Informatik II: Algorithmen und Datenstrukturen SS 2017

Vorlesung 1a, Dienstag, 25. April 2017
(Gesamtüberblick, Sortieren, Kurssysteme)

Prof. Dr. Hannah Bast
Lehrstuhl für Algorithmen und Datenstrukturen
Institut für Informatik
Universität Freiburg

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Überblick über die gesamte Veranstaltung
  - Inhalt: kurze Übersicht
  - Ablauf: Vorlesungen, Übungen, Stil, Aufwand
  - Klausur: Zulassungskriterien, Termin, Note
- Sortieren, Coding Standards, Kurssysteme
  - Sortieren: ein erster Algorithmus + Programm dazu
  - Coding Standards: Unit Tests, Stil, Build Framework
  - Kurssysteme: Daphne, Forum, SVN, Jenkins
  - **ÜB 1:** Implementieren Sie MergeSort (kommt morgen dran) und vergleichen Sie die Laufzeit mit MinSort (wie heute)

Inhalt 1/4

- Ausgangsbasis
  - Sie können schon kleinere Programme schreiben

    Zum Beispiel: berechnen, ob eine gegebene Zahl prim ist
  - In Python oder Java oder C++ (Ihnen überlassen!)

    Die meisten von Ihnen haben im 1. Semester **Python** gelernt

    Parallel lernen die meisten jetzt im 2. Semester **Java**
  - Verständnis einiger Grundkonzepte des Programmierens, z.B.

    Variablen, Funktionen, Schleifen, Ein- und Ausgabe, Rekursion

    Wer da noch Lücken hat, kann auch mitmachen, aber es wird dann deutlich mehr Aufwand … siehe Folie 11 zum Aufwand

Inhalt 2/4

- Grundbausteine, die man immer wieder braucht
  - Zum Beispiel:

    Sortieren
    Dynamische Felder / Assoziative Felder / Hash Maps
    Prioritätswürgeschlangen
    Suchbäume
    Berechnen kürzester Wege in Netzwerken
    Suchen von Mustern in Zeichenketten

  - In Python, Java, C++ gibt es dafür Bibliotheken … in dieser Vorlesung lernen Sie, was dahinter steckt

    Für komplexere Projekte braucht man oft eine Variante von den Grundbausteinen … oder neue Bausteine, die es noch gar nicht gibt

Inhalt 3/4

- Effizienz
  - In der "Informatik I" haben Fragen der Effizienz noch kaum eine Rolle gespielt ... in dieser Veranstaltung ist es ein ganz wesentlicher Aspekt:

    Wie schnell läuft mein Programm / eine Funktion?

    Wie kann ich es schneller machen?

    Wie weiß ich, ob es noch schneller geht?

    Manchmal geht es auch um andere Ressourcen, zum Beispiel Speicherverbrauch, aber meistens geht es um **Geschwindigkeit**

Inhalt 4/4

- Methoden
  - Laufzeitanalyse (O-Notation)

    Zum Beispiel: mein Programm läuft in Linearzeit (= gut)

    Oder: mein Programm hat quadratische Laufzeit (= böse)
  - Den einen oder anderen Korrektheits**beweis**

    Zum Beispiel: dynamische Felder mit konstanter Laufzeit pro Operation (im Durchschnitt)
  - Die Mathematik, die wir in diesem Kurs verwenden, ist sehr "basic", aber es ist schon Mathematik, nicht nur "Rechnen"

    Wir machen hier keine Theorie um der Theorie willen, sondern nur da wo hilfreich oder nötig, dann aber gerne !

Ablauf 1/5

- Vorlesungen
  - **Di 14:15 – 15:45** Uhr und **Mi 16:15 – 17:45** Uhr, im **HS 026**
  - Insgesamt 25 Vorlesungstermine (der letzte am 26. Juli)

    Keine Vorlesung am 6. + 7. Juni (Pfingstpause)
  - Die Vorlesungen werden aufgezeichnet

    Folien + Audio + Video … Schnitt: Alexander Monneret
  - Auf unserem Wiki finden Sie alle Kursmaterialien

    Aufzeichnungen, Folien, Übungsblätter, Code aus der Vorlesung + ggf. zusätzliche Hinweise, Musterlösungen

    Auch im **SVN**, Unterordner ***/public*** (außer die Aufzeichnungen)

Ablauf 2/5

- Übungen
  - Die Übungen sind der wichtigste Teil der Veranstaltung
  - Sie bekommen jede Woche ein Übungsblatt, insgesamt **12**
  - Das können Sie machen wo und wann Sie wollen (im Rahmen der Abgabefrist)
  - Aber Sie müssen es **selber** machen

