Übung Algorithmen I
29.4.15

Christoph Striecks
Christoph.Striecks@kit.edu

Christian Staudt
Christian.Staudt@kit.edu

(Mit Folien von Julian Arz, Timo Bingmann und Sebastian Schlag.)

Roadmap

- Kurze Wiederholung, Nachtrag zum O-Kalkül
- Teile-und-Herrsche-Paradigma, Karatsuba-Ofman
- Rekurrenzen
- Amortisierte Analyse
- Unbounded Arrays

Nachtrag: Übungsblätter

- Bitte Nummer des Tutoriums groß auf die erste Seite schreiben (z. B. eingerahmt oben rechts)
- Oder Deckblatt von `https://webinscribe.ira.uka.de/deckblatt/index.php?course=10516` verwenden
- Bitte Quellen und Hilfsmittel zur Bearbeitung des Übungsblatts auf Übungsblatt angeben
- Blatt 2, Aufgabe 4, a): "Geben Sie einen nicht-trivialen rekursiven Algorithmus in Pseudocode an, der die fehlende Zahl in $\mathcal{O}(n)$ Schritten bestimmt."

Kurze Wiederholung

- Effizienz von Algorithmen:
  - Beispiel: Sortieren von n Zahlen
  - Laufzeit und Eingabegröße
  - O-Kalkül, Asymptotik
- Korrektheit von Algorithmen:
  - Schleifen-Invarianten (Initialisierung, Fortsetzung, Terminierung)

Nachtrag zum O-Kalkül

$$\mathrm{O}(g(n)) = \{f(n) : \text{ es exist. pos. Konstanten } c \text{ und } n_0, \text{ sodass } 0 \leq f(n) \leq c \cdot g(n) \text{ für alle } n \geq n_0\}$$

$$\Theta(g(n)) = \{f(n) : \text{ es exist. pos. Konstanten } c_1, c_2 \text{ und } n_0, \text{ sodass } 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \text{ für alle } n \geq n_0\}$$

$$\Omega(g(n)) = \{f(n) : \text{ es exist. pos. Konstanten } c \text{ und } n_0, \text{ sodass } 0 \leq c \cdot g(n) \leq f(n) \text{ für alle } n \geq n_0\}$$

Und weitere (siehe VL): $\mathrm{o}(g(n)), \omega(g(n))$

Nachtrag zum O-Kalkül

Für nicht-negative $f, g : \mathbb{N} \to \mathbb{R}$ gelten folgende Äquivalenzen:

- $f(n) = \mathrm{O}(g(n)) \iff 0 \leq \limsup_{n\to\infty} \frac{f(n)}{g(n)} = c < \infty\,,$
- $f(n) = \Omega\left(g(n)\right) \iff 0 < \liminf_{n\to\infty} \frac{f(n)}{g(n)} \leq \infty\,,$
- $f(n) = \mathrm{o}(g(n)) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0\,,$
- $f(n) = \omega(g(n)) \iff \limsup_{n\to\infty} \frac{f(n)}{g(n)} = \infty\,,$
- $f(n) = \Theta(g(n)) \impliedby 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} = c < \infty\,.$

Teile-und-Herrsche-Paradigma

Hintergrund des Teile-und-Herrsche-Paradigmas

- Viele Algorithmen sind rekursiv aufgebaut.
- Werden oft mehrmals ausgeführt, um “eng verwandte” Teilprobleme zu lösen.
- Arbeite nach Methode “Teile und Herrsche”.

Paradigma von Teile-und-Herrsche [CLRS04]

- Umfasst drei Schritte auf jeder Rekursionsebene:
  - Teile das Problem in Teilproblemen.
  - Beherrsche die Teilprobleme durch rekursives Lösen. Wenn die Teilprobleme hinreichend klein sind, dann löse diese direkt.
  - Verbinde die Teillösungen zur Lösung des Ausgangsproblem.

