WS 2006/07

Effiziente Algorithmen und Datenstrukturen

Ernst W. Mayr

Fakultät für Informatik
TU München

http://www14.in.tum.de/lehre/2006WS/ea/

Wintersemester 2006/07

Kapitel 0 Organisatorisches

- Vorlesungen:
  - 4SWS Di 8:30–10:00 (MW1801), Fr 8:30–10:00 (MI HS1) Wahlpflichtvorlesung im Gebiet Algorithmen (Theoretische Informatik, Informatik III), Bioinformatik
- Übung:
  - 2SWS Zentralübung: Mi 9:30–11:00 (Multimedia-Hörsaal 00.13.009A)
  - Übungsleitung: Matthias Baumgart
- Umfang:
  - 4V+2ZÜ, 8 ECTS-Punkte
- Sprechstunde:
  - Fr 12:00–13:00 oder nach Vereinbarung

- Vorkenntnisse:
  - Einführung in die Informatik I/II/III/IV
  - Diskrete Strukturen I/II (DS, DWT)
- Weiterführende Vorlesungen:
  - Effiziente Algorithmen und Datenstrukturen II
  - Randomisierte Algorithmen
  - Komplexitätstheorie
  - Internetalgorithmik
  - ...
- Webseite:

  http://wwwmayr.in.tum.de/lehre/2006WS/ea/

- Übungsleitung:
  - Matthias Baumgart, MI 03.09.060 (baumgart@in.tum.de)
    Sprechstunde: Dienstag, 10:30Uhr und nach Vereinbarung
- Sekretariat:
  - Frau Schmidt, MI 03.09.052 (schmiann@in.tum.de)

- Übungsaufgaben und Klausur:
  - Ausgabe jeweils am Freitag in der Vorlesung bzw. auf der Webseite der Vorlesung
  - Abgabe eine Woche später vor der Vorlesung
  - Besprechung in der Zentralübung
- Klausur:
  - Zwischenklausur (50% Gewicht), Termin: 15.12.2006, 13–15Uhr, MW 0001
  - Endklausur (50% Gewicht), Termin: 13.02.2007, 10–12Uhr, MW 0001
  - Wiederholungsklausur, Termin TBA
  - bei den Klausuren sind *keine* Hilfsmittel außer einem handbeschriebenen DIN-A4-Blatt zugelassen
  - Zulassungsvoraussetzung (außer für Studierende im Diplomstudiengang Informatik) sind 40% der erreichbaren Hausaufgabenpunkte
  - vorauss. 12 Übungsblätter, das letzte am 26. Januar 2007, 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


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


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


Uwe Schöning:
Algorithmik,
Spektrum Akademischer Verlag, 2001

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

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

$$c_1 = \frac{1}{2} + \frac{\sqrt{5}}{10}$$
$$c_2 = \frac{1}{2} - \frac{\sqrt{5}}{10}$$

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**
  $F[i] := F[i-1] + F[i-2]$

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

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

2. Versuch einer Definition

Was sind „kombinatorische“ Algorithmen?

Eine mögliche Antwort:

Probleme, die, könnten wir alle Fälle aufzählen, trivial wären, aber eine sehr große Anzahl von Fällen haben.

Beispiele: Hamiltonscher Kreis (NP-vollständig), Eulerscher Kreis (P)