    Sie können gerne zusammen über die Übungsblätter nachdenken, diskutieren, etc. … aber die Lösungen bzw. Programme müssen Sie dann **100% selber** schreiben

    Auch das teilweise Übernehmen gilt als Täuschungsversuch, siehe auch die Erklärungen auf Seite 2 vom 1. Übungsblatt

Ablauf 3/5

- Übungsgruppen
  - Wie in allen unseren Lehrveranstaltungen läuft der Übungsbetrieb komplett **online**
  - Sie bekommen jede Woche Feedback zu Ihren Aufgaben, von Ihrem Tutor bzw. Ihrer Tutorin:

    Maya Schöchlin, Sebastian Holler, Daniel Tischner, Daniel Bindemann, Danny Stoll, Simon Selg
  - Assistent der Vorlesung ist: **Axel Lehmann**
  - Für Fragen aller Art gibt es ein **Forum** (siehe Wiki)

    Es wird in der zweiten oder dritten Woche auch eine Fragestunde geben … erfahrungsgemäß wollen die alle haben und dann geht aber keiner hin

Ablauf 4/5

- Stil der Veranstaltung
  - Vorlesungen: viele Beispiele + viele Programme (live)

    Die Details müssen Sie sich selber aneignen, sonst lernt man sie nicht, u.a. dafür sind die Übungsblätter da

    Die Vorlesungen enthalten alles, was man dafür braucht

    Am Ende jedes Foliensatzes: weiterführende Literatur
- Praxisbezug
  - Die Übungsblätter sind zu 1/3 Theorie, zu 2/3 Praxis

    Möglichst praxisnahe Übungsaufgaben, damit Sie sehen, wozu man das alles braucht ... später mehr, weil komplexer

    Theorie genau da wo nötig bzw. hilfreich zum Verständnis

Ablauf 5/5

- Aufwand / ECTS Punkte
  - Veranstaltung zählt 8 ECTS Punkte = 240 Arbeitsstunden
  - Insgesamt 25 Vorlesungen = 50 (relaxte) Arbeitsstunden
  - Außerdem 12 Übungsblätter

    Sorgfältige Bearbeitung der Übungsblätter + gründliches Verständnis dahinter = beste Vorbereitung auf die Klausur
  - Optionen für Ihr Zeitmanagement ÜB = Übungsblatt

    A. 9-12 Std / ÜB, wenig lernen für Klausur **EMPFOHLEN**

    B. 4-6 Std / ÜB, viel lernen für Klausur MINIMUM

    C. 0 Std / ÜB, ??? lernen für Klausur **UNMÖGLICH**

Klausur 1/2

- Zulassung
  - Für jedes Übungsblatt gibt es maximal **20** Punkte

    12 Übungsblätter → maximal 240 Punkte
  - Davon müssen Sie insgesamt mindestens die Hälfte (120 Punkte) erreichen, um zur Klausur zugelassen zu werden

    Das können alle schaffen, die kontinuierlich mitarbeiten
  - Für das Ausfüllen des Evaluationsbogens am Ende gibt es noch mal 20 Punkte, mit denen Sie die Punktezahl Ihres schlechtesten Übungsblattes ersetzen können

    U.a. als Joker für Kranksein oder sonstige Umstände

Klausur 2/2

- Termin
  - Am [Wochentag], den [Tag]. [Monat] 2017 von 14 – 17 Uhr
  - 6 Aufgaben à 20 Punkte, wir zählen die besten 5
  - Also maximal 100 Punkte
- Die Endnote
  - ... ergibt sich linear aus der Punktzahl in der Klausur

50 – 54:	4.0;	55 – 59:	3.7;	60 – 64:	3.3
65 – 69:	3.0;	70 – 74:	2.7;	75 – 79:	2.3
80 – 84:	2.0;	85 – 89:	1.7;	90 – 94:	1.3
95 – 100:	1.0				

Sortieren 1/4

- Problemdefinition
  - **Eingabe:** eine Folge von n Elementen $x_1, \ldots, x_n$

    Sowie ein transitiver Operator $\leq$ auf diesen Elementen

    Transitiv: $x \leq y$ und $y \leq z \Rightarrow x \leq z$
  - **Ausgabe:** die n Elemente in gemäß diesem Operator sortierter Reihenfolge, zum Beispiel