Beispiel: Karatsuba-Ofman

1: **Function** *recMult*(a, b)
2: **assert** a und b haben $n = 2k$ Ziffern, n ist Zweierpotenz
3: **if** $n = 1$ **then return** $a \cdot b$
4: Schreibe a als $a_1 \cdot B^k + a_0$
5: Schreibe b als $b_1 \cdot B^k + b_0$
6: $c_{11} := \text{recMult}(a_1, b_1)$
7: $c_{00} := \text{recMult}(a_0, b_0)$
8: $c_{1010} := \text{recMult}((a_1 + a_0), (b_1 + b_0))$
9: $e := c_{11} \cdot B^{2k} + (c_{1010} - c_{11} - c_{00})B^k + c_{00}$
10: **return** e

Karatsuba-Ofman, Schema

$a \cdot b$	a_1	$\cdot$	b_1	$=$	$\mathsf{recMult}(a_1, b_1)$
	a_0	$\cdot$	b_0	$=$	$\mathsf{recMult}(a_0, b_0)$
e	$(a_1 + a_0) \cdot (b_1 + b_0)$			$=$	$\mathsf{recMult}((a_1 + a_0), (b_1 + b_0))$

Karatsuba-Ofman, Beispiel

$1242 \cdot 3163$	$12 \cdot 31 =$
	$42 \cdot 63 =$
	$54 \cdot 94 =$

Karatsuba-Ofman, Beispiel

$1242 \cdot 3163$	$12 \cdot 31 =$
	$42 \cdot 63 =$
	$54 \cdot 94 =$

$12 \cdot 31$	$1 \cdot 3 =$
	$2 \cdot 1 =$
	$3 \cdot 4 =$

Karatsuba-Ofman, Beispiel

$1242 \cdot 3163$	$12 \cdot 31 = 372$
	$42 \cdot 63 =$
	$54 \cdot 94 =$

$12 \cdot 31$	$1 \cdot 3 = 3$
	$2 \cdot 1 = 2$
372	$3 \cdot 4 = 12$

Karatsuba-Ofman, Beispiel

$1242 \cdot 3163$	$12 \cdot 31 = 372$
	$42 \cdot 63 =$
	$54 \cdot 94 =$

$12 \cdot 31$	$1 \cdot 3 = 3$
	$2 \cdot 1 = 2$
372	$3 \cdot 4 = 12$

$42 \cdot 63$	$4 \cdot 6 =$
	$2 \cdot 3 =$
	$6 \cdot 9 =$

Karatsuba-Ofman, Beispiel

1242·3163	12·31= 372
	42·63=2646
	54·94=

12·31	1·3= 3
	2·1= 2
372	3·4=12

42·63	4·6=24
	2·3= 6
2646	6·9=54

Karatsuba-Ofman, Beispiel

$1242 \cdot 3163$	$12 \cdot 31 = 372$
	$42 \cdot 63 = 2646$
	$54 \cdot 94 =$

$12 \cdot 31$	$1 \cdot 3 = 3$
	$2 \cdot 1 = 2$
372	$3 \cdot 4 = 12$

$42 \cdot 63$	$4 \cdot 6 = 24$
	$2 \cdot 3 = 6$
2646	$6 \cdot 9 = 54$

$54 \cdot 94$	$5 \cdot 9 =$
	$4 \cdot 4 =$
	$9 \cdot 13 =$

Karatsuba-Ofman, Beispiel

1242·3163	12·31= 372
	42·63=2646
	54·94=

12·31	1·3= 3
	2·1= 2
372	3·4=12

42·63	4·6=24
	2·3= 6
2646	6·9=54

54·94	5· 9 =45
	4· 4 =16
	9·13=

Karatsuba-Ofman, Beispiel

1242·3163	12·31= 372
	42·63=2646
	54·94=

12·31	1·3= 3
	2·1= 2
372	3·4=12

42·63	4·6=24
	2·3= 6
2646	6·9=54

54·94	5· 9 =45
	4· 4 =16
	9·13=

9·13	0·1=
	9·3=
	9·4=

Karatsuba-Ofman, Beispiel

1242·3163	12·31= 372
	42·63=2646
	54·94=

12·31	1·3= 3
	2·1= 2
372	3·4=12

42·63	4·6=24
	2·3= 6
2646	6·9=54

54·94	5· 9 = 45
	4· 4 = 16
5076	9·13=117

9·13	0·1= 0
	9·3=27
117	9·4=36

Karatsuba-Ofman, Beispiel

1242·3163	12·31= 372
	42·63=2646
3928446	54·94=5076

12·31	1·3= 3
	2·1= 2
