

Diskrete Mathematik

Anna-Lena Rädler
Christina Kohl
Georg Moser
Christian Sternagel
Vincent van Oostrom

Institut für Informatik @ UIBK
Sommersemester 2018



Übersicht

Inhalte der Lehrveranstaltung

Beweismethoden

deduktive Beweise, Beweise von Mengeninklusionen, Kontraposition, Widerspruchsbeweise, vollständige Induktion, wohlfundierte Induktion, strukturelle Induktion, Gegenbeispiele

Relationen, Ordnungen und Funktionen

Äquivalenzrelationen, partielle Ordnungen, Wörter, asymptotisches Wachstum von Funktionen

Graphentheorie

gerichtete Graphen, ungerichtete Graphen

Zähl- und Zahlentheorie

Aufzählen und Nummerieren von Objekten, Abzählbarkeit, **Lösen von Rekursionsformeln**, **Mastertheorem**, Rechnen mit ganzen Zahlen, euklidischer Algorithmus, Primzahlen, Restklassen

Zusammenfassung der letzten LVA

Definition

- Eine Menge M heißt **endlich**, wenn es eine natürliche Zahl m und eine bijektive Abbildung $\alpha: \{0, 1, \dots, m-1\} \rightarrow M$ gibt
- In diesem Fall ist m eindeutig bestimmt und man nennt $\#(M) := m$ die **Anzahl** der Elemente von M

Definition

Eine Menge M heißt **abzählbar unendlich**, wenn eine bijektive Abbildung $\alpha: \mathbb{N} \rightarrow M$, $i \mapsto x_i$ existiert. Man schreibt dann $M = \{x_0, x_1, x_2, \dots\}$ nennt α eine **Aufzählung** von M und α^{-1} eine **Nummerierung** von M .

Beispiel

Sei Σ ein endliches Alphabet. Dann ist das Wortmonoid $\Sigma^* := \bigcup_{n \geq 0} \Sigma^n$ abzählbar. Andererseits sind die reellen Zahlen nicht abzählbar.

Lösen von Rekurrenzgleichungen

Definition

- Eine rekursiv definierte Funktion $f: \mathbb{N} \rightarrow \mathbb{N}$ nennt man **Rekurrenz** oder **Rekurrenzrelation**
- Rekurrenzen schreibt man oft in Indexschreibweise: f_n statt $f(n)$

Beispiel

Betrachte die folgende Funktion $f: \mathbb{N} \rightarrow \mathbb{N}$, definiert für $n \geq 1$:

$$f(n) = \begin{cases} 1 & n = 1 \\ 2f(\frac{n}{2}) + n & n \geq 2 \end{cases}$$

Beispiel

Die **Fibonacci-Zahlen** sind rekursiv definiert durch

$$f_0 = 0 \quad f_1 = 1 \quad f_n = f_{n-1} + f_{n-2}$$

Lemma

Sei $T: \mathbb{N} \rightarrow \mathbb{N}$ gegeben durch

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$a, b \in \mathbb{N}$ mit $b > 1$; $\exists k$ mit $n = b^k$. Dann gilt

$$T(n) = a^k T(1) + \sum_{i=0}^{k-1} a^i f\left(\frac{n}{b^i}\right) \quad (1)$$

Beweis.

Durch iteriertes Einsetzen der Rekursionsformel sehen wir für alle $\ell \geq 1$:
 $a^{\ell-1} T\left(\frac{n}{b^{\ell-1}}\right) = a^{\ell-1} T\left(\frac{n}{b^{\ell}}\right) + a^{\ell-1} f\left(\frac{n}{b^{\ell-1}}\right)$ und somit

$$T(n) = a^k T(1) + a^{k-1} f\left(\frac{n}{b^{k-1}}\right) + \dots + a f\left(\frac{n}{b}\right) + f(n)$$

Lösen von Rekurrenzgleichungen

Methode des Einsetzens

Beispiel (cont'd)

