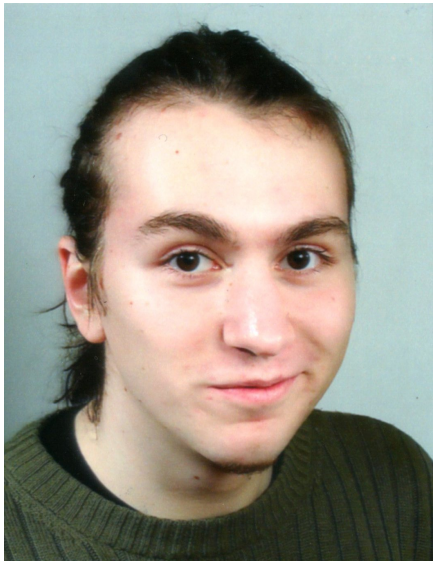
Entwurf und Analyse von Algorithmen (EAA)

Prof. Dr. Markus Nebel

WS 2014/15

Wer wird sind



~~Prof. Nebel~~
Sebastian Wild

Übungen

Prof. Nebel

Vorlesung
(ab nächster Woche)

Raphael Reitzig

Beweistechniken

Willkommen zur Vorlesung EAA im WS 14/15

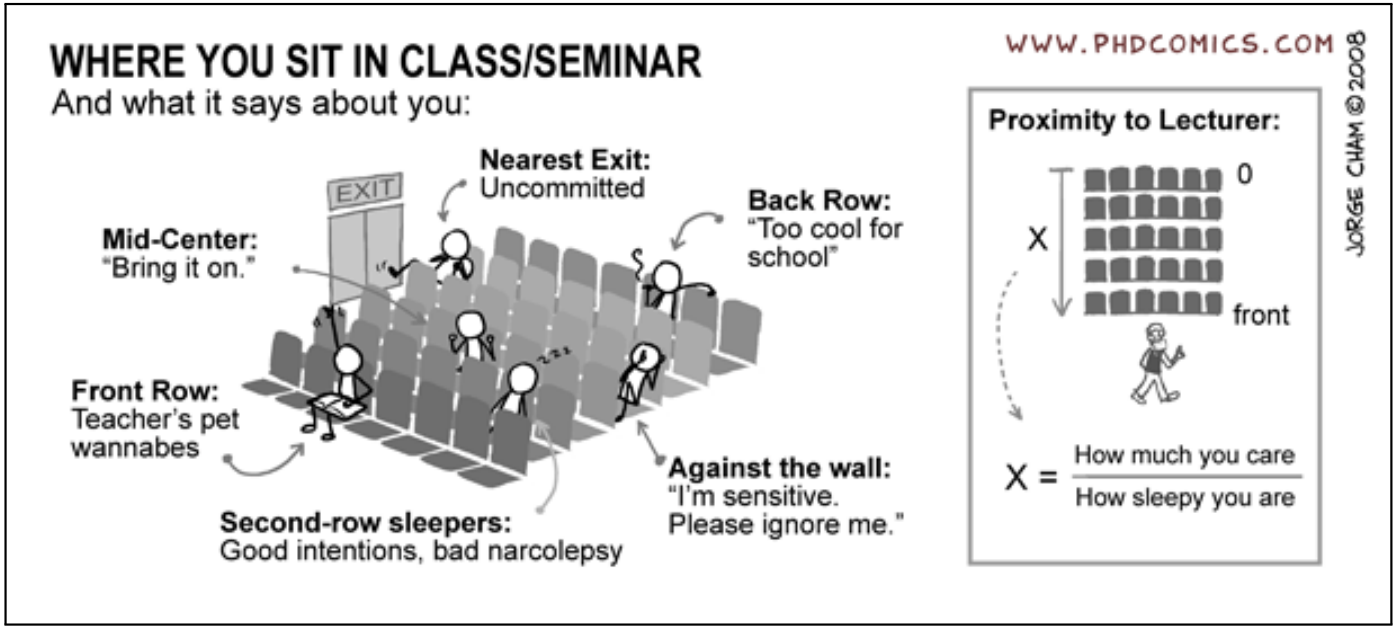
Mit Aufnahme eines Studiums wurden Sie in eine wissenschaftliche Gemeinschaft **eingeladen** und dürfen an deren Fortbestand und Weiterentwicklung mitwirken.
Die Freiheit des Studiums besteht entsprechend auch darin hinzugehen ... oder eben daheim zu bleiben!