372	3·4=12

42·63	4·6=24
	2·3= 6
2646	6·9=54

54·94	5· 9 = 45
	4· 4 = 16
5076	9·13=117

9·13	0·1= 0
	9·3=27
117	9·4=36

Karatsuba-Ofman, Analyse

- Die Analyse von Karatsuba-Ofman ergibt eine Laufzeit von

$$T(n) \leq \begin{cases} 1 & \text{falls } n = 1 \\ 3 \cdot T(\lceil n/2 \rceil) + 10n & \text{falls } n > 1 \end{cases}$$

Mastertheorem

Einfache Form

Für positive Konstanten a, b, c, d, sei $n = b^k$ für ein $k \in \mathbb{N}$.

$$T(n) = \begin{cases} a & \text{falls } n = 1 \text{ Basisfall} \\ cn + dT\left(\frac{n}{b}\right) & \text{falls } n > 1 \Rightarrow \text{teile und herrsche.} \end{cases}$$

Es gilt

$$T(n) = \begin{cases} \Theta(n) & \text{falls } d < b \\ \Theta(n \log n) & \text{falls } d = b \\ \Theta\left(n^{\log_b d}\right) & \text{falls } d > b\,. \end{cases}$$

- Das Mastertheorem gibt uns für Karatsuba-Ofman:

$$T(n) = \Theta\left(n^{\log_2 3}\right)$$

Abschätzen von Rekurrenzen, Substitution

Abschätzen von Rekurrenzen

- Oft kann man Lösungen erraten.
- Falls nicht: obere Schranken zeigen!

Beispiel:

$$T(2n) = 2T(n) + 2n - 1, \quad T(2) = 1, \quad n \text{ ist Zweierpotenz}$$

$$T(n) = O(g(n))?$$

(Für $g(n)$ wie oben.)

Abschätzen von Rekurrenzen

Erster Versuch:

$$T(2n) = 2T(n) + 2n - 1, \quad T(2) = 1$$

- Vermutung: $T(n) = n^2$?
- Müssen zeigen: $T(2n) = (2n)^2$

$$\begin{aligned} T(2n) &= 2T(n) + 2n - 1 \\ &\leq 2n^2 + 2n - 1 \\ &< (2n)^2 \quad (\text{für } n > 0) \end{aligned}$$

$$\Longrightarrow T(n) < n^2 \,.$$

Abschätzen von Rekurrenzen

Zweiter Versuch:

$$T(2n) = 2T(n) + 2n - 1, \quad T(2) = 1$$

- Vermutung: $T(n) = c \cdot n$?
- Müssen zeigen: $T(2n) = c \cdot (2n)$

$$\begin{aligned} T(2n) &= 2T(n) + 2n - 1 \\ &= 2cn + 2n - 1 \\ &> 2cn \quad (\text{für } n > 0) \end{aligned}$$

$$\Longrightarrow T(n) > cn\,.$$

- Suche also Schranke zwischen cn und n^2

Abschätzen von Rekurrenzen

Dritter Versuch:

$$T(2n) = 2T(n) + 2n - 1, \quad T(2) = 1$$

- Vermutung: $T(n) = n \log_2 n$?
- Müssen zeigen: $T(2n) = 2n \log_2 2n$

$$\begin{aligned} T(2n) &= 2T(n) + 2n - 1 \\ &= 2n \log_2 n + 2n - 1 \\ &= 2n(\log_2 n + \log_2 2) - 1 \\ &= 2n \log_2 2n - 1 \\ &< 2n \log_2 2n \end{aligned}$$

$$\Longrightarrow T(n) \leq n \log_2 n\,.$$

Abschätzen von Rekurrenzen

Allgemeiner

$$T(2n) = 2T(n) + 2n - 1, \quad T(2) = 1$$

- Gleichung nur für $n = 2^k$ mit $k \in \mathbb{N}$ definiert!
- Allgemeiner:

$$T(n) = 2T(\lfloor n/2 \rfloor) + n - 1$$

- (Beide stimmen Gleichungen überein für $n = 2^k$.)

