Informatik II: Algorithmen und Datenstrukturen SS 2013

Vorlesung 1a, Dienstag, 16. April 2013

(Organisatorisches, Programmierumgebung, Sortieren)

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

UNI FREIBURG

Allgemeines zu dieser Vorlesung 1/3

UNI FREIBURG

- Thema allgemein
  - Letztes Semester haben Sie (die Informatiker/innen) die Grundzüge des Programmierens gelernt (in Java und Haskell)
  - Fragen der Performanz spielten noch kaum eine Rolle
  - Genau darum geht es jetzt in dieser Vorlesung

    Wie schnell ist mein Programm?

    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 bei uns hier meistens um **Zeit**

Allgemeines zu dieser Vorlesung 2/3

UNI FREIBURG

- Themen im Speziellen
  - Sortieren, dynamische Felder, assoziative Felder, Hashing, Prioritätswarteschlangen, Listen, Graphen-Algorithmen, Suchbäume, String-Algorithmen, etc.
- Methodologisches
  - Laufzeitanalyse
  - O-Notation
  - Den einen oder anderen Korrektheits**beweis**
  - Die Mathematik, die wir in diesem Kurs verwenden, ist sehr "basic", aber es ist schon Mathematik, nicht nur "Rechnen"

Allgemeines zu dieser Vorlesung 3/3

UNI FREIBURG

- Art und Weise
  - Großer Praxisbezug (wie in allen meinen Vorlesungen)
  - Aber auch Theorie (sofern sie der Praxis nutzt)
  - Übungsblätter ca. 2/3 praktisch, ca. 1/3 theoretisch
  - Zum Ablauf des Übungsbetriebs gleich mehr ...
- Aufwand
  - Es gibt für die Veranstaltung 8 ECTS (für die Informatiker)
  - Das entspricht 240 Arbeitsstunden für das ganze Semester
  - Davon ca. 160h für VL + Übungen, ca. 80h für die Klausur
  - 13 Vorlesungswochen, 12 Übungsblätter
  - Das macht ca. 10 Stunden pro Übungsblatt

Übungen

UNI FREIBURG

- Übungsblätter
  - Um an der Klausur teilnehmen zu können, brauchen Sie mindestens 50% der Punkte in den Übungsblättern
  - Wer die Übungsblätter alle macht, braucht am Ende kaum noch was zu lernen für die Klausur !
  - Abgabe / Korrektur über unser SVN ... später mehr dazu
- Übungsgruppen
  - Die Tutorinnen und Tutoren sind:
    Markus Näther, Katja Faist, Betim Musa, Mathieu Wacker, Johanna Götz, Michael Gießwein, Tobias Strickfaden
  - Assistent der Vorlesung ist: Axel Lehmann
  - Kein Übungsgruppentermin, es läuft alles **online** ... gleich mehr

Klausur / Note

UNI FREIBURG

- Die Klausur
  - ... findet statt am 28. August 2013 von 14 – 17 Uhr
  - 6 Aufgaben a 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				

Daphne

UNI FREIBURG

- Daphne ist unser **Kursverwaltungssystem**
  - Link auf dem Wiki zum Kurs, **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
  - Anmerkung: in der Vorlesung "Programmieren in Java" wird (dieses Jahr zum ersten Mal) dasselbe System verwendet

Forum

UNI FREIBURG

- Es gibt ein **Forum** zur Veranstaltung
  - Link dazu auf dem Wiki und auf Ihrer Daphne Seite
  - Bitte fragen Sie, wann immer etwas nicht klar ist
  - Nur keine Hemmungen, auch wenn Sie denken, die Frage ist blöd
  - Aber bitte möglichst konkret fragen: z.B. bei Fehlern im Programm nicht sagen "mein Programm geht nicht" ...

    Sondern Angabe von Fehlermeldung, Zeilennummer + zugehöriger Programmcode mit Zeilennummern
  - Und fragen Sie uns nicht einzeln, sondern auf dem Forum ... praktisch immer interessiert das auch andere
  - Entweder Ich oder Axel Lehmann oder einer der TutorInnen wird dann möglichst schnell antworten

SVN

UNI FREIBURG

- SVN = **Subversion** http://subversion.apache.org/
  - 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 / Ihrer Tutorin
    - Vorlesungsdateien / Musterlösungen
  - Ich werde es Ihnen heute einfach einmal vormachen ...

Coding Standards

UNI FREIBURG

- Neben dem eigentlichen Thema (AlgoDat)
  - … sollen Sie auch (weiter) **gutes Programmieren** lernen **!**
  - Dazu ein paar bewährte Standards
    - Unit Tests für alle nicht-trivialen Methoden … Junit / GTest

      Grund : Siehe nächste Folie
    - Befolgen eines Stylesheets … Checkstyle / Cpplint

      Grund: Damit ihr Code lesbar und verständlich ist
    - Standardisiertes Build-Framework Ant / Make

      Grund: Damit wir nicht bei jeder Abgabe getrennt schauen müssen, wie man den Code testet / ausführt
  - Ich werde Ihnen das heute alles einmal vormachen …
  - Ob Java oder C++ ist Ihnen überlassen !