$\Rightarrow$ Für diese Vorlesung sowie für den Übungsbetrieb besteht keine Anwesenheitspflicht, jedoch sehr wohl ein **Anwesenheitsrecht** und die Verpflichtung sich einzubringen!

Hausordnung

Uns ist wichtig, dass konzentriertes Mitdenken in der Vorlesung möglich ist. Deshalb

- ist die Verwendung von Notebooks, Tablets, Handys, Smartphones, Phablets, Smartwatches, Mobilkonsolen, ... untersagt, (die Vorlesungsumfragen der vergangenen Jahre bestärken uns in diesem Punkt);
- bitten wir darum, pünktlich zu erscheinen.

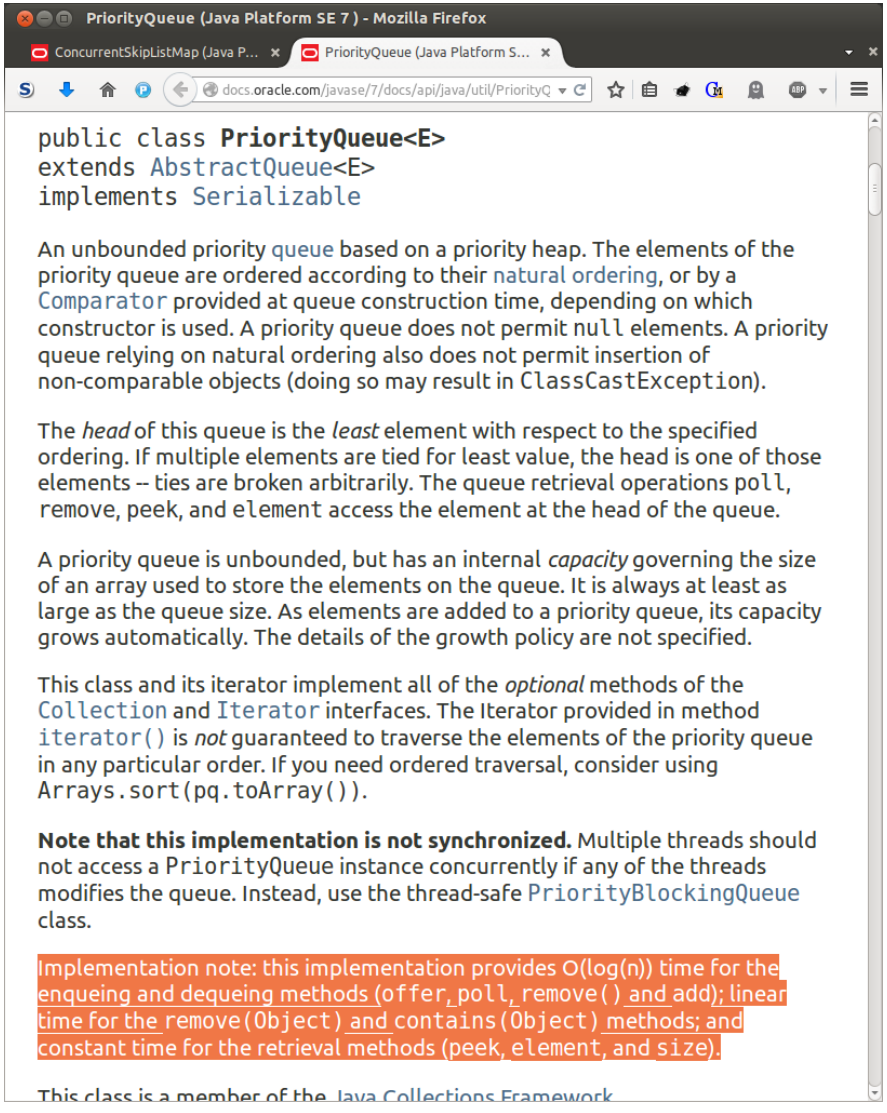


Wozu EAA?

