.1 Lösen homogener Rekursionsgleichungen

Gegeben sei die homogene Rekursionsgleichung 2.1 mit den k *Anfangsbedingungen* $t_0 \ldots t_{k-1}$

1. Bestimme die l *Nullstellen* $x_0 \ldots x_{l-1}$ mit den *Häufigkeiten* $m_0 \ldots m_{l-1}$ des charakteristischen Polynoms.

2. Die *allgemeine Lösung* wird bestimmt als

$$t_n = \sum_{i=0}^{k-1} c_i \cdot y_i \text{ mit } \{y_0, \ldots, y_{k-1}\} = \bigcup_{i=0}^{l-1} \bigcup_{j=0}^{m_i-1} \left\{ \binom{n}{j} \cdot j! \cdot x_i^n \right\} \tag{2.3}$$

   und Konstanten $c_i (i \in \{0, \ldots, k-1\})$

3. Bestimme die $c_i (i \in \{0, \ldots, k-1\})$ und damit die *exakte Lösung* durch Einsetzen der Anfangsbedingungen $t_0 \ldots t_{k-1}$

2.1.1 Beispiel: Fibonaccizahlen, Lösen der homogenen Rekursionsgleichung

Beispiel 2.1. Betrachte die homogene Form der in Beispiel 1.1 hervorgebrachten Rekursionsgleichung

$$t_n - t_{n-1} - t_{n-2} = 0$$

mit den Anfangswerten $t_0 = 2$ und $t_1 = 3$
Das charakteristische Polynom ist daher

$$P(x) = x^2 - x - 1 = 0$$

Dieses hat die Nullstellen

$$x_{0,1} = \frac{1 \pm \sqrt{5}}{2}$$

woraus sich die allgemeine Lösung ergibt zu

$$\begin{gathered} t_n = c_0 \cdot y_0 + c_1 \cdot y_1 \\ \text{mit} \\ y_0 = \binom{n}{0} \cdot 0! \cdot x_0^n = 1 \cdot 1 \cdot x_0^n \\ \text{und} \\ y_1 = \binom{n}{0} \cdot 0! \cdot x_1^n = 1 \cdot 1 \cdot x_1^n \end{gathered}$$

Also ist die *allgemeine Lösung*:

$$t_n = c_0 \cdot x_0^n + c_1 \cdot x_1^n$$

Durch Einsetzen der Anfangsbedingungen bekommt man das lineare Gleichungssystem:

$$\begin{array}{ccccc} c_0 \cdot x_0^0 & + & c_1 \cdot x_1^0 & = & 2 \\ c_0 \cdot x_0^1 & + & c_1 \cdot x_1^1 & = & 3 \end{array}$$

bzw.

$$\begin{array}{ccccc} c_0 & + & c_1 & = & 2 \\ c_0 \cdot x_0 & + & c_1 \cdot x_1 & = & 3 \end{array}$$

und nach Lösen des linearen Gleichungssystems erhält man

$$c_0 = 2 - \frac{3 - 2 \cdot x_0}{x_1 - x_0} \text{ und } c_1 = \frac{3 - 2 \cdot x_0}{x_1 - x_0}$$

Mit den eingesetzten Werten der beiden Nullstellen x_0, x_1 ergibt sich nach einigen Äquivalenzumformungen:

$$c_0 = \frac{5 + 2\sqrt{5}}{2} \text{ und } c_1 = \frac{5 - 2\sqrt{5}}{2}$$

Damit ist die homogene Rekursionsgleichung gelöst:

$$t_n = \frac{5 + 2\sqrt{5}}{2} \cdot \left(\frac{1 + \sqrt{5}}{2}\right)^n + \frac{5 - 2\sqrt{5}}{2} \cdot \left(\frac{1 - \sqrt{5}}{2}\right)^n$$

2.1.2 Vereinfachung des Rechenweges

Bemerkung 2.1 (zur Vereinfachung)**.** Da die Konstanten c_i anfangs noch nicht bestimmt sind, ist es möglich die *allgemeine Lösung* **zu vereinfachen**.