Unit Tests

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
- Was ist ein "guter" Unit Test
  - Ein Unit Test soll überprüfen ob eine Methode für eine gegebene Eingabe, die gewünschte Ausgabe berechnet
  - Für mindestens **eine typische** Eingabe
  - Für mindestens **einen kritischen** "Grenzfall", wenn es einen solchen gibt … z.B. leeres Feld beim Sortieren

Jenkins

UNI FREIBURG

- Jenkins ist unser **Build System**
  - Damit können Sie schauen, ob Ihr Code, so wie Sie ihn bei uns hochgeladen haben, kompiliert und läuft
    - Insbesondere ob die Unit Tests alle durchlaufen
    - Und ob Checkstyle mit allem zufrieden ist
  - Werde ich Ihnen auch heute vormachen ...

Sortieren

$x < y \wedge y < z \Rightarrow x < z$

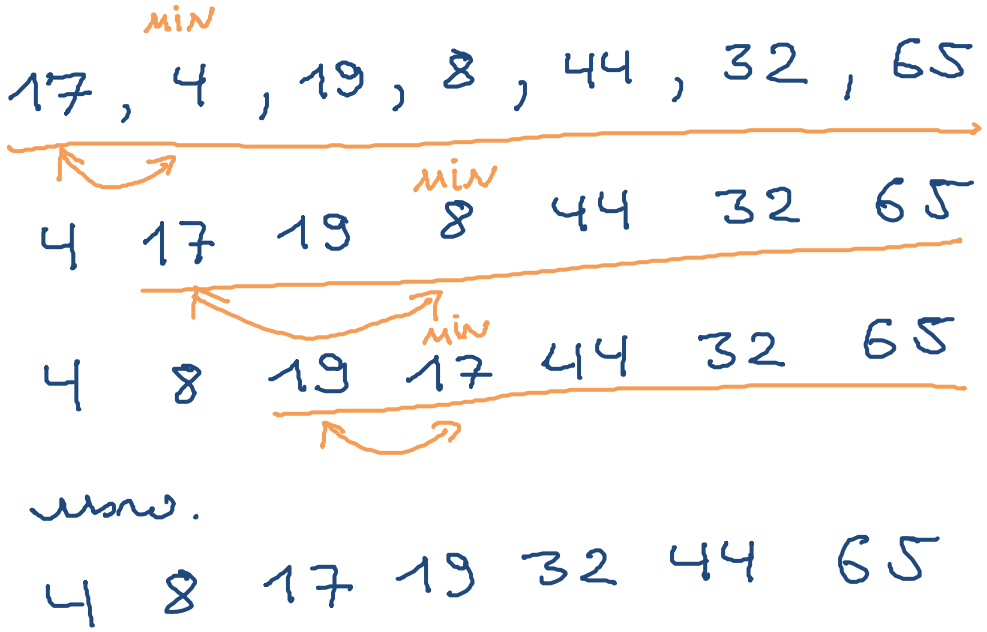
- Problemdefinition
  - **Eingabe:** eine Folge von n Elementen $x_1, \ldots, x_n$
  - Sowie ein (transitiver) Vergleichsoperator < der für zwei beliebige Elemente sagt, welches davon kleiner ist
  - **Ausgabe:** die n Elemente in gemäß diesem Operator sortierter Reihenfolge, zum Beispiel

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

    Ausgabe: 4, 8, 17, 19, 32, 44, 65
- Wo braucht man Sortieren?
  - In praktisch **jedem** größeren Programm
  - Beispiel: Bauen eines Indexes für eine Suchmaschine

MinSort — Algorithmus

- 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.



MinSort — Programm

UNI FREIBURG

- Das schreiben wir jetzt zusammen
  - Bei der Gelegenheit zeige ich Ihnen dann auch gleich wie das alles geht mit
    - den Unit Tests ... Junit / GTest
    - unseren Coding Standards ... Checkstyle / Cpplint
    - unserem Build Framework ... Ant / Make
    - unserem Build System ... Jenkins
    - unserem SVN
  - Ich mache es erst für Java vor
  - Dann zeige ich, dass es für C/C++ ganz analog geht

MinSort — Laufzeit

UNI FREIBURG

- Wie lange läuft unser Programm?
  - 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

Literatur / Links

UNI FREIBURG

- Allgemein zur Vorlesung
  - Cormen / Leiserson / Rivest: Introduction to Algorithms

    Klassisches Lehrbuch zu Algorithmen und Datenstrukturen. Nur das Inhaltsverzeichnis ist online, muss man kaufen oder ausleihen.

    http://mitpress.mit.edu/algorithms/
  - Mehlhorn / Sanders: Algorithms and Data Structures, The Basic Toolbox

    Neueres Lehrbuch zu Algorithmen und Datenstrukturen, mit viel praktischerer Ausrichtung als der Cormen/Leiserson/Rivest. Das ganze Buch steht online!

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