



# Algorithm Engineering Sommersemester 2010 Universität Bremen

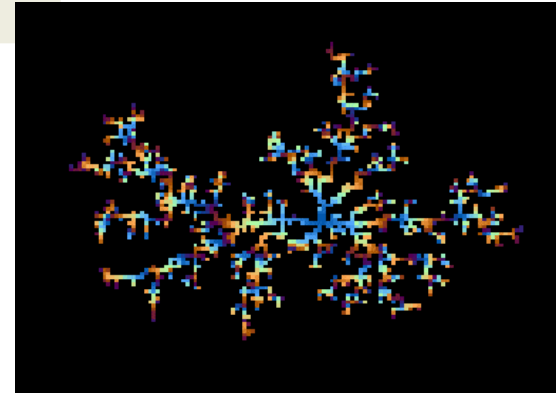
Stefan Edelkamp

# Algorithm Engineering = Algorithmen Ingenieur

- ▶ Werden Sie **Dipl. Ing. Alg.**



# Struktur



## Inhalte

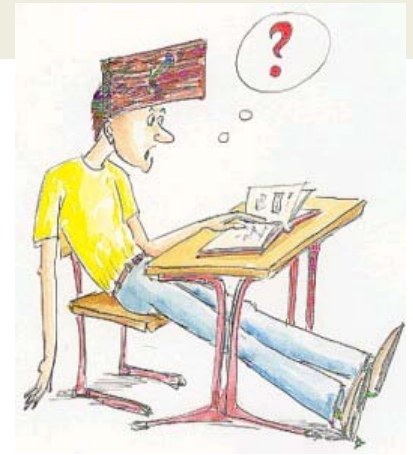
- ▶ Ansatz, grundlegende Methodologie
- ▶ Schnelles Sortieren
- ▶ (Worst-Case) Effiziente Prioritätslisten
- ▶ Perfekte Hash-Funktionen
- ▶ Konstruktion von Suffix Bäumen
- ▶ Festplattenalgorithmen
- ▶ Externes Suchen und Sortieren inkl. unterer Schranken
- ▶ Flashspeicher- und GPU-Algorithmen

## ▶ Formales

- 6 ECTS
- Termine:  
Mittwoch 10-12 Uhr (V)  
Do. 12-14 (Ü)  
TAB 1.50

## Modul

602 (Algorithmen- und Komplexitätstheorie)



# Übungen / Prüfungen / Web

- ▶ Übungszettel alle 14 Tage (erste Stunde: 22. April 2010)
- ▶ Übung wechselt mit Vorlesung (n.V.)
- ▶ Theoretisch mit praktischen Teil
- ▶ Globalübung
- ▶ Prüfungen mündlich (Modulprüfung & Fachgespräch)
- ▶ Termine nach Vereinbarung zum Ende des Semesters
- ▶ Internet-Präsenz  
[www.tzi.de/~edelkamp/lectures/ae](http://www.tzi.de/~edelkamp/lectures/ae)

# Vorlesung vs. Besprechung

## Vorlesungsskript



- ▶ Anstatt reinem Folienvortrag können wir die im Netz bereitgestellten Folien in der Vorlesungszeit auch besprechen.
- ▶ Es gibt eine Skript zur Vorlesung in Form einer Kopiervorlage.



# Anrede



91%

-  Sietzen
-  Dutzen



# Vorlesungsausfall

Es gibt nichts Ergreifenderes  
im Leben als einem kleinen  
Menschen das erste Mal die  
Hand zu reichen und zu spüren  
dass wir seine Wurzeln im Baum  
des Lebens sind, die ihm Halt  
und Geborgenheit geben.