Beispiel 2.2. $(x - 5)^3$ sei ein Faktor des charakteristischen Polynoms.
Merke: Dieser ist dreifache Nullstelle!
Daraus bekommt man in der *allgemeinen Lösung*:

$$\begin{array}{rl} & c_0 \cdot \binom{n}{0} \cdot 0! \cdot 5^n + c_1 \cdot \binom{n}{1} \cdot 1! \cdot 5^n + c_2 \cdot \binom{n}{2} \cdot 2! \cdot 5^n \\ = & c_0 \cdot 5^n + c_1 \cdot n \cdot 5^n + c_2 \cdot \frac{n(n-1)}{2!} \cdot 2! \cdot 5^n \\ = & c_0 \cdot 5^n + c_1 \cdot n \cdot 5^n + c_2 \cdot n(n-1) \cdot 5^n \\ = & c_0 \cdot 5^n + c_1 \cdot n \cdot 5^n + c_2 \cdot n^2 \cdot 5^n - c_2 \cdot n \cdot 5^n \\ = & c_0 \cdot 5^n + (c_1 \cdot n \cdot 5^n - c_2 \cdot n \cdot 5^n) + c_2 \cdot n^2 \cdot 5^n \\ = & c_0 \cdot 5^n + (c_1 - c_2) \cdot n \cdot 5^n + c_2 \cdot n^2 \cdot 5^n \\ = & c_0 \cdot 5^n + c_1' \cdot n \cdot 5^n + c_2' \cdot n^2 \cdot 5^n \end{array}$$

Wie man sieht, wurde der $c_2 \cdot n$ Anteil mit $c_1 \cdot n$ zu $\acute{c_1} \cdot n$ zusammen gefügt. Dafür musste c_2 in $\acute{c_2}$ abgeändert werden.
Es ist daher sinnvoll, beim Aufstellen der allgemeinen Lösung, direkt auf die vereinfachte Form zu schliessen.

$$t_n = \sum_{i=0}^{k-1} c_i \cdot y_i \text{ mit } \{y_0, \ldots, y_{k-1}\} = \bigcup_{i=0}^{l-1} \bigcup_{j=0}^{m_i-1} \{n^j \cdot x_i^n\} \tag{2.4}$$

Jede der l Nullstellen x_i ($i \in \{0, \ldots, l-1\}$, $l \in \mathbb{N}$) des charakteristischen Polynoms liefert zu t_n einen additiven Beitrag in Abhängigkeit von seiner Häufigkeit m_i.
Dieser Beitrag heiße $B(x_i)$, kurz $B(i)$.
$B(i)$ ist von der Form:

$$B(i) = \sum_{j=0}^{m_i-1} c_{j_i} \cdot n^j \cdot x_i^n \tag{2.5}$$

Damit lässt sich t_n auch ausdrücken als:

$$\begin{aligned} t_n &= \sum_{i=0}^{l-1} B(i) \\ &= \sum_{i=0}^{l-1} B(x_i) \\ &= \sum_{i=0}^{l-1} \sum_{j=0}^{m_i-1} c_{j_i} \cdot n^j \cdot x_i^n \end{aligned} \tag{2.6}$$

3 Inhomogene Rekursionsgleichungen

Wenn man mit Rekursionsgleichungen arbeitet, dann wird man es häufig mit *inhomogenen Rekursionsgleichungen* machen müssen.

Definition 3.1 (Inhomogene Rekursionsgleichung)**.** Wenn Rekursionsgleichungen eine Störfunktion ungleich der Nullfunktion aufweisen, so nennt man sie inhomogen.

Einschränkung: In diesem Abschnitt beschränken wir uns auf Rekursionsgleichungen in Standardform mit einer polynomiellen Störfunktion $s(n) \in \mathbb{P}$

$$\sum_{i=0}^{k} a_i \cdot t_{n-k+i} = s(n) \text{ mit } s(n) = \sum_{j=0}^{h} (b_j^n \cdot p_j(n)) \tag{3.1}$$

Mit Konstanten b_j und Polynomen p_j mit dem Grad d_j.
Diese sind jedoch für die meisten Komplexitätsberechnungen ausreichend.

