Informatik II - SS 2014

(Algorithmen & Datenstrukturen)

Vorlesung 1 (29.4.2014)

Organisation, Übungen, Sortieren I

UNI FREIBURG

Fabian Kuhn

Algorithmen und Komplexität

Allgemeines Thema der Vorlesung

UNI FREIBURG

- Letztes Semester haben Sie (die Informatiker/innen) die Grundzüge des Programmierens gelernt
- Fragen der Effizienz spielten eine untergeordnete Rolle
- Dies ist das generelle Thema dieser Vorlesung

  Wie schnell ist mein Programm? Wie misst man das?

  Wie kann ich es schneller machen?

  Wie kann ich beweisen, dass es immer so schnell läuft?

- Manchmal geht es auch um Sparsamkeit im Platzverbrauch oder andere Ressourcen, aber hier meistens um **Zeit**

Etwas spezifischer...

UNI FREIBURG

Algorithmen:

- Algorithmenentwurf: Wie löst man ein geg. Problem effizient?
- Wie analysiert man einen Algorithmus?
  - O-Notation, Laufzeitanalyse, Korrektheit (auch math. Beweise)
- Beispiele: Sortieren, Suchen, Graphenalgorithmen (kürzeste Wege, Spannbäume, Breiten-/Tiefen-Suche)

Datenstrukturen:

- Wie legt man Daten ab, so dass man schnell darauf zugreifen kann?
- Gute/geeignete Datenstrukturen $\Longrightarrow$ effiziente Algorithmen
- Beispiele: Hashtabellen, Suchbäume, (Prioritäts-)Warteschlangen, dynamische Arrays, Repräsentation von Graphen

Organisation - Vorlesung

UNI FREIBURG

Vorlesung (101-00-026)

- Dienstag: 16:15-17:45
- Mittwoch: 16:15-17:45

Webpage: http://ac.informatik.uni-freiburg.de/teaching/ss_14/info2.html

Übungen

- Ein Übungsblatt pro Woche (12 Übungsblätter)
- Übungen ca. ½ theoretisch und ½ praktisch
- Ablauf der Übungen alles online → gleich mehr...

Aufwand

- 8 ECTS (für die Informatiker) = ca. 240 Arbeitsstunden
- Bei 160h Vorlesungen/Übungen und 80h Klausurvorbereitung
  - ca. **10 Stunden pro Übung**...

Übungen / Klausurzulassung

UNI FREIBURG

Übungsblätter

- 12 Übungsblätter, pro Übungsblatt max. 20 Punkte
- Um an der Klausur teilzunehmen, müssen Sie 50% der Punkte aus den Übungen machen.
- Übungen sind grundsätzlich die beste Klausurvorbereitung!!

Übungsgruppen

- Keine Übungsgruppentermine (alles online)
- Assistenten:
  Sebastian Daum (theor. Übungen), Oleksii (Alex) Saukh (prakt. Ü.)
- Tutoren:
  Stefan Dilger, Sebastian Dufner, Martin Goth, Kevin Hättig, Christian Klinger, Fabian Ullrich und Simon Weidner

Kursverwaltungssystem - Daphne

UNI FREIBURG

benutzen **Daphne** als **Kursverwaltungssystem**

- Link auf der Vorlesungswebpage: **bitte anmelden!**
- In Daphne haben Sie eine Übersicht über folgende Infos
  - Wer ihr/e Tutor/in ist
  - Ihre Punkte in den Übungsblättern
  - Infos zum aktuellen Übungsblatt
  - Link zum Forum ... gleich mehr dazu
  - Link zum SVN ... gleich mehr dazu
  - Link zu unseren Coding Standards ... gleich mehr dazu
  - Link zu unserem Build System ... gleich mehr dazu
- Daphne kennen Sie von Informatik I ???
- Das gleiche System wird auch in “Programmieren in C++” verwendet…

Forum

UNI FREIBURG

Forum zu Vorlesung / Übungen

- Link dazu auf der Vorlesungswebseite / Daphne-Seite
- (Alle) Fragen zu Übungen / Vorlesung bitte übers Forum fragen!
  - Wir können so am schnellsten antworten…
  - Antwort hilft oft vielen anderen auch…
- Nutzen Sie die Gelegenheit, über's Forum zu fragen!
- Stellen Sie konkrete Fragen!
  - Auf Fragen wie "Programm läuft nicht", "Ich habe keine Ahnung, wie ich das beweisen soll" können Sie keine Antwort erwarten
  - Bei Fragen zum Programmieren: Bitte Code-Segmente angeben, welche nicht funktionieren, jedoch nicht ganze Programme ins Forum posten!
  - Angabe von Fehlermeldung
- Falls nötig, können Sie sich natürlich mit Ihrem Tutor auch mal persönlich treffen…

SVN (Subversion)