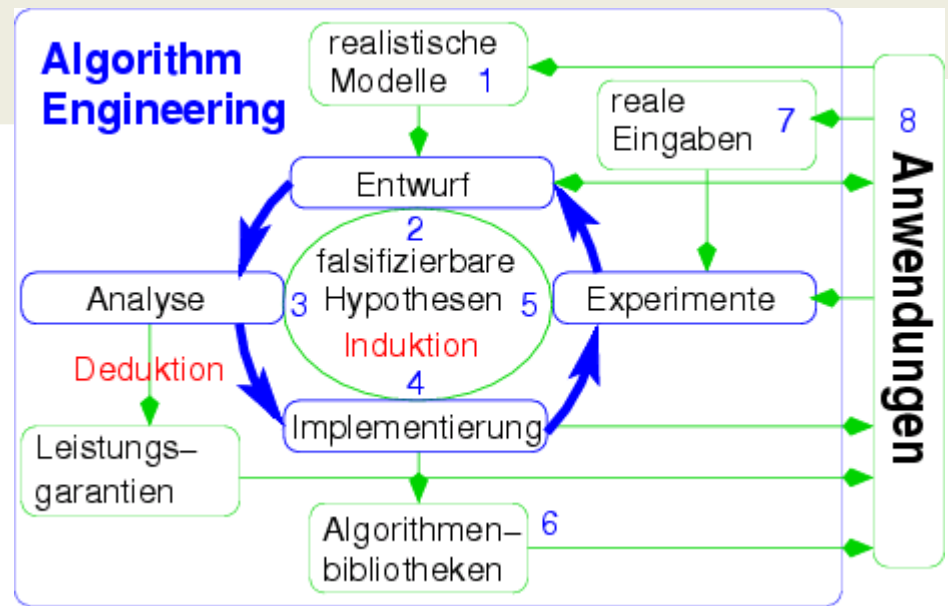
# Laptops



# Laptops



# Thema



Im Mittelpunkt von Algorithm Engineering steht ein von falsifizierbaren Hypothesen getriebener Kreislauf aus Entwurf, Analyse, Implementierung, und experimenteller Bewertung von praktikablen Algorithmen.

Realistische Modelle, für Algorithmenbibliotheken und Sammlungen realer Eingabeinstanzen erlauben eine zusätzliche Kopplung an Anwendungen

**Bruder Jakob**  
Volkslied aus Frankreich (1870)

The image shows a musical score for the song 'Bruder Jakob'. It consists of three staves of music in G major, 3/4 time. The first staff has a treble clef and a key signature of one sharp (F#). Above the staff are the notes F, C, F, F, C, F. The lyrics are 'Bru - der Ja - kob, Bru - der Ja - kob, schläfst du noch,'. The second staff continues the melody with lyrics 'schläfst du noch? Hörst du nicht die Glo-cken, hörst du nicht die Glo-cken?'. The third staff has lyrics 'Ding, dong, dong, ding, dong, dong!'. The score is marked with first, second, third, and fourth endings.

# Alles Kanonisch?

Auf den ersten Blick gleichen die Themen dieser Vorlesung denen einer "kanonischen" Algorithmenvorlesung.

Allerdings geht es hier nicht um die Vermittlung der Grundideen, sondern um größtmögliche praktische Effizienz.

Erstaunlicherweise ist das immer noch ein aktuelles Forschungsthema.

# Zwei Gründe

Reale Maschinen haben sich weit vom einfachen von Neumann-Modell entfernt.

Insbesondere Speicherhierarchien und parallele Befehlsausführung machen das bloße Zählen von Befehlen ungenau.

Die Algorithmen-Theorie hat faszinierende Techniken erfunden, die z.T. als nicht implementierbar gelten.

Durch die Weiterentwicklung dieser Techniken lassen sich solche Lücke zwischen Theorie und Praxis manchmal überwinden.

# Theoretisch oder Praktisch?

- ▶ AE das Zugeständnis, dass aktuelle Rechnerentwicklungen nach dem Auslauf des von-Neumann Modells neue Komplexitätsmodelle, Algorithmenentwürfe und Analysen bedürfen.
- ▶ Der Kurs richtet sich an den gleichnamigen Veranstaltungen von Peter Sanders, Ulrich Meyer und Petra Mutzel und ist an den dortigen Universitäten eindeutig in den theoretischen Curricula verankert

# Theoretisch oder Praktisch?

- ▶ In Deutschland verbindet AE die führenden Theoretiker des Landes, siehe [www.algorithm-engineering.de](http://www.algorithm-engineering.de)
- ▶ Auch Jyrki Katajainen, Thomas Ottmann, Martin Dietzfelbinger, Peter Widmayer, Rolf Klein, Kurt Mehlhorn, Emo Wetzl u.v.a.m. sind Vertreter der Verbindung von effizienten Algorithmen auf der einen und komplexitäts-theoretischen Betrachtungen auf der anderen Seite

# Theoretisch oder Praktisch?

- ▶ Keine Seite ist mehr oder weniger "theoretisch", eher das Gegenspiel von Gut und Böse. Eine lange Zeit wurde gern von "Algrithmentheorie" gesprochen.
- ▶ Natürlich muss sich die Theorie immer rechtfertigen, dass sie mit ihren Modellen zur Beschreibung der Phänomene in der Praxis noch die richtigen sind, um verändernd einzugreifen

# Theoretisch oder Praktisch?

Welche formalen

Beschreibungsmethoden werden eingesetzt?