Ziele sind

1. **Baukasten** aus Algorithmen und Datenstrukturen aufbauen,
   - *best practices:* Liste von A. & DS. für Standardprobleme
   - „hübsche“ Errungenschaften der Informatik
2. Algorithmen **bewerten** (analysieren) können,
   - $\mathcal{O}$-Notation für Laufzeit/„Komplexität“
   - manchmal genauerer Blick nötig
     - *average case* oder *worst case*?
     - *amortisiert* oder *Erwartungswert*?
3. mit anderen Experten über Algorithmen **reden** können.
   - Vokabular aufbauen ($\rightsquigarrow$ Baukasten)
   - präzise Beschreibungen verstehen und erstellen,
     z. B. Javadoc der Java API (siehe nächste Folie)
   - erkennen, wenn jemand „Mist“ erzählt

Wozu EAA? – Beispiele aus der Java API



PriorityQueue (Java Platform SE 7) - Mozilla Firefox

ConcurrentSkipListMap (Java P... | PriorityQueue (Java Platform S...

docs.oracle.com/javase/7/docs/api/java/util/PriorityQ

```
public class PriorityQueue<E>
extends AbstractQueue<E>
implements Serializable
```

An unbounded priority queue based on a priority heap. The elements of the priority queue are ordered according to their natural ordering, or by a Comparator provided at queue construction time, depending on which constructor is used. A priority queue does not permit null elements. A priority queue relying on natural ordering also does not permit insertion of non-comparable objects (doing so may result in ClassCastException).

The *head* of this queue is the *least* element with respect to the specified ordering. If multiple elements are tied for least value, the head is one of those elements -- ties are broken arbitrarily. The queue retrieval operations poll, remove, peek, and element access the element at the head of the queue.