UNI FREIBURG

SVN : http://subversion.apache.org

- Kennen Sie bereits aus Informatik I?
- Dateien liegen auf einem zentralen Server, in einem sogenannten repository, die typische Operationen sind
  - Update: neuste Version vom Server ziehen
  - Commit: letzte Änderungen auf den Server hochladen
- Vollständige Historie von allen Änderungen an den Dateien
- Insbesondere nützlich für das Schreiben von Code
- Wir benutzen das hier für
  - die Abgaben Ihrer Übungsblätter (Code + alles andere)
  - das Feedback von Ihrem Tutor
  - Vorlesungsdateien / Musterlösungen

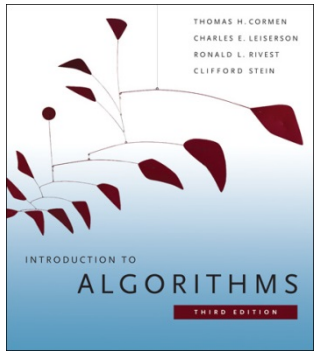
Programmierübungen

UNI FREIBURG

Richtlinien für die Abgabe der Programmierübungen:

- Sie können die Übungen in Java oder C++ machen
- Unit tests für alle nichttrivialen Methoden
  - Ohne Test ist die Chance gross, dass die Methode falsch ist…
  - Vereinfacht das Debuggen von grösseren Projekten
  - Wir/Sie können so testen, ob das Programm das Richtige tut
- Befolgen von Stylesheets (checkstyle / cpplint)
  - Macht den Code leserlich (auch für andere!)
- Standardisiertes Build-Framework (jenkins)
  - So können wir effizient Ihre Programme ausprobieren und testen
- Ich werde dies heute und morgen mal demonstrieren…

Literatur : Bücher

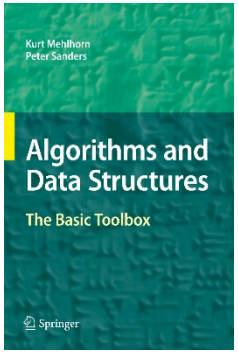


T. Cormen, C. Leiserson, R. Rivest, C. Stein
Introduction to Algorithms, Third Edition,
MIT Press, 2009

Klassisches Lehrbuch zum Thema (auf Englisch)

T. Ottmann, P. Widmayer
Algorithmen und Datenstrukturen, 5. Auflage,
Spektrum Akademischer Verlag, Heidelberg, 2012

Deutsche Alternative (aus Freiburg / Zürich)



K. Mehlhorn, P. Sanders
Algorithms and Data Structures,
Springer, 2008, online verfügbar unter

http://www.mpi-inf.mpg.de/~mehlhorn/Toolbox.html

Literatur : Online

UNI FREIBURG

Wikipedia:

- ist bei Standardalgorithmen / -datenstrukturen recht gut…
- z.T. auch für weitergehende Algorithmen

Alte Vorlesungen, z.B.:

- verfügbares Material von früheren Informatik II Vorlesungen
  - 2013: https://ad-wiki.informatik.uni-freiburg.de/teaching/AlgoDatSS2013
- MIT Courseware: http://ocw.mit.edu
  - u.a. mit Aufzeichnungen von 2005 / 2011

Acknowledgement

UNI FREIBURG

- Prof. Bast hat mir netterweise alles Material der letztjährigen Vorlesung zur Verfügung gestellt

- Vorlesung wird ähnlich wie die letztjährige sein, aber möglicherweise etwas theoretischer…

Sortieren

transitiv: $x \leq y$ and $y \leq z \Rightarrow x \leq z$

$x \leq y \wedge y \leq x \Rightarrow x = y$

$x \leq y \vee y \leq x$

Problemdefinition

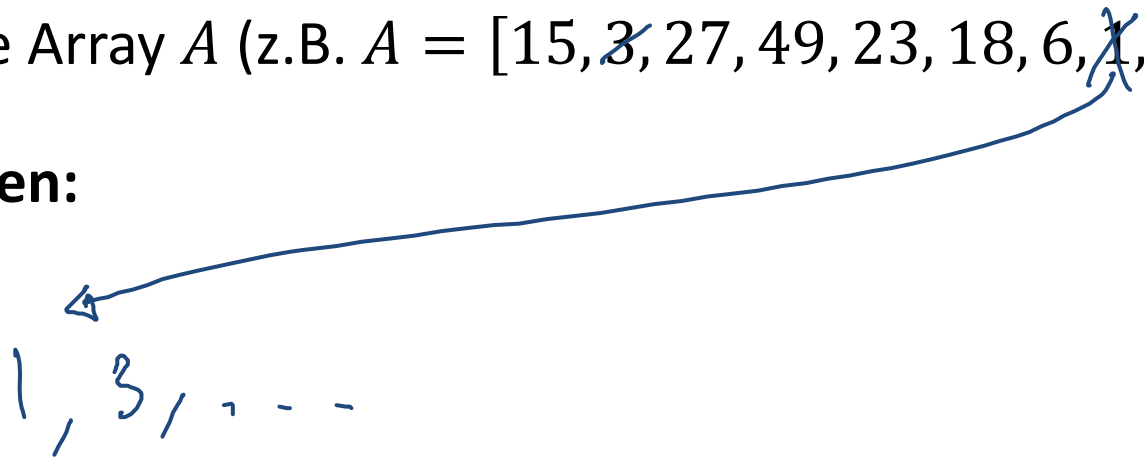
- **Eingabe**: Sequenz von n Elementen $x_1, \dots, x_n$
- Ordnungsrelation $\leq$ auf den Elementen
  - Vergleichsoperation, mit welcher zwei beliebige Elemente miteinander verglichen werden können
  - Bsp. 1: Sequenz von Zahlen mit üblicher $\leq$-Relation
  - Bsp. 2: Sequenz von Strings mit lexikographischer (alphabetischer) Ordnung
- **Ausgabe**: Gemäss $\leq$-Ordnung sortierte Sequenz der n Elemente
- **Beispiel**:
  - Eingabe: $[15, 3, 27, 49, 23, 18, 6, 1, 31]$
  - Ausgabe: $[1, 3, 6, 15, 18, 23, 27, 31, 49]$
- Sortieren wird in (fast) allen grösseren Programmen gebraucht!