- ▶ Die jeweils passenden. Wie im RAM Modell zur sequentiellen Berechnung und dem PRAM Modell zur parallelen Berechnung werden bei Festplattenalgorithmen die Blockzugriffe gezählt.
- ▶ So bedürfen effiziente Algorithmen häufig der amortisierten Analyse.

Welche Arten von Theoremen werden mit welchen Methoden bewiesen?

- ▶ Korrektheitsbeweise sowie Laufzeitanalysen z.B. im I/O-Komplexitätsmodell von Vitter und Shriver...
- ▶ ... gepaart mit einigen Unmöglichkeitensanalysen und unteren Schranken zum Beispiel für das externe Sortieren.

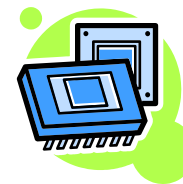
# Algorithmen größtmöglicher Effizienz

- ▶ Für das Sortieren (großer Datenmengen)?
- ▶ Für Prioritätslisten (Vorangwarteschlangen)?
- ▶ Für Hashing (großer Datenmengen)?
- ▶ Für Graphsuche (Puzzleprobleme, kürzeste Wege)?
- ▶ Für Zeichenkettensuche (Substringsuche)?
- ▶ Für SLP (Studi-Lieblingsproblem)?

# Neue Hardware



- ▶ Die Vorlesung ist ausgerichtet auf Algorithmen für moderne PC-Architekturen mit mehreren Berechnungseinheiten, hierarchisch organisiertem Prozessorcaches und externe Medien, wie riesige Festplatten und Flashspeicher.





# Veranstalter



## Werdegang

- ▶ **TU Dortmund** (Sortieren, DA)
- ▶ **U Freiburg** (Suchen & Planen, Promotion & Habilitation)
- ▶ **TU Dortmund** (MC & KI, Nachwuchsgruppe)
- ▶ **U Bremen** (Sicherheit & KI)

## Weitere Interessen:

Sortieren & Suchen  
insbesondere mit aktueller  
Hardware (HDD, SSD, GPU)

## Projektgruppen

- ▶ **GPS-Route**  
(automatische Kartengenerierung & Routing auf GPS-Spuren)
- ▶ **Mod-Plan**  
(Wissensmodellierung für die Handlungsplanung)
- ▶ **Bug-Finder**  
(Automatische Fehlersuche in Programmen)
- ▶ **FIDIUS** (IDS mit KI)



# Betreuer

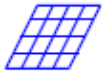







## Werdegang

- ▶ **TU Dortmund**  
(Sequenzalignierung,  
Algorithm Engineering)
- ▶ **U Bremen**  
(Handlungsplanung,  
Allgemeine Spiele)

## Preise u.a.

- ▶ **Amtierender Planungsweltmeister** im deterministischen und nicht-deterministischen Planen



Theorie		↔	Praxis	
einfach		<b>Problemmodell</b>		komplex
einfach		<b>Maschinenmodell</b>		real
komplex		<b>Algorithmen</b>	<code>FOR</code>	einfach
fortgeschr.		<b>Datenstrukturen</b>		Felder,...
worst case	<code>max</code>	<b>Komplexitätsmaß</b>		Eingaben
asympt.	<code>O(.)</code>	<b>Effizienz</b>	<code>42%</code>	Konstanten