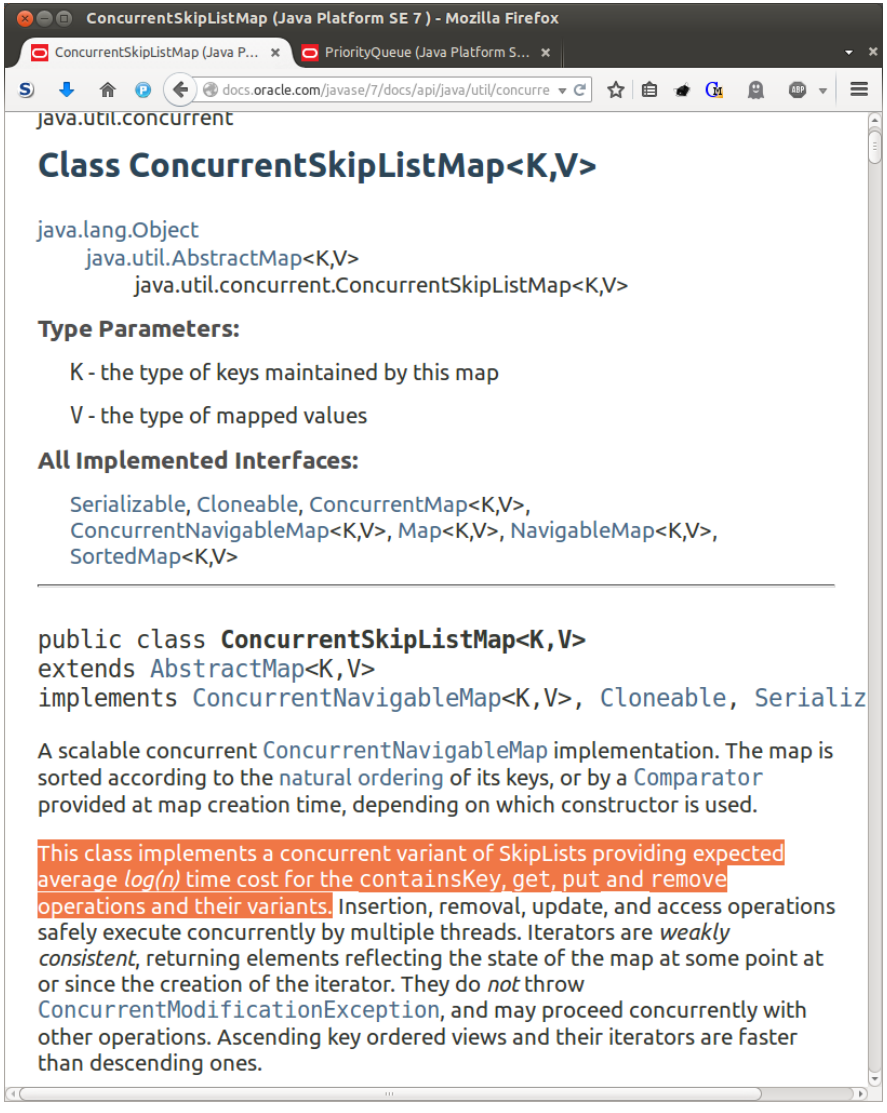
A priority queue is unbounded, but has an internal *capacity* governing the size of an array used to store the elements on the queue. It is always at least as large as the queue size. As elements are added to a priority queue, its capacity grows automatically. The details of the growth policy are not specified.

This class and its iterator implement all of the *optional* methods of the Collection and Iterator interfaces. The Iterator provided in method iterator() is *not* guaranteed to traverse the elements of the priority queue in any particular order. If you need ordered traversal, consider using Arrays.sort(pq.toArray()).

Note that this implementation is not synchronized. Multiple threads should not access a PriorityQueue instance concurrently if any of the threads modifies the queue. Instead, use the thread-safe PriorityBlockingQueue class.

Implementation note: this implementation provides O(log(n)) time for the enqueing and dequeing methods (offer, poll, remove() and add); linear time for the remove(Object) and contains(Object) methods; and constant time for the retrieval methods (peek, element, and size).

This class is a member of the Java Collections Framework



ConcurrentSkipListMap (Java Platform SE 7) - Mozilla Firefox

ConcurrentSkipListMap (Java P... | PriorityQueue (Java Platform S...

docs.oracle.com/javase/7/docs/api/java/util/concurre

java.util.concurrent

Class ConcurrentSkipListMap<K,V>

java.lang.Object
  java.util.AbstractMap<K,V>
    java.util.concurrent.ConcurrentSkipListMap<K,V>

Type Parameters:

K - the type of keys maintained by this map

V - the type of mapped values

All Implemented Interfaces:

Serializable, Cloneable, ConcurrentMap<K,V>, ConcurrentNavigableMap<K,V>, Map<K,V>, NavigableMap<K,V>, SortedMap<K,V>

```
public class ConcurrentSkipListMap<K,V>
extends AbstractMap<K,V>
implements ConcurrentNavigableMap<K,V>, Cloneable, Serializ
```

A scalable concurrent ConcurrentNavigableMap implementation. The map is sorted according to the natural ordering of its keys, or by a Comparator provided at map creation time, depending on which constructor is used.

This class implements a concurrent variant of SkipLists providing expected average *log(n)* time cost for the containsKey, get, put and remove operations and their variants. Insertion, removal, update, and access operations safely execute concurrently by multiple threads. Iterators are *weakly consistent*, returning elements reflecting the state of the map at some point at or since the creation of the iterator. They do *not* throw ConcurrentModificationException, and may proceed concurrently with other operations. Ascending key ordered views and their iterators are faster than descending ones.