Sei $f: \mathbb{N} \rightarrow \mathbb{N}$ wie folgt definiert:

$$f(n) = \begin{cases} 1 & n = 1 \\ 2f\left(\frac{n}{2}\right) + n & n = 2^k, k \geq 1 \end{cases}$$

Dann ist $a = b = 2$ und Einsetzen in die Formel (??) liefert

$$f(n) = 2^k + \sum_{i=0}^{k-1} 2^i \frac{n}{2^i} = 2^k + n \sum_{i=0}^{k-1} 1^i = 2^k + nk = n + n \log_2 n$$

Frage

Wie erhält man eine geschlossene Form für die Elemente der Fibonaccifolge?

Lösen von Rekurrenzgleichungen

Methode der erzeugenden Funktionen

Definition

- Für eine Folge $f: \mathbb{N} \rightarrow \mathbb{R}$ heißt die Potenzreihe

$$F(x) := \sum_{n=0}^{\infty} f(n) \cdot x^n$$

die **erzeugende Funktion** von f

- Die Methode der erzeugenden Funktionen versucht, aus den Rekursionsformeln für $f(n)$ Gleichungen für $F(x)$ herzuleiten und diese mit algebraischen oder analytischen Mitteln zu lösen

Beispiel (Geometrische Reihe)

$$\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}$$

Beispiel

Betrachte die folgende Funktion $a: \mathbb{N} \rightarrow \mathbb{N}$:

$$a_0 = 0 \quad a_1 = 1 \quad a_n = 5a_{n-1} - 6a_{n-2} \quad n \geq 2$$

Schritt 1: Darstellung als Erzeugende Funktion

Einsetzen der Folge $(a_n)_{n \geq 0}$ in die erzeugende Funktion $A(x)$:

$$\sum_{n=2}^{\infty} a_n x^n = \sum_{n=2}^{\infty} (5a_{n-1} - 6a_{n-2}) x^n = 5 \sum_{n=2}^{\infty} a_{n-1} x^n - 6 \sum_{n=2}^{\infty} a_{n-2} x^n$$

Die Summen der rechte Seite können wie folgt geschrieben werden:

$$\begin{aligned} \sum_{n=2}^{\infty} a_{n-1} x^n &= \sum_{n=1}^{\infty} a_n x^{n+1} = x \sum_{n=1}^{\infty} a_n x^n = x(A(x) - a_0) = xA(x) \\ \sum_{n=2}^{\infty} a_{n-2} x^n &= \sum_{n=0}^{\infty} a_n x^{n+2} = x^2 \sum_{n=0}^{\infty} a_n x^n = x^2 A(x) \end{aligned}$$

Schritt 1: Fortsetzung

Die linke Seite kann wie folgt geschrieben werden:

$$\sum_{n=2}^{\infty} a_n x^n = A(x) - a_0 - a_1 x = A(x) - x$$

In Summe erhalten wir $A(x) - x = 5xA(x) - 6x^2A(x)$

Schritt 2: Auflösen nach $A(x)$

Aus Schritt 1 folgt, dass gilt $A(x) - x = 5xA(x) - 6x^2A(x)$, also

$$\begin{aligned} A(x) &= \frac{x}{1 - 5x + 6x^2} \\ &= \frac{1}{2x - 1} - \frac{1}{3x - 1} = -\frac{1}{1 - 2x} + \frac{1}{1 - 3x} \\ &= -\sum_{n=0}^{\infty} (2x)^n + \sum_{n=0}^{\infty} (3x)^n = \sum_{n=0}^{\infty} (-2^n + 3^n)x^n \end{aligned}$$

Schritt 3: Koeffizientenvergleich

Aus Schritt 2 erhalten wir $A(x) = \sum_{n=0}^{\infty} (-2^n + 3^n)x^n$; Koeffizientenvergleich mit der erzeugenden Funktion ergibt eine geschlossene Form für a :

$$\begin{aligned} \sum_{n=0}^{\infty} a_n x^n &= \sum_{n=0}^{\infty} (-2^n + 3^n)x^n \\ a_n &= 3^n - 2^n \end{aligned}$$