Abschätzen von Rekurrenzen

Allgemeiner:

$$T(n) = 2T(\lfloor n/2 \rfloor) + n - 1$$

- es gilt: $T(n) \leq T(n+1)$
- für $2^{k-1} < n < 2^k \Rightarrow T(2^{k-1}) \leq T(n) \leq T(2^k)$
- also:

$$\begin{aligned} T(n) &\leq T(2^k) \\ &\leq c2^k \log_2 2^k \\ &\leq c2n \log_2(2n) \\ &\leq c_1 n \log_2 n \end{aligned}$$

$$\Longrightarrow T(n) \in \mathrm{O}(n \log n) \ .$$

Häufige Formen von Rekurrenzen

- Allgemeine Rekurrenz/Rekursionsgleichung:

$$T(g(n)) = A(T, n)$$

  - $g : \mathbb{N} \to \mathbb{N}$ Funktion die Wachstum der Rekurrenz definiert.
  - $A(T, n)$ ein Ausdruck der $T(n)$ und n enthält.
  - z.B. $g(n) = 2n$, $A(T, n) = 2T(n) + 2n - 1$

- Lineare Rekurrenz k-ter Ordnung:

$$T(n) = a_1 T(n-1) + a_2 T(n-2) + \cdots + a_k T(n-k)$$

mit k Koeffizienten a_i und Anfangswerten $T(i) = b_i$ für $i = 1, \ldots, k$.

- In der Informatik oft: Rekurrenzen aus Divide-Conquer-Algorithmen

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

- Mastertheorem (Wiederholung):

$$T(n) = \begin{cases} a & \text{falls } n = 1 \text{ Basisfall} \\ cn + dT\left(\frac{n}{b}\right) & \text{falls } n > 1 \Rightarrow \text{teile und herrsche.} \end{cases}$$

Es gilt

$$T(n) = \begin{cases} \Theta(n) & \text{falls } d < b \\ \Theta(n \log n) & \text{falls } d = b \\ \Theta\left(n^{\log_b d}\right) & \text{falls } d > b\,. \end{cases}$$

Variablenwechsel

Variablenwechsel

$$T(n) = T(\sqrt{n}) + 1 \quad \text{für } n = 2^{2^i}, \qquad T(4) = 1$$

Variablenwechsel

Wie löst man Rekurrenzen der Form

$$T(n) = T(\sqrt{n}) + 1?$$

- Setze $m = \log n$, also $n = 2^m$.
- Dann:

$$T(2^m) = T(2^{\frac{m}{2}}) + 1$$

Variablenwechsel

$$T(2^m) = T(2^{\frac{m}{2}}) + 1$$

- Trick: setze $S(m) := T(2^m)$
- liefert:

$$S(m) = S\left(\frac{m}{2}\right) + 1$$

- $S(m) = O(?)$

Variablenwechsel

$$T(2^m) = T(2^{\frac{m}{2}}) + 1$$

- Trick: setze $S(m) := T(2^m)$
- liefert:

$$S(m) = S\left(\frac{m}{2}\right) + 1$$

- $S(m) = \mathrm{O}(\log m)$
- Alternativ: nochmaliges Ausführen der Substitution (setze $p = \log_2 m$, erhalte $2^p = m$, $S(2^p) = S(2^{p-1}) + 1$, weiter $U(p) = S(m) = S(2^p) = U(p-1) + 1 = \mathrm{O}(p)$ und einsetzen)

Variablenwechsel

$$\begin{aligned} T(n) &= T(\sqrt{n}) + 1 \\ m &= \log n \\ S(m) &= S(\tfrac{m}{2}) + 1 \end{aligned}$$

- $S(m) = \mathrm{O}(\log m)$
- liefert für $n = 2^{2^i}$:

$$\begin{aligned} T(n) &= T(2^m) \\ &= S(m) \\ &= \mathrm{O}(\log m) \\ &= \mathrm{O}(\log \log n)\ . \end{aligned}$$