Wozu EAA? – Stimmen aus der Praxis

Jörn Hees (Absolvent des Fachbereichs):
„EAA (und die AdB) haben mir schon des Öfteren geholfen; ich habe beide Skripte immer noch im Büro und schon öfters darin nachgeschaut, wenn es z.B. um die Auswahl eines Algorithmus zum Sortieren oder zur Suche ging, mich z.B. daran erinnert habe, dass darin die selbstorganisierenden Splay Trees erklärt wurden oder es einen ‚Kniff' gab um z.B. suffix-tries effizienter aufzubauen."

Robert Muth (Entwickler bei Google New York):
"Programmers today routinely write code that runs on thousands if not millions of machines, ranging from map-reduces running in the datacenter on high performance servers to mobile apps running on battery operated phones and tablets. Amplified in this way the impact of proper algorithm design on resources is enormous and it is not uncommon that a redesigned algorithm will result in savings dwarfing the salary of the programmer."

Der Spagat

Diese Vorlesung instantiiert 3 Module

1. EAA (für Studierende der Informatik),
2. EAA für Mathematiker, und
3. EAA für Angewandte Informatik (für Studierende der Angewandten Informatik, Wirtschaftsingenieure, . . .).

Alle drei haben leicht unterschiedliche Lernziele,

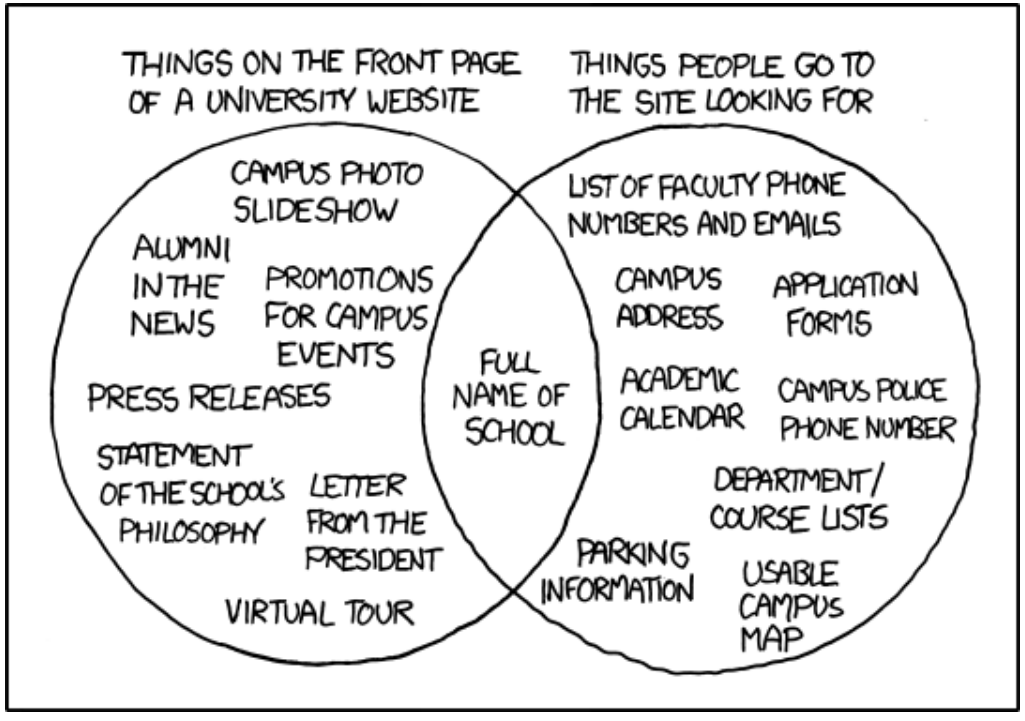
1. EAA für Informatiker ist stärker auf die mathematische Modellierung und Analyse fokussiert,
2. EAA für Angewandte Informatik bzw. Mathematiker ist mehr mit dem Algorithmenentwurf befasst (bei unterschiedlichen Voraussetzungen),

es ist jedoch nicht möglich 3 getrennte Vorlesungen anzubieten. Wir werden jedoch versuchen, diesem Umstand durch *getrennte* Übungen, Zwischen- und Abschlussklausuren gerecht zu werden.