Schritt 4: Kontrolle

Schließlich kontrollieren wir, ob die Methode das richtige Resultat erhalten hat:

$$\begin{aligned} a_0 &= 3^0 - 2^0 = 0 \\ a_1 &= 3^1 - 2^1 = 1 \\ a_n &= 3^n - 2^n = \\ &= 5(3^{n-1} - 2^{n-1}) - 6(3^{n-2} - 2^{n-2}) \\ &= 5a_{n-1} - 6a_{n-2} \end{aligned}$$

Beispiel (Fibonaccifolge, cont'd)

Für die erzeugende Funktion $F(x)$ der Fibonaccifolge erhalten wir

$$\begin{aligned} F(x) &= \sum_{n=0}^{\infty} f_n x^n = f_0 + f_1 x + \sum_{n=2}^{\infty} (f_{n-1} + f_{n-2}) x^n \\ &= x + x \cdot F(x) + x^2 \cdot F(x) \end{aligned}$$

und durch Partialbruchzerlegung

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

(NB: Die Lösungsformel für $ax^2 + bx + c = 0$ lautet

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

folglich gilt: $1 - x - x^2 = (-x - \frac{1+\sqrt{5}}{2}) \cdot (x + \frac{1-\sqrt{5}}{2})$

Beispiel (cont'd)

Einsetzen der geometrischen Reihe $\sum_{n \geq 0} x^n = \frac{1}{1-x}$, dh. lösen nach $F(x)$ liefert

$$F(x) = \frac{1}{\sqrt{5}} \left[\sum_{n=0}^{\infty} \left(\frac{1+\sqrt{5}}{2} \right)^n x^n - \sum_{n=0}^{\infty} \left(\frac{1-\sqrt{5}}{2} \right)^n x^n \right]$$

Mit dem abschließenden Koeffizientenvergleich erhalten wir

$$f_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right]$$

Definition (Divide-and-Conquer-Algorithmen)

- Algorithmus löst Instanzen bis zur Größe m direkt
- Instanzen der Größe $n > m$ hingegen zerlegt der Algorithmus in a Teilinstanzen mit Größen $\lfloor n/b \rfloor$ und $\lceil n/b \rceil$; löst diese rekursiv und setzt Teillösungen zusammen

Definition

- Zeit zum Aufteilen und Zusammenfügen betrage $f(n)$
- Gesamtzeit sei $T(n)$, wobei wir annehmen $T(n+1) \geq T(n)$
- Wir definieren

$$T^-(n) := \begin{cases} a \cdot T^-(\lfloor n/b \rfloor) + f(n) & \text{falls } n > m \\ T(n) & \text{falls } n \leq m \end{cases}$$

$$T^+(n) := \begin{cases} a \cdot T^+(\lceil n/b \rceil) + f(n) & \text{falls } n > m \\ T(n) & \text{falls } n \leq m \end{cases}$$

Beispiel (Merge Sort)

Betrachte Merge Sort:

```
merge :: Ord a => [a] -> [a] -> [a]
merge xs [] = xs
merge [] ys = ys
merge (x:xs) (y:ys)
  | (x <= y) = x:(merge xs (y:ys))
  | otherwise = y:(merge (x:xs) ys)
```

```
mergesort :: Ord a => [a] -> [a]
mergesort [] = []
mergesort [x] = [x]
mergesort xs = merge (mergesort (fsthalf xs))
                     (mergesort (sndhalf xs))
```

Frage

Kann die Komplexität von Merge-Sort abgeschätzt werden?