# Historie der Algorithmik

## 50er/60er: Pionierzeit

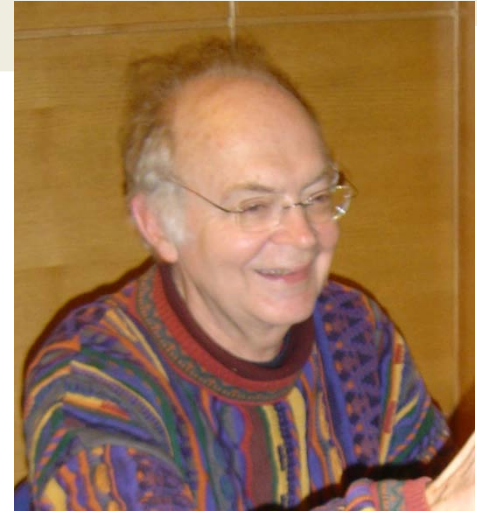
Pioniere der Algorithmik (Knuth, Floyd,...) gaben Implementierungen an

aber selten: Tests oder Vergleiche

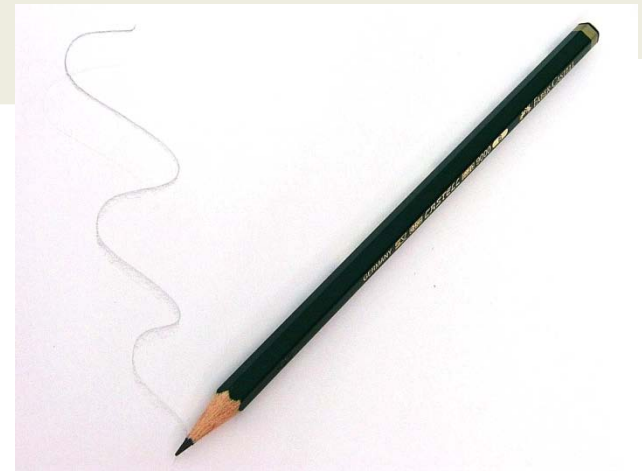
Knuth:

„Beware of bugs in the above code;

I have only proved it correct, not tried it.“



# 70er/80er: Papier-und Bleistift Jahre



Algorithmen werden nie implementiert

Viele Schichten komplexer, unimplementierter Algorithmen und Datenstrukturen

Sehr abstrakte Beschreibungen auf hohem Level

Fehler werden nicht entdeckt

→ viele nicht-korrekte Algorithmen werden publiziert, z.B. Planare Einbettung bei Planaritätstest, Hopcroft & Tarjan 1974, Mehlhorn 1982

# Lakonisch

“If you don’t make mistakes,  
you’re not working on hard  
enough problems” [F. Wikzek]

Algorithmentheorie entfernt sich immer weiter  
von der Praxis!

Hinwendung zu praxisfernen Problemen

Rückzug in wissenschaftlichen „Elfenbeinturm“

Implementierung = „Finger schmutzig machen“

# 80er: Erste experimentelle Vergleiche

Bentley 1983: „Programming Pearls“

Jones 1986: Exp. Vergleich von

Datenstrukturen für

Prioritätswarteschlangen

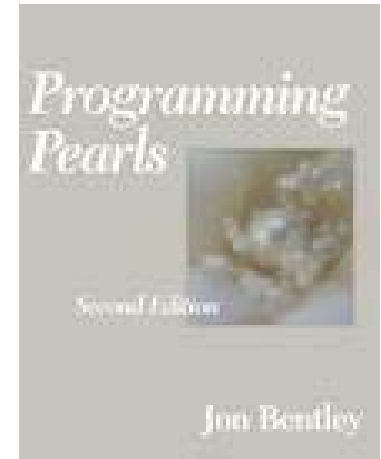
Stasko & Vitter 1987: Pairing heaps

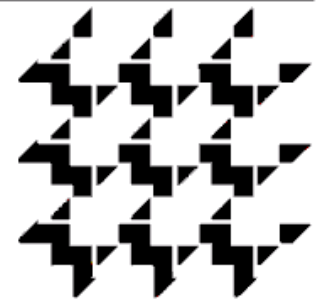
Aber: viele Experimente sind unzureichend:

Es existieren keine Benchmark-Bibliotheken →

Instanzen bei denen man selbst gut abschneidet

Selten Vergleiche mit „state-of-the-art“ Algorithmen





## 90er: Beginn des AE

Johnson 1991: Erster DIMACS Computational Challenge: network flow & shortest path

→ Benchmark Libraries, Datenformate, beste Verfahren

- Moret & Shapiro 1991: Sortierverfahren, 1994: MST
- Cherkassky, Goldberg, Radzig 1996: Kürzeste Wege
- Mehlhorn & Näher: Beginn von LEDA: C++-Library