Organisatorisches

Alle folgenden Informationen finden Sie auch auf unserer Website:

http://wwwagak.cs.uni-kl.de/Vorlesung/eaa1415.html



Übungsbetrieb – Vorgedanken

Ziele

- Vertiefung des Vorlesungsstoffes – Wissen
- Üben von Grundfertigkeiten – „Aufschreiben“
- Klausurvorbereitung – Anwendung, Transfer
- Regelmäßiges Feedback über individuellen Leistungsstand

Idee

- Angebot von verschieden gelagerten Übungsaufgaben
- Detaillierte Korrektur von eingereichten Lösungen nach Klausurmaßstäben

Übungsbetrieb – Vorgedanken

Erfahrung

1. Bedürfnisse sind unterschiedlich.
2. Freiwillige Angebote gehen zugunsten anderen Pflichten unter.
3. Gruppenabgaben können Probleme einzelner kaschieren.

Konsequenz

1. Übungsbetrieb (grob) nach Lernzielen geteilt
2. Extrinsische Motivation durch moderate Mindestpunktzahl
3. Teilweise Einzelabgaben

Übungsbetrieb – Implementierung

Aufteilung

- Track ε – Informatik, Mathematik
- Track AI – Angewandte Informatik, WI Informatik
- Andere Studiengänge – individuell (bitte melden!)

Elemente

1. Grundlagentest
2. Beweistechniken (nur Track AI)
3. Basisaufgaben
4. Aufbauaufgaben

Übungsbetrieb – Grundlagentest

- **Ziel:** Ermitteln, worauf wir aufbauen können
- **kein** Stoff der EAA selbst, sondern
  Grundlagen aus Mathematik und Softwareentwicklung
- Multiple Choice
- Ersetzt formal die Zwischenklausur
- Keine Mindestpunktzahl zum Bestehen,
  aber **Teilnahmepflicht**!

Termin: Nächster(!) Montag, 03.11, 19:00 Uhr Mensa

- Dauer: etwa 1 h
- Bitte bis Freitag **im OLAT anmelden** ($\rightarrow$ Übungsanmeldung)

Übungsbetrieb – Beweistechniken

Nur relevant für Track AI – Der Rest hat kurz Pause.

- **Ziel:** Grundlagen aus Logik und diskreter Mathematik vermitteln
  - Was ist ein Beweis? (Und was ist keiner?)
  - Wie sind Beweise aufgebaut?
  - Wie führe ich selbst Beweise?
- zwei Vorlesungen, beide diese Woche

Mi (29.10.) 15:30 Uhr in 24-102

Fr (31.10.) 08:15 Uhr in 42-110

- (vorauss.) sechs Übungsblätter

$\rightsquigarrow$ Details am Mittwoch von Raphael

Übungsbetrieb – Basisaufgaben

- **Ziele:** Wiederholung, „Aufschreiben“
- **Einzelabgaben:**
  Jede(r) gibt je Blatt eine Lösung zu **einer** Basisaufgabe ab.
- In jeder Abgabe(vierer)gruppe jede Aufgabe nur einmal.
- mindestens 18 von $13 \cdot 3$ Punkten ($\approx 45\,\%$)
- Präsentation von detaillierten Lösungen in Saalübung

Saalübung: Dienstags, 15:30 Uhr in 52-207

erster Termin: 11.11.2014

Übungsbetrieb – Aufbauaufgaben

- **Ziele:** Anwendung, Transfer
- Gemeinsame Abgabe von Lösungen in Vierergruppen.
- 20 % der Gesamtpunktzahl nötig.
- Diskussion der Lösungen in Übungsgruppen (mit Tutor)

Übungsbetrieb – Anmeldung

- Ab **heute, 16:30 Uhr**, bis **Freitag, 20:00 Uhr**!
- Anmeldung in OLAT mittels RHRK-Account auf

  https://olat.vcrp.de/url/RepositoryEntry/1232109987

  (Link ist auch auf unserer Website zu finden.)