3.1 Lösen inhomogener Rekursionsgleichungen

Gegeben sei die inhomogene Rekursionsgleichung 3.1 mit den k mit den k *Anfangsbedingungen* $t_0 \dots t_{k-1}$

1. Bestimme das charakteristische Polynom $P(x)$ für die homogene Rekursionsgleichung.

2. Erweitere dieses zu
$$P'(x) = P(x) \cdot \prod_{j=0}^{h} (x - b_j)^{d_j+1} \tag{3.2}$$
$P'(x)$ wird auch *charaktieristisches Polynom der inhomogenen Rekursionsgleichung* genannt.

3. Verfahre zum Finden der allgemeinen Lösung mit dem erweiterten Polynom wie im Verfahren zur Lösung homogener Rekursionsgleichungen.

4. Die Konstanten c_i können durch zusätzliche Anfangsbedingungen berechnet werden. *Sind diese nicht vorhanden*, so kann, zur Bestimmung der zusätzlichen c_i, die *allgemeine Lösung in die Rekursionsgleichung* eingesetzt werden. Durch *Koeffizientenvergleich* ist es dann möglich diese zu berechnen. Die restlichen Konstanten bekommt man durch Einsetzen der k Anfangsbedingungen $t_0 \dots t_{k-1}$

3.1.1 Beispiel: Fibonaccizahlen, Lösen der inhomogenen Rekursionsgleichung

Beispiel 3.1. Betrachte die Rekursionsgleichung aus Beispiel 1.1 hervorgebrachten Rekursionsgleichung.

$$t_n - t_{n-1} - t_{n-2} = 3$$

mit den Anfangsbedingungen $t_0 = 2$ und $t_1 = 3$. Die rechte Seite ist darstellbar als $b^n \cdot p(n)$ mit $b = 1$ und $p(n) = 3 \cdot n^0$. $p(n)$ hat den Grad 0. Mit x_0 und x_1 und dem charakteristischen Polynom $P(x)$ aus dem Beispiel 2.1 entsteht dadurch

$$P'(x) = P(x) \cdot (x-1)^{0+1} = (x - x_0) \cdot (x - x_1) \cdot (x - 1)$$

als charakteristisches Polynom dieser inhomogenen Rekursionsgleichung.
Dieses hat daher die Nullstellen

$$x_{0,1} = \frac{1 \pm \sqrt{5}}{2} \text{ sowie } x_2 = 1$$

Die *allgemeine Lösung* ergibt sich dann zu

$$t_n = c_0 \cdot x_0^n + c_1 \cdot x_1^n + c_2 \cdot 1^n$$

Es stehen keine zusätzlichen Anfangsbedingungen zur Verfügung, weshalb c_2 durch *Einsetzen der allgemeinen Lösung in die Rekursionsgleichung*

$$t_n - t_{n-1} - t_{n-2} = 3$$

bestimmt wird, also:

$$\begin{array}{rcl}
(c_0x_0^n + c_1x_1^n + c_2) - (c_0x_0^{n-1} + c_1x_1^{n-1} + c_2) - (c_0x_0^{n-2} + c_1x_1^{n-2} + c_2) & = & 3 \\
c_0(x_0^n - x_0^{n-1} - x_0^{n-2}) + c_1(x_1^n - x_1^{n-1} - x_1^{n-2}) - c_2 & = & 3 \\
c_0x_0^{n-2}\underbrace{(x_0^2 - x_0 - 1)}_{=0} + c_1x_1^{n-2}\underbrace{(x_1^2 - x_1 - 1)}_{=0} - c_2 & = & 3 \\
c_2 & = & 3
\end{array}$$

c_0 und c_1 werden wiederrum durch die Anfangsbedingungen gewonnen:

$$\begin{array}{rcrcl}
c_0 \cdot x_0^0 & + & c_1 \cdot x_1^0 - 3 & = & 2 \\
c_0 \cdot x_0^1 & + & c_1 \cdot x_1^1 - 3 & = & 3
\end{array}$$

bzw.