Definition (Wiederholung)

- Algorithmus löst Instanzen bis zur Größe m direkt
- Instanzen der Größe $n > m$ hingegen zerlegt der Algorithmus in a Teilinstanzen mit Größen $\lfloor n/b \rfloor$ und $\lceil n/b \rceil$; löst diese rekursiv und setzt Teillösungen zusammen

Beobachtung

- Sei $n = m \cdot b^k$
- Algorithmus teilt k -Mal, sodass für $r := \log_b a$ gilt: es gibt

$$a^k = (b^r)^k = (b^k)^r = \left(\frac{n}{m}\right)^r,$$

Basisinstanzen

- Lösung der Basisinstanzen alleine kostet $\Theta(n^r)$ an Aufwand

Beobachtung

- $a \cdot T(\lfloor n/b \rfloor) + f(n) \leq T(n) \leq a \cdot T(\lceil n/b \rceil) + f(n)$
- Berücksichtigung des Aufwandes für das Aufteilen und Zusammensetzen, erlaubt die asymptotische Analyse von $T^\pm(n)$

Satz (Master-Theorem)

Sei $T(n)$ eine wachsende Funktion, die folgende Rekurrenzgleichungen erfüllt

$$T(n) = \begin{cases} c & n = 1 \\ aT(\frac{n}{b}) + f(n) & n = b^k, k = 1, 2, \dots \end{cases}$$

wobei $a \geq 1$, $b \geq 2$, $c > 0$. Wenn $f \in \Theta(n^s)$, wobei $s \geq 0$, dann gilt

$$T(n) \in \begin{cases} \Theta(n^{\log_b a}) & \text{wenn } a > b^s \\ \Theta(n^s \log n) & \text{wenn } a = b^s \\ \Theta(n^s) & \text{wenn } a < b^s \end{cases}$$

Beispiel (Merge-Sort, cont'd)

Wegen $a = b = 2$ und $f \in \Theta(n)$ gibt das Master-Theorem die Laufzeitabschätzung $T(n) \in \Theta(n \cdot \log n)$, da gilt $r := \log_b a = 1$ und somit $f(n) \in \Theta(n^r)$ (zweiter Fall)

Beispiel

Betrachte die folgende Rekurrenz:

$$T(n) = 4T\left(\frac{n}{2}\right) + n^1$$

dann gilt $a = 4$, $b = 2$, $r = \log_b a = 2$ und $a > b^r$, also gilt mit dem ersten Fall des Satzes: $T(n) \in \Theta(n^2)$

Beweis des Mastertheorems

Fall $f \in \Theta(n^s)$ mit $a = b^s$

- setze $r = \log_b a$
- dann gilt also $r = s$
- wir verwenden Eigenschaften von Θ , bzw. Eigenschaften der Exponentialfunktion um zu schließen:

$$a^i f\left(\frac{n}{b^i}\right) = \Theta\left(a^i \frac{n^r}{(b^i)^r}\right) = \Theta\left(a^i \frac{n^r}{(br)^i}\right) = \Theta\left(a^i \frac{n^r}{a^i}\right) = \Theta(n^r)$$

- somit erhalten wir:

$$\sum_{i=0}^k a^i f\left(\frac{n}{b^i}\right) = \Theta\left(\sum_{i=0}^k n^r\right) = \Theta(kn^r) = \Theta(n^r \log_2 n)$$

- außerdem wissen wir bereits, dass

$$a^k T(1) \in \Theta(n^r)$$

Beweis (Fortsetzung)

- wir erinnern uns an die Formel (??)

$$T(n) = a^k T(1) + \sum_{i=0}^{k-1} a^i f\left(\frac{n}{b^i}\right)$$

- die Terme habe die folgenden beiden Abschätzungen:

$$a^k T(1) \in \Theta(n^r)$$

$$\sum_{i=0}^{k-1} a^i f\left(\frac{n}{b^i}\right) \in \Theta(n^r \log_2 n)$$

- somit erhalten wir

$$T(n) \in \Theta(n^r \log_2 n)$$