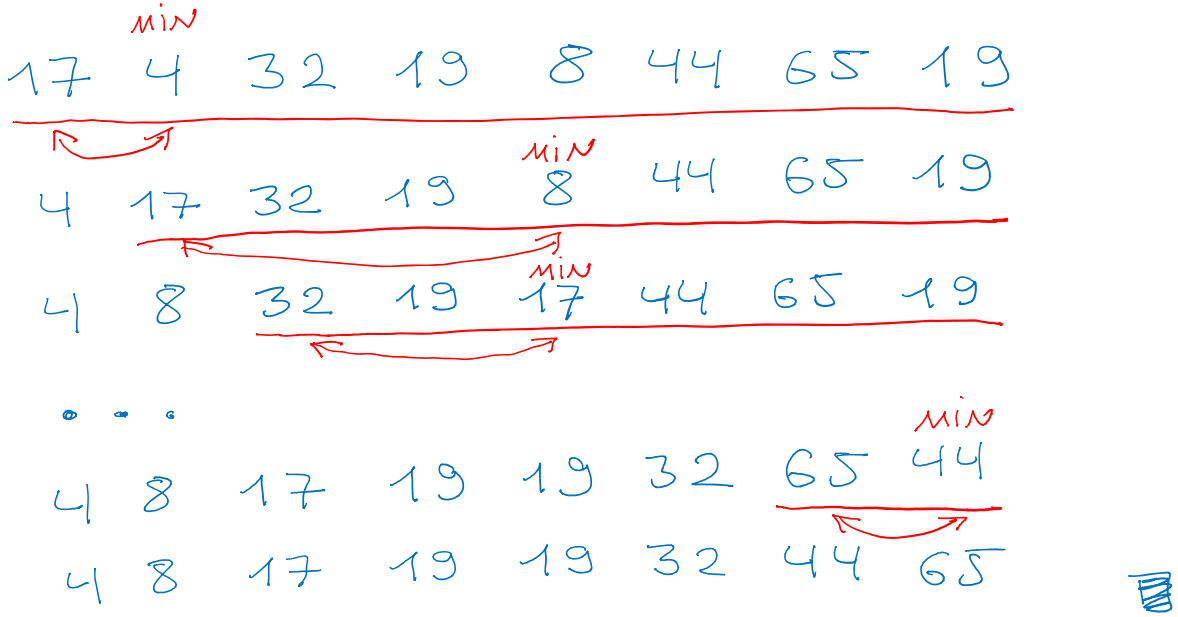
    Eingabe: 17, 4, 32, 19, 8, 44, 65, 19

    Ausgabe: 4, 8, 17, 19, 19, 32, 44, 65
- Wo braucht man Sortieren?
  - In praktisch **jedem** größeren Programm

    Beispiel: Bauen eines Indexes für eine Suchmaschine

Sortieren 2/4

- MinSort: Informale Beschreibung
  - Finde das Minimum und tausche es an die erste Stelle
  - Finde das Minimum im Rest und tausche es an die zweite Stelle
  - Finde das Minimum im Rest und tausche es an die dritte Stelle
  - usw.



Sortieren 3/4

- MinSort: Programm
  - Ich mache es heute in Python vor
  - Morgen zeige ich auch Code in Java und C++

  Im Laufe der Vorlesung plane ich öfter mal hin- und her zu wechseln, je nach Problemstellung

  Sie können sich auch für jedes ÜB neu entscheiden (es gibt keine Abhängigkeiten der ÜB untereinander)

  Es gibt aber evtl. einige wenige ÜB, wo Python nicht erlaubt oder möglich sein wird (weil manche Effizienz-Betrachtungen in Python einfach keinen Sinn machen)

Sortieren 4/4

UNI FREIBURG

- MinSort: Laufzeit
  - Wir testen das mal für verschiedene Eingabegrößen

    Beobachtung: Es wird "unverhältnismäßig" langsamer, je mehr Zahlen sortiert werden
  - Um das genauer zu machen, malen wir ein Schaubild:

    x-Achse: Eingabegröße, y-Achse: Laufzeit

    Beobachtung: Die Laufzeit "wächst schneller als linear"

    Das heißt, für doppelt so viele Zahlen braucht es (viel) **mehr** als doppelt so viel Zeit
  - Nächste Woche machen wir das präziser, diese Woche bleiben wir erst mal noch bei Schaubildern

    Für das 1. Übungsblatt sollen Sie auch eins malen

Coding Standards 1/3

UNI FREIBURG

- Was und warum
  - Sie sollen auch (weiter) **gutes Programmieren** lernen
  - Dazu gehören … für: Python / Java / C++

    Unit Tests für jede nicht-triviale Funktion
    Für korrekten Code … mit: doctest / junit / gtest

    Einheitlicher Stil Ihrer Codes
    Für verständlichen Code … mit: flake8 / checkstyle / cpplint