**Pioniere:** Goldberg, Johnson, Karp, McGeoch, Mehlhorn, Moret, Orlin, Pevzner

Seit 1996: ACM Journal of Experimental Algorithmics (JEA)

Seit 1997: Workshop on Algorithm Engineering  
(WAE, Italiano) → European Symposium on Algorithms /  
Applied Track

Seit 1999: Workshop on Algorithm Engineering and  
Experiments (ALENEX) → co-located mit ACM Symposium  
on Discrete Algorithms (SODA)

Seit 2001: Workshop on Experimental Algorithmics  
ACM Symposium on Computational Geometry: Applied  
Track (inzwischen integriert)

Viele Konferenzen haben mehr „Praxis“ im „Call for Papers“

## 90er: Beginn des AE

Aho, Johnson, Karp, Kosaraju, McGeoch, Papadimitriou, Pevzner 1996 (seminal NSF-paper): „Emerging Opportunities for Theoretical Computer Science“:

„Efforts must be made to ensure that promising algorithms discovered by the theory community are implemented, tested and refined to the point where they can be usefully applied in practice.“

“...to increase the impact of theory on key application areas.“

# Ende der 90er: Rechnerarchitektur

LaMarca & Ladner 1996: „Cache-Optimierung ist machbar, algorithmisch interessant, lohnenswert (auch für Sortierverfahren)

Vitter 2001: External Memory Modelle und Algorithmen: tiefere Speicherhierarchien als vor 40 Jahren, weit größere Datenmengen





# Einige Erfolgsstories



Erfolgreiche Algorithmenbibliotheken:

- LEDA, CGAL, AGD, CPLEX, ABACUS

Viele vergleichende experimentelle Studien über „beste“ Algorithmen und Datenstrukturen

- Z.B. Prioritätswarteschlangen, Suchbäume, Hashtabellen, TSP, MST, kürzeste Wege, Konvexe Hülle, Delaunay Triangulierung, Matching, Flüsse)

Große Instanzen NP-schwerer Probleme gelöst (TSP, Set Cover,...)

Spektakuläre Speed-Ups über „Everyday“ Code

# Aufgaben des AE

1. Studium von realistischen Modellen für algorithmische Probleme.
2. Studium von realistischen Modellen für realistische Maschinen.
3. Entwurf von einfachen und auch in der Realität effizienten Algorithmen.

# Aufgaben des AE

4. Analyse praktikabler Algorithmen zwecks Etablierung von Leistungsgarantien, die Theorie und Praxis näher bringen.
5. Implementierungen, die Lücken zwischen bestem theoretischen Algorithmus und bestem implementierten Algorithmus verkleinern.
6. Systematische, reproduzierbare Experimente, die der Widerlegung oder Stärkung aussagekräftiger, falsifizierbarer Hypothesen dienen.

# Aufgaben des AE

7. Entwicklung und Ausbau von Algorithmenbibliotheken, die Anwendungsentwicklungen beschleunigen und algorithmisches Know-how breit verfügbar machen.
8. Sammeln und verfügbar machen von großen und realistischen Probleminstanzen sowie Entwicklung von Benchmarks.
9. Einsatz von algorithmischem Know-how in konkreten Anwendungen.