- Anmelden für
  1. Eine **Übungsgruppe** im richtigen Track
  2. Für die **Zwischenklausur** (Grundlagentest) im richtigen Track
  3. nur Track AI: Für **Beweistechniken**
- Auf jeden Fall anmelden!
  Zulassung schon erworben $\rightarrow$ Dummy-Übungsgruppe „Keine“.

Terminübersicht

		Außerordentliche Termine	Reguläre Termine
Mo	27.10.	**Anmeldung** Übung & Grundlagentest ab 16:30 Uhr bis Fr 31.10. 20:00 Uhr	Vorlesung
Di	28.10.		
Mi	29.10.	Beweistechniken, 15:30 Uhr, 24-102	
Do	30.10.	Ausgabe erstes Übungsblatt	Vorlesung
Fr	31.10.	Beweistechniken, 08:15 Uhr, 42-110 Deadline Anmeldungen um 20:00 Uhr	
Mo	03.11.	**Grundlagentest**, 19:00 Uhr, Mensa	Vorlesung
Di	04.11.	erste Übungen	Übungen
Mi	05.11.		Übungen
Do	06.11.		Vorlesung, Übungen
Fr	07.11.	Abgabe erstes Übungsblatt	Saalübung Beweistechniken
Mo	10.11.		Vorlesung
Di	11.11.	erste Saalübung, 15:30 Uhr, 52-207	Saalübung, Übungen
Mi	12.11.		Übungen
Do	13.11.		Vorlesung, Übungen
Fr	14.11.		Saalübung Beweistechniken

Literatur

Die über mehrere Jahre hinweg gewachsenen Materialien zur Vorlesung EAA wurden erneut überarbeitet, um interessante Inhalt ergänzt und als Buch veröffentlicht.

Markus Nebel
Entwurf und Analyse von Algorithmen
Springer Vieweg Verlag

- Grundlage dieser Vorlesung
- In der Lehrbuchsammlung der Zentralbibliothek verfügbar.
- Kostenlos als eBook aus dem Uni-Netz (siehe Vorlesungswebsite)
- Weitere Literatur auf unserer Website

Aus dem Inhalt:

1. Einführung in die Algorithmentheorie (`While`-berechenbare Funktionen);
2. (Mathematische) Grundlagen (asymptotische Notationen, amortisierte Kosten, Rekursionsgleichungen, abstrakte Datentypen);
3. Elementare Datenstrukturen (Lineare Listen, Stacks und Queues, Mengen, Graphen und Bäume);
4. Wörterbücher (Baumstrukturen [BSTs, balancierte Suchbäume, Splay-Trees,...], Hashing);
5. Graphalgorithmen (kürzeste Wege, Spannbäume);

Aus dem Inhalt (Forts.):