$$\begin{array}{rcrcl}
c_0 & + & c_1 & = & 5 \\
c_0 \cdot x_0 & + & c_1 \cdot x_1 & = & 6
\end{array}$$

und damit (unter Berücksichtigung der Nullstellen x_i)

$$c_0 = \frac{25 + 7\sqrt{5}}{10} \text{ und } c_1 = \frac{5 - 5\sqrt{5} - 7}{2\sqrt{5}}$$

Die *exakte Lösung* ist daher

$$t_n = \frac{25 + 7\sqrt{5}}{10} \cdot \left(\frac{1 + \sqrt{5}}{2}\right)^n - 3$$

Der in Beispiel 1.1 angegebene Algorithmus hat also die Komplexität $O((\frac{1+\sqrt{5}}{2})^n)$ und ist somit exponentiell.

3.1.2 Weiteres Beispiel

Beispiel 3.2. Betrachte die Rekursionsgleichung

$$t_n - 2t_{n-1} = (n+5) \cdot 3^n + 1$$

mit der *Anfangsbedingung* $t_0 = 12$. Das charakteristische Polynom ist $P(x) = x - 2 = 0$. Die rechte Seite hat die Form $p_0(n) \cdot b_0^n + p_1(n) \cdot b_1^n$ mit $p_0(n) = n + 5$ (*Grad* $d_0 = 1$), $b_0 = 3$, $p_1(n) = 1$ (*Grad* $d_1 = 0$) und $b_1 = 1$. Man erhält also insgesamt das Polynom

$$P'(x) = (x-2) \cdot (x-3)^{(1+1)} \cdot (x-1)^{(0+1)} = (x-1) \cdot (x-2) \cdot (x-3)^2$$

Die Nullstellen sind: $x_0 = 1, x_1 = 2, c_3 = 3$
Nun wird die *allgemeine Lösung* nach Bemerkung 2.1 gebildet.

$$\begin{aligned}
B(0) &= (c_{0_0} \cdot n^0 \cdot 1^n) = (c_0) \\
B(1) &= (c_{0_1} \cdot n^0 \cdot 2^n) = (c_1 \cdot 2^n) \\
B(2) &= (c_{0_2} \cdot n^0 \cdot 3^n) + (c_{1_2} \cdot n^1 \cdot 3^n) = (c_2 \cdot 3^n) + (c_3 \cdot n \cdot 3^n)
\end{aligned}$$

$$t_n = (c_0) + (c_1 \cdot 2^n) + (c_2 \cdot 3^n) + (c_3 \cdot n \cdot 3^n)$$

Nun wird die *allgemeine Lösung* in die Rekursionsgleichung

$$t_n - 2t_{n-1} = (n+5) \cdot 3^n + 1$$

eingesetzt.
Dies liefert:

$$\begin{aligned}
& (c_0 + c_1 2^n + c_2 3^n + c_3 n 3^n) - 2(c_0 + c_1 2^{n-1} + c_2 3^{n-1} + c_3(n-1)3^{n-1}) \\
\stackrel{!}{=} \; & (n+5) \cdot 3^n + 1 \\
\Leftrightarrow \quad & c_0(1-2) + c_1 2^n \underbrace{\left(1 - 2 \cdot \frac{1}{2}\right)}_{=0} + c_2 3^n \left(n - \tfrac{2}{3} \cdot n + \tfrac{2}{3}\right) \\
\stackrel{!}{=} \; & (n+5) \cdot 3^n + 1
\end{aligned}$$