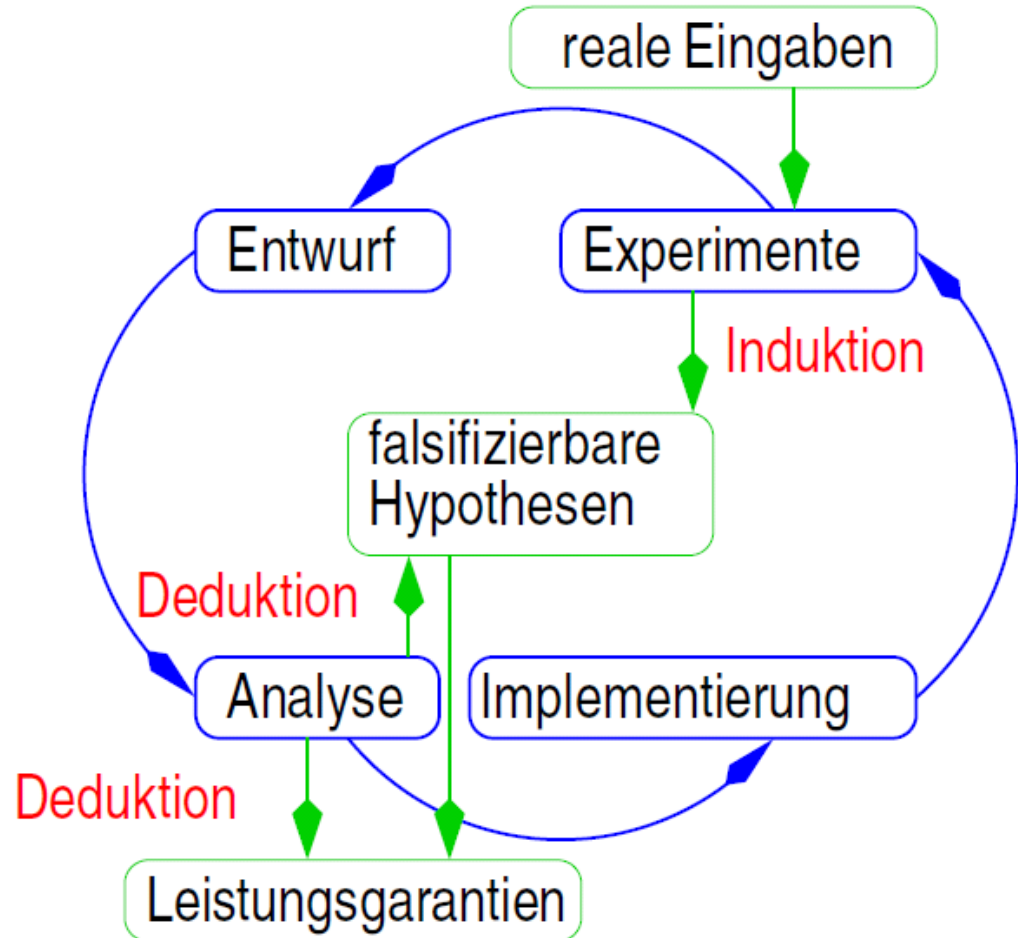
# Algorithm Engineering

## Scientific Method

Das Modell

der Naturwissenschaften

[Popper 1934]



# Ziele

- Entstandene **Lücken überbrücken**
- Schneller **Transfer** algorithmischer Ergebnisse in **Anwendungen**
- Theorie trifft Technologie —  
**Maschinenmodelle** müssen der technologischen Entwicklung gerecht werden



# Grundlegende Algorithmen

- Listen, Arrays, Stacks, Queues
- Sortieren
- Prioritätslisten
- Sortierte Listen/Suchbäume
- Hashtabellen
- Graphenalgorithmen
  - Graphtraversierung (DFS, BFS)
  - Kürzeste Wege
  - Minimale Spannbäume
  - Flussprobleme
- Strings

# Inhalte

- ▶ Schnelles Sortieren z.B. mit Quick- und Weak-Heapsort
- ▶ Cache- und Worst-Case Effiziente Prioritätslisten z.B. mit Relaxed Weak Queues
- ▶ Perfekte Hash-Funktionen z.B. zur Kompression von Daten
- ▶ Strings: Konstruktion von Suffix Bäumen und Arrays
- ▶ Festplattenalgorithmen: Externe Such- und Spannbäume, Graphsuche z.B. BFS mit „Delayed Duplicate Detection“
- ▶ Flashspeicheralgorithmen: Schnelles Lesen auf der Solid-State-Disk
- ▶ GPU-Algorithmen: Parallele Algorithmen auf der Grafikkarte

# Inhalte

- ▶ Schnelles Sortieren z.B. mit Quick- und Weak-Heapsort
- ▶ Cache- und Worst-Case Effiziente Prioritätslisten z.B. mit Relaxed Weak Queues
- ▶ Perfekte **Hash-Funktionen** z.B. zur Kompression von Daten
- ▶ Strings: Konstruktion von Suffix Bäumen und Arrays
- ▶ **Festplattenalgorithmen**: Externe Such- und Spannbäume, Graphsuche z.B. BFS mit „Delayed Duplicate Detection“
- ▶ Flashspeicheralgorithmen: Schnelles Lesen auf der Solid-State-Disk
- ▶ GPU-Algorithmen: Parallele Algorithmen auf der Grafikkarte