Algorithmus zum Sortieren?

UNI FREIBURG

Aufgabe: Sortiere Array A (z.B. $A = [15, 3, 27, 49, 23, 18, 6, 1, 31]$)

Algorithmus, Ideen:

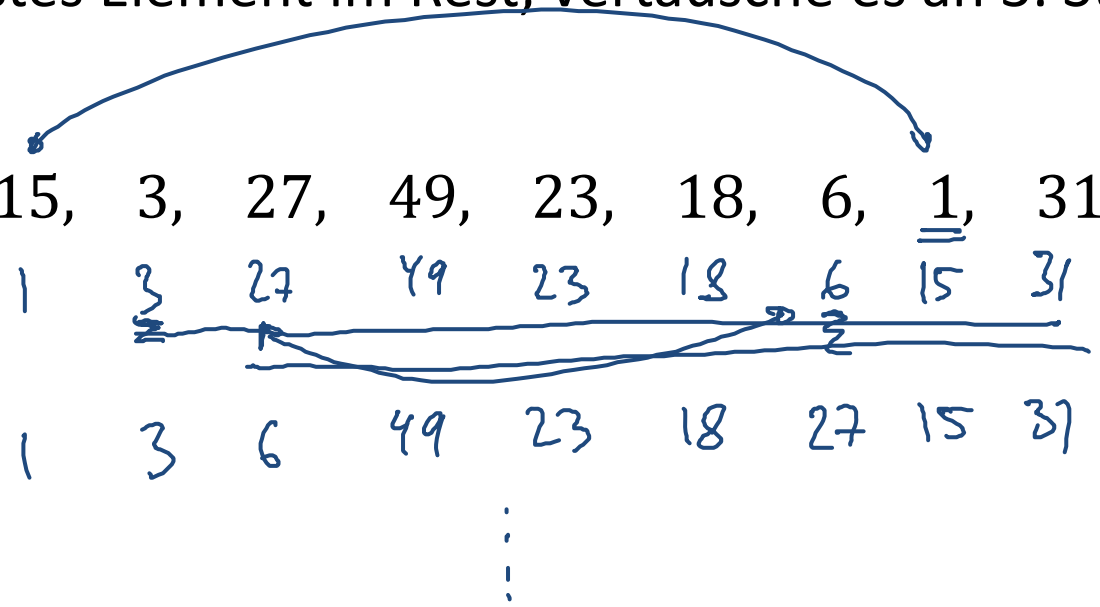


SelectionSort Algorithmus

SelectionSort (informell):

1. Finde kleinstes Element im Array, vertausche es an 1. Stelle
2. Finde kleinstes Element im Rest, vertausche es an 2. Stelle
3. Finde kleinstes Element im Rest, vertausche es an 3. Stelle
4. ...

Beispiel: $A = [15,\ 3,\ 27,\ 49,\ 23,\ 18,\ 6,\ 1,\ 31]$



SelectionSort: Pseudocode

Eingabe: Array A der Grösse n

```
SelectionSort(A):
 for i=0 to n-2 do
   // find min in A[i..n-1]
   minIdx = i
   for j=i+1 to n-1 do
      if A[j] < A[minIdx] then
         minIdx = j

   // swap A[i] with min of A[i..n-1]   swap(A[i], A[minIdx])
   tmp = A[i]
   A[i] = A[minIdx]
   A[minIdx] = tmp
```

SelectionSort: Pseudocode

Eingabe: Array A der Grösse n

```
SelectionSort(A):
 1: for i=0 to n-2 do
 2:   // find min in A[i..n-1]
 3:   minIdx = i
 4:   for j=i to n-1 do
 5:     if A[j] < A[minIdx] then
 6:       minIdx = j
 7:   // swap A[i] with min of A[i..n-1]
 8:   tmp = A[i]
 9:   A[i] = A[minIdx]
10:   A[minIdx] = tmp
```