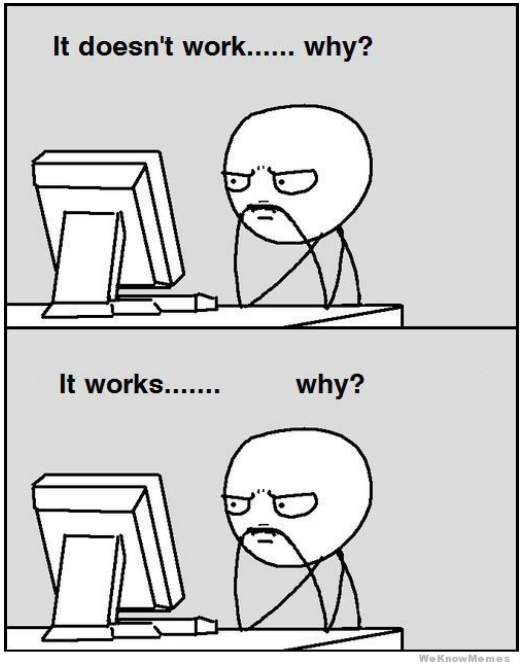
6. Sortieren (primitive Sortieralgorithmen, Quicksort, Heapsort, ..., Sortiernetzwerke);
7. Entwurfsmethoden für Algorithmen (Divide and Conquer, Dynamisches Programmieren, Greedy, Lineares Programmieren);
8. Komplexitätstheorie ($\mathcal{NP}$-Vollständigkeit, Auswahl $\mathcal{NP}$-vollständiger Probleme);
9. Schwere Optimierungsprobleme (Branch & Bound, Approximationsalgorithmen, Randomisierte Algorithmen);

Doch kommen wir endlich zu den Inhalten selbst ...

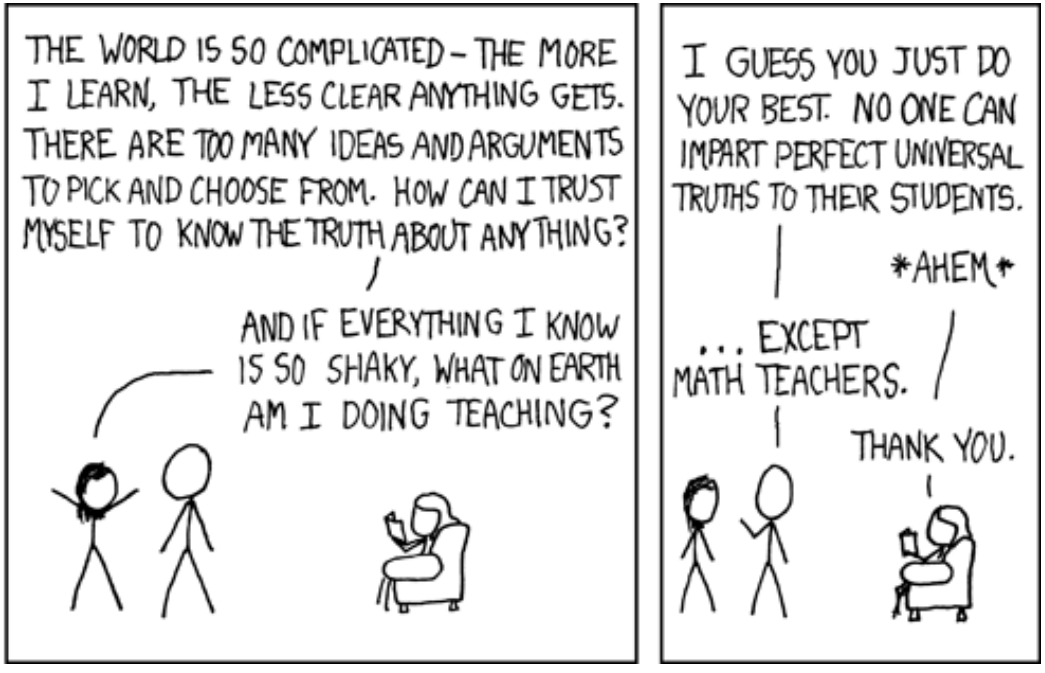
Ein Hinweis

Die EAA ist Teil eines **wissenschaftlichen** Studiengangs (siehe Akkreditierung). Daher erwarten wir ein gewisses Maß an Verständnis und Fähigkeiten im Umgang mit Konzepten der Modellbildung und Analyse!

Weniger . . .



. . . und mehr



⇝ Fokus auf “universal truths” der Algorithmik.

⇝ Mathematische/Theoretische Grundlagen nötig!