# Echtzeit Hash-Tabellen

Einfügen, löschen in konstanter erwarteter Zeit

Suchen in konstanter Zeit, worst case

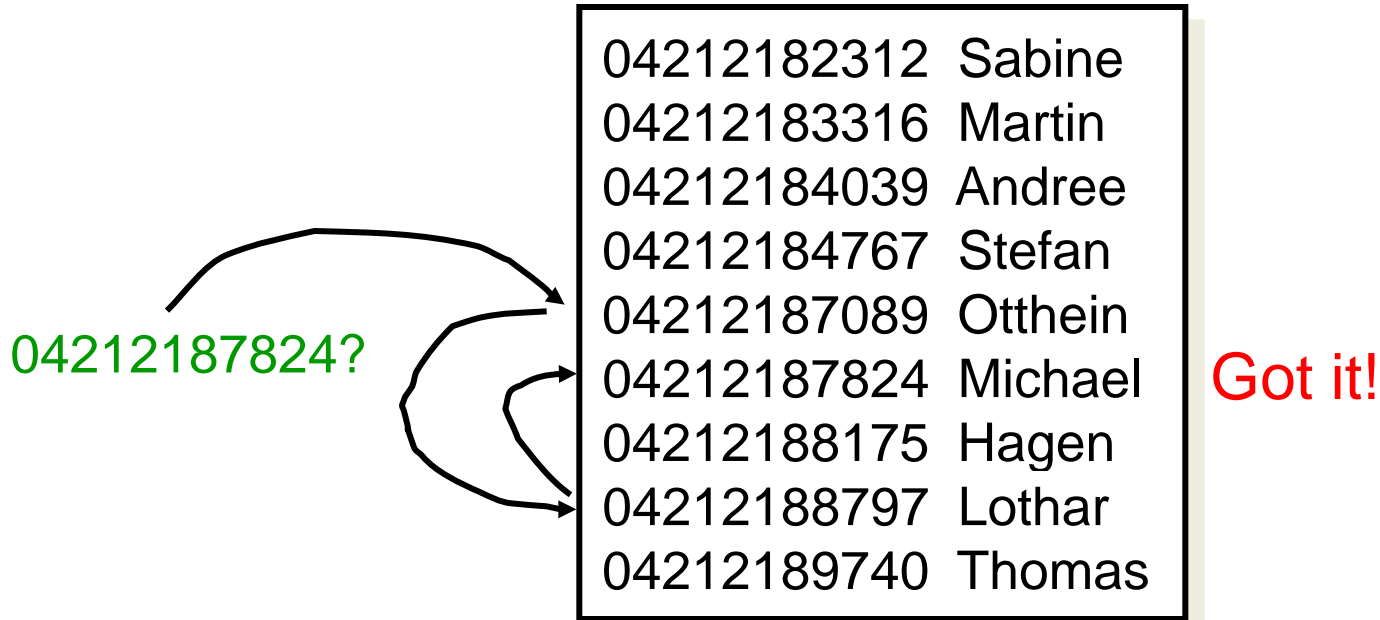


Theorie: [\[Dietzfelbinger/Karlin/Mehlhorn/MadH/Rohnert/Tarjan 94\]](#)