$$\begin{aligned}
\Leftrightarrow & \quad -c_0 + \tfrac{1}{3}c_2 3^n + c_3 3^n \left(\tfrac{1}{3}n + \tfrac{2}{3}\right) \\
& \overset{!}{=} (n+5) \cdot 3^n + 1 \\
\Leftrightarrow & \quad -c_0 + \tfrac{1}{3} \cdot c_2 \cdot 3^n + \tfrac{2}{3} \cdot c_3 \cdot 3^n + \tfrac{1}{3} \cdot c_3 \cdot n \cdot 3^n \\
& \overset{!}{=} (n+5) \cdot 3^n + 1 \\
\Leftrightarrow & \quad \tfrac{1}{3} \cdot c_3 \cdot n \cdot 3^n + \left(\tfrac{1}{3} \cdot c_2 + \tfrac{2}{3} \cdot c_3\right) \cdot 3^n - c_0 \\
& \overset{!}{=} n \cdot 3^n + 5 \cdot 3^n + 1
\end{aligned}$$

Durch *Koeffizientenvergleich* bekommt man $-c_0 = 1$, $\frac{1}{3}c_3 = 1$ und $\frac{1}{3}c_2 + \frac{2}{3}c_3 = 5$ und somit $c_0 = -1$, $c_2 = 9$ und $c_3 = 3$.
Durch die *Anfangsbedingung* $t_0 = 12$ kann man die letzte Unbekannte, c_1, berechnen.

$$t_0 = (-1 \cdot 1^0) + (c_1 \cdot 2^0) + (9 \cdot 3^0) + (3 \cdot 0 \cdot 3^0) = c_1 + 8 = 12$$
$$\Rightarrow c_1 = 4$$

Die *Lösung der Rekursionsgleichung* ist also

$$t_n = -1 + 4 \cdot 2^n + 9 \cdot 3^n + 3 \cdot n \cdot 3^n$$

4 Variablentransformation

4.1 Gebrochene Rekursionsvariable

Nicht alle Algorithmen lassen sich direkt in eine Rekursionsgleichung der obigen Form, der *Standardform*, umwandeln. Dann müssen noch zusätzliche Transformationen durchgeführt werden. In den meisten Fällen kommt n dabei in gebrochener Form vor.
Zum Beispiel:

$$t(n) = t\left(\frac{n}{m}\right) + n$$

Um dies in die oben besprochene Form zu bringen, muss n mit $n = m^k$ ersetzt werden. Man beschränkt sich dabei für n auf Potenzen von m. In diesem Fall hätte man also

$$t(m^k) = t(m^{k-1}) + m^k$$

und somit

$$t_k = t_{k-1} + m^k$$

Nach dem Auflösen dieser Rekursionsgleichung, muss man dann nur noch mit $n = log_m(k)$ *zurücksubstituieren.*

4.1.1 Beispiel: Mergesort, Behandlung von gebrochener Rekursionsvariable

Beispiel 4.1 (zur Variablentransformation)**.** Betrachte den Mergesort-Algorithmus:

Algorithmus 2.

```
sort(s[1..n])
{
     b1[1..n/2] = sort(a[1..n/2]);
     b2[1..n/2] = sort(a[(n/2)+1..n]);
   return merge(b1[1..n/2],b2[1..n/2]);
 }
```

Der Algorithmus spaltet eine Sequenz der Länge n in zwei Teilsequenzen der Länge $\frac{n}{2}$ auf. Diese werden rekursiv weiterbehandelt und sortiert zurückgegeben. Dann werden die beiden sortierten Sequenzen in linearer Zeit zusammengefügt (**merge**). Als Laufzeitkomplexität bekommt man also

$$t(n) = 2 \cdot t\left(\frac{n}{2}\right) + n$$

Wenn man n auf Zweierpotenzen einschränkt, kommt man mit der *Substitution*

$$n = 2^k$$

auf die Gleichung

$$t(2^k) = 2 \cdot t(2^{k-1}) + 2^k$$

dies entspricht der Rekursionsgleichung

$$t_k = 2 \cdot t_{k-1} + 2^k$$

Diese Rekursionsgleichung besitzt das *charaktieristische Polynom*

$$P(x) = (x-2)^2$$

und daher die *allgemeine Lösung*

$$t_k = c_0 \cdot 2^k + c_1 \cdot k \cdot 2^k$$

Nun muss die Substition rückgängig gemacht werden. Durch *Zurücksubstition* erhält man nun:

$$\begin{array}{rll} & k & = \log_2(n) \\ \Rightarrow & t_n & = c_0 \cdot 2^{\log_2(n)} + c_1 \cdot log_2(n) \cdot 2^{\log_2(n)} \\ \Leftrightarrow & t_n & = c_0 \cdot n + c_1 \cdot \log_2(n) \cdot n \end{array}$$

und somit gilt $t_n \in O(n \cdot log(n))$

Bemerkung 4.1. Eine Variablentransformation wie im letzten Beispiel bezieht sich immer auf den Definitionsbereich einer Komplexitätsabbildung $f : \mathbb{N} \to \mathbb{R}^+$. Manchmal ist es jedoch auch sinnvoll, eine Transformation im Wertebereich in Erwägung zu ziehen. Im folgenden Beispiel wird gezeigt, dass das bisherige Verfahren manchmal nicht ausreichend ist.

4.2 Nichtlineare Rekursion

Bisher haben wir uns mit *linearen* Rekursionsgleichungen beschäftigt, es gibt aber auch *nichtlineare* Rekursionsgleichungen.

Definition 4.1. Nichtlineare Rekursionsgleichungen sind Rekursionsgleichungen, der Form:

$$t_n = \sum_{i=0}^{k} a_i \cdot (t_{n-k+i})^j = s(n) \text{ mit } j \in \mathbb{N} \tag{4.1}$$

Das nächste Beispiel zeigt, dass man auch nichtlineare Rekursionsgleichungen, geeignete Transformatin vorausgesetzt, mit dem oben beschriebenen Verfahren lösen kann.

4.2.1 Einfaches Beispiel

Beispiel 4.2 (für weitere Transformationen)**.** Betrachte $t(n) = n \cdot (t(\frac{n}{2}))^2$ mit $n \in \mathbb{N}$ und den Anfangsbedingungen $t(1) = 6$. Man beschränke sich hier wieder auf Zweierpotenzen und führe wie im letzten Beispiel eine Variablentransformation $n = 2^k$ durch. Man erhält

$$t_k = 2^k \cdot t_{k-1}^2$$

Diese Gleichung jedoch ist nicht mehr linear, auch die Koeffizienten sind nicht konstant. Sie passt also nicht in die oben betrachtete Form, weshalb noch weitere Transformationen durchgeführt werden müssen.
In diesem Falle $u_k = log_2(t_k)$. Dann ergibt sich

$$u_k = log_2(2^k \cdot t_{k-1}^2) = \log_2(t_{k-1}^2) = k + 2 \cdot u_k$$

mit der Anfangsbedingung $u_0 = \log_2(t_0) = \log_2(t(2^0)) = \log_2(t(1)) = \log_2(6) = 1 + \log_2 3$. Diese Gleichung kann wie gewohnt gelöst werden.

Literaturverzeichnis

[Brü03] Brünning, Dipl.-Inf. Andreas: *Rekursionsgleichungen - Material zur Vorlesung „Algorithmentheorie", WS 2003/04, bei Prof. Kupka*, 2003 2

[Har05] Hartmann, Robert: *Vorlesungsskript „Fibonaccizahlen - Schönheit und Nutzen in Informatik und Mathematik", SS 2005, Prof. Lex*, 2005 2

[Pri02] Pristerjahn, Steffen: *Rekursionsgleichungen - Material zur Vorlesung „Praktische und angewandte Informatik 1", WS 2002/03, bei Prof. Ecker*, 2002 2

[Sta01] Stamm, Dipl.-Inf. Frank: *Rekursionsgleichungen - Material zur 1.Übung der Vorlesung „Praktische und angewandte Informatik 1", WS 2001/02, bei Prof. Ecker*, Oktober 2001 2

[Wun01] Wunsch, Marko: *Vorlesungsskript „Kombinatorik", SS 2001, Prof. Klotz*, 2001 2