    Build-Framework zum automatischen Testen
    Damit wir nicht bei jeder Abgabe getrennt schauen müssen, wie man den Code testet etc … mit: make / ant / make
  - **Ich werde das jetzt mal für Python vormachen**

    Java und C++ Code im SVN unter public/code/vorlesung-01

Coding Standards 2/3

UNI FREIBURG

- Warum Unit Tests
  - **Grund 1:** Eine nicht-triviale Methode ohne Unit Test ist mit hoher Wahrscheinlichkeit nicht korrekt
  - **Grund 2:** Macht das Debuggen von größeren Programmen viel leichter und angenehmer
  - **Grund 3:** Wir und Sie selber können automatisch testen ob Ihr Code das tut was er soll

    Am Anfang nervt es vielleicht etwas, aber mit der Zeit kapiert man, wofür es gut ist und irgendwann macht es dann sogar Spaß!

Coding Standards 3/3

- Mindestanforderungen für diese Veranstaltung
  - Ein Unit Test für **jede nicht-triviale Funktion**
  - Für mindestens **eine typische** Eingabe
  - Für mindestens **einen kritischen** "Grenzfall", wenn es einen solchen gibt **...** z.B. leeres Feld beim Sortieren

    Das ist in der Regel sehr wenig Arbeit … mit sehr großem Nutzen (für Sie und für uns)

    Siehe auch noch mal Rückseite vom 1. Übungsblatt

Kurssysteme 1/4

UNI FREIBURG

- **Daphne**, unser Kursverwaltungssystem
  - Link auf der Wiki-Seite zum Kurs, **bitte anmelden!**
  - In Daphne haben Sie eine Übersicht über folgende Infos
    - Wer Ihr Tutor ist
    - Ihre Punkte in den Übungsblättern
    - Link zu den Coding Standards ... siehe letzte Folien
    - Link zum Forum ... siehe nächste Folie
    - Link zum SVN ... siehe übernächste Folie
    - Link zu Jenkins ... siehe überübernächste Folie

Kurssysteme 2/4

- **Forum** zur Veranstaltung
  - Link dazu auf dem Wiki und auf Ihrer Daphne Seite
  - Bitte fragen Sie, wann immer etwas nicht klar ist !

    Aber bitte möglichst **konkret** fragen: siehe Anleitung dazu auf dem Wiki

    Und fragen Sie uns bitte nicht einzeln, sondern auf dem Forum ... praktisch immer interessiert das auch andere
  - Entweder **Ich** oder **Axel Lehmann** oder eine*r der **Tutor*innen** wird dann möglichst schnell antworten

Kurssysteme 3/4

- **SVN** = Subversion http://subversion.apache.org/
  - Dateien liegen bei uns auf einem zentralen Server

    Typische Operationen: svn **add**, svn **commit**, svn **update**

    Vollständige Historie von allen Änderungen an den Dateien

    Kurze Anleitung auf dem Wiki
  - Wir benutzen das hier für fast alles:

    Die Abgaben Ihrer Übungsblätter (Code + alles andere)

    Das Feedback von Ihrem Tutor / Ihrer Tutorin

    Folien / Vorlesungsdateien / Übungsblätter / Musterlösungen
  - **Ich werde das jetzt mal kurz vormachen**

Kurssysteme 4/4

UNI FREIBURG

- **Jenkins** ist unser automatisches Build System
  - Damit können Sie schauen, ob Ihr Code, so wie Sie ihn in unser SVN hochgeladen haben, kompiliert und läuft

    Insbesondere ob die **Unit Tests** alle durchlaufen

    Und ob der **Stil** in Ordnung ist
  - **Zeige ich Ihnen auch gerade mal**
  - Wichtig: Abgaben, für die "compile" fehlschlägt oder die keine Tests haben werden **nicht** korrigiert

    Es wäre sonst unzumutbar viel Arbeit für die Tutoren

Literatur / Links

- Weiterführende Literatur
  - Mehlhorn / Sanders: Algorithms and Data Structures, The Basic Toolbox

    Neueres Lehrbuch zu Algorithmen und Datenstrukturen, mit praktischerer Ausrichtung als ältere Lehrbücher. Online:

    http://www.mpi-inf.mpg.de/~mehlhorn/Toolbox.html
  - Für fast alle grundlegenden Algorithmen und Datenstrukturen gibt es inzwischen gute bis sehr gute Wikipedia Artikel

    Und wenn man etwas rumgoogelt, findet man auch die eine oder andere hilfreiche Animation oder Live-Demo

    Eigentlich sind die Vorlesungen aber self-contained, d.h. Sie kriegen alles erklärt, was Sie für die Übungsblätter brauchen