WS 2010/11

Effiziente Algorithmen und Datenstrukturen

Ernst W. Mayr

Fakultät für Informatik
TU München

http://www14.in.tum.de/lehre/2010WS/ea/

Wintersemester 2010/11

Kapitel 0 Organisatorisches

- Vorlesungen:
  - 4SWS
    Di 8:30–10:00 (MI 00.13.009A)
    Do 8:30–10:00 (MI HS2)
    Wahlpflichtvorlesung im Fachgebiet Algorithmen und Wissenschaftliches Rechnen (AWR), Bioinformatik
- Übung:
  - 2SWS Zentralübung: Di 14:15–15:45 (MI 00.08.038)
  - Übungsleitung: Tobias Lieber
- Umfang:
  - 4V+2ZÜ, 8 ECTS-Punkte (Modulnr.: IN2003)
- Sprechstunde:
  - nach Vereinbarung

- Vorkenntnisse:
  - Einführung in die Informatik
  - Grundlagen: Algorithmen und Datenstrukturen (GAD)
  - Einführung in die Theoretische Informatik (THEO)
  - Diskrete Strukturen, Diskrete Wahrscheinlichkeitstheorie (DS, DWT)
- Weiterführende Vorlesungen:
  - Effiziente Algorithmen und Datenstrukturen II
  - Randomisierte Algorithmen
  - Komplexitätstheorie
  - Approximationsalgorithmen
  - Internetalgorithmik
  - ...
- Webseite:

  http://wwwmayr.in.tum.de/lehre/2010WS/ea/

- Übungsleitung:
  - Tobias Lieber, MI 03.09.060 (lieber@in.tum.de)
    Sprechstunde: Mi 13:00–14:00
- Sekretariat:
  - Frau Lissner, MI 03.09.052 (lissner@in.tum.de)

- Übungsaufgaben und Klausur:
  - Ausgabe jeweils am Dienstag auf der Webseite der Vorlesung
  - Abgabe eine Woche später vor der Vorlesung
  - Besprechung in der Zentralübung
- Klausur:
  - Zwischenklausur (50% Gewicht), Termin: Fr 17.12.2010, 16:30–18:30 (MI HS1)
  - Endklausur (50% Gewicht), Termin: Fr 18.02.2011, 15:00–17:00 (MI HS1)
  - Wiederholungsklausur, Termin: tba
  - bei den Klausuren sind *keine* Hilfsmittel außer einem handbeschriebenen DIN-A4-Blatt zugelassen
  - Für das erfolgreiche Bestehen des Moduls sind erforderlich:
    1. Bestehen der zweigeteilten Klausur (mindestens 40% der Gesamtpunktzahl)
    2. Erreichen von mindestens 40% der Punkte bei den Hausaufgaben
  - vorauss. 12 Übungsblätter,
    das erste am 26. Oktober, das letzte am 2. Februar,
    jedes 40 Punkte

1. Vorlesungsinhalt

- Grundlagen
  - Maschinenmodelle
  - Komplexitätsmaße
- Höhere Datenstrukturen
  - Suchbäume
  - Hashing
  - Priority Queues
  - selbstorganisierende Datenstrukturen
  - Union/Find-Datenstrukturen
- Sortieren und Selektieren
- Minimale Spannbäume
- Kürzeste Wege
- Matchings in Graphen
- Netzwerkfluss

2. Literatur

- Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman:
  The design and analysis of computer algorithms,
  Addison-Wesley Publishing Company: Reading (MA), 1974

- Thomas H. Cormen, Charles E. Leiserson, Ron L. Rivest, Clifford Stein:
  Introduction to algorithms,
  McGraw-Hill, 1990

- Michael T. Goodrich, Roberto Tamassia:
  Algorithm design: Foundations, analysis, and internet examples,
  John Wiley & Sons, 2002

Volker Heun:
Grundlegende Algorithmen: Einführung in den Entwurf und die Analyse effizienter Algorithmen,
2. Auflage, Vieweg, 2003

Donald E. Knuth:
The art of computer programming. Vol. 1: Fundamental algorithms,
3. Auflage, Addison-Wesley Publishing Company: Reading (MA), 1997

Donald E. Knuth:
The art of computer programming. Vol. 3: Sorting and searching,
3. Auflage, Addison-Wesley Publishing Company: Reading (MA), 1997

- Uwe Schöning:
  Algorithmik,
  Spektrum Akademischer Verlag, 2001
- Christos H. Papadimitriou, Kenneth Steiglitz:
  Combinatorial Optimization: Algorithms and Complexity,
  Prentice Hall, 1982
- Steven S. Skiena:
  The Algorithm Design Manual,
  Springer, 1998

Kapitel I Grundlagen

1. Ein einleitendes Beispiel

Berechnung von F_n, der n-ten Fibonacci-Zahl:

$$\begin{aligned}
F_0 &= 0 \ , \\
F_1 &= 1 \ , \\
F_n &= F_{n-1} + F_{n-2} \text{ für alle } n \geq 2 \ .
\end{aligned}$$

Also:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

1. Methode: Funktionales Programmieren

$$
\begin{aligned}
f(n) := \ & \textbf{if } n = 0 \textbf{ then } 0\\
& \textbf{elif } n = 1 \textbf{ then } 1\\
& \textbf{else } f(n-1) + f(n-2)\\
& \textbf{fi}
\end{aligned}
$$

Sei $T(n)$ der Zeitbedarf für die Berechnung von $f(n)$. Dann gilt:

$$
\begin{aligned}
T(0) &= T(1) = 1\ ;\\
T(n) &= T(n-1) + T(n-2) \text{ für } n \geq 2\ .
\end{aligned}
$$

Damit gilt:

$$T(n) = c_1 \left(\frac{1 + \sqrt{5}}{2} \right)^n + c_2 \left(\frac{1 - \sqrt{5}}{2} \right)^n$$

für geeignete Konstanten c_1 und c_2.

Einsetzen der Werte für $n = 0$ und $n = 1$ ergibt

$$\begin{aligned} c_1 &= \frac{1}{2} + \frac{\sqrt{5}}{10} \\ c_2 &= \frac{1}{2} - \frac{\sqrt{5}}{10} \end{aligned}$$

und damit

$$T(n) = \left(\frac{1}{2} + \frac{\sqrt{5}}{10} \right) \left(\frac{1 + \sqrt{5}}{2} \right)^n + \left(\frac{1}{2} - \frac{\sqrt{5}}{10} \right) \left(\frac{1 - \sqrt{5}}{2} \right)^n .$$

2. Methode: Dynamische Programmierung

array $F[0:n]$; $F[0] := 0$; $F[1] := 1$;
for $i := 2$ **to** n **do**
$\quad F[i] := F[i-1] + F[i-2]$

Der Zeitbedarf dieses Algorithmus ist offensichtlich

$$\mathcal{O}(n)\ .$$

3. Methode: Direkt (mit etwas mathematischer Arbeit)

$$F(n) := \text{round}\left(\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$$
$$= \left\lfloor \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n + \frac{1}{2} \right\rfloor .$$

Der Zeitbedarf hier ist

$$\mathcal{O}(\log n) .$$