langsam, kompliziert, platzaufwendig

# Motivation: Finden einer Telefonnummer

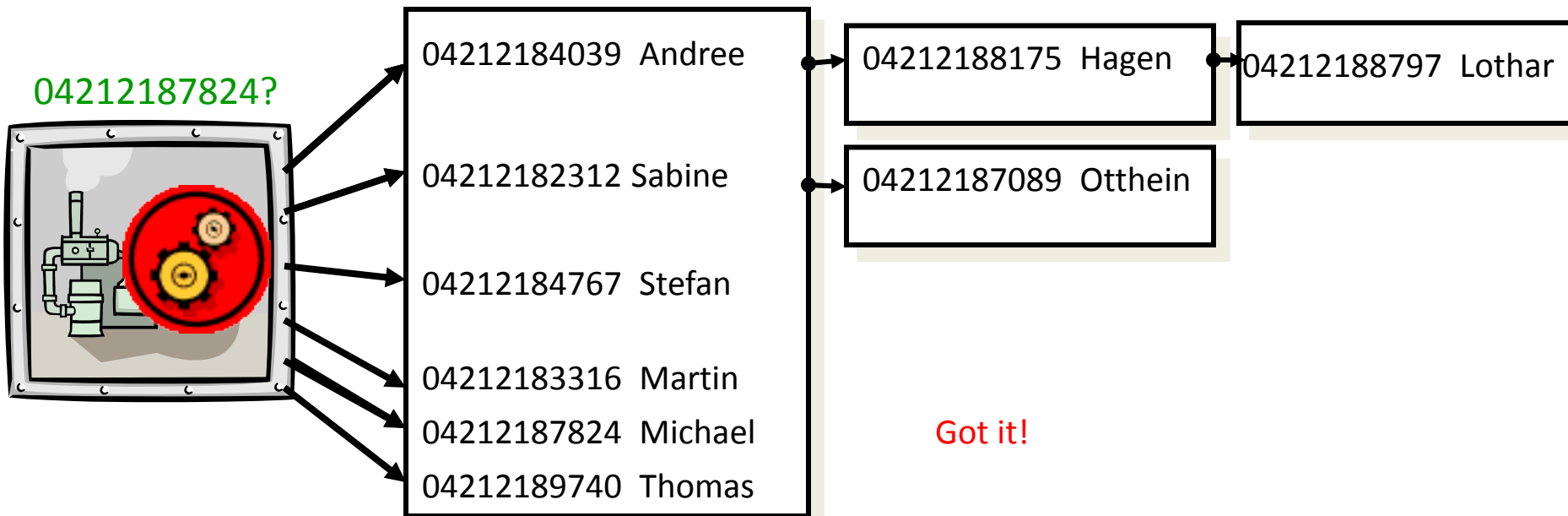
a) In einer sortierten Liste



Anzahl der Operations *steigt* mit der Anzahl der Einträge.

# Motivation

## b) In einer Hashtabelle



Nutzen einer "Hashfunktion" um Adressen zu generierung und zu merken

# Cuckoo Hashing

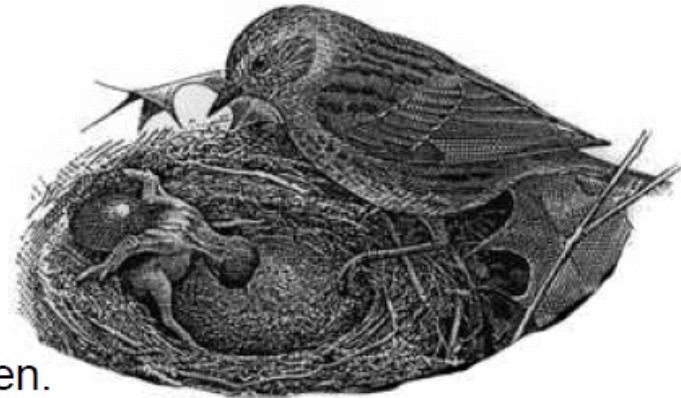
[Pagh Rodler 01]

Tabelle der Größe  $(2 + \epsilon)n$ .

**Zwei** Optionen für jedes Element.

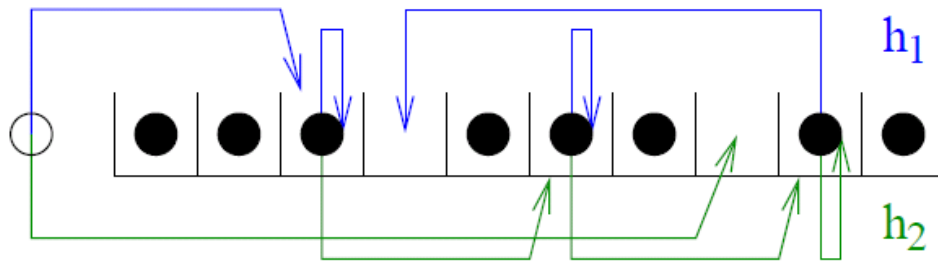
Einfügen verschiebt Elemente;

Neuaufbau falls nötig.



Sehr schnelles Einfügen und Löschen.

Konstante erwartete Einfügezeit.



# Cuckoo Hashing

**Suche:** Die Hashfunktion gibt zwei Möglichkeiten aus.

04212188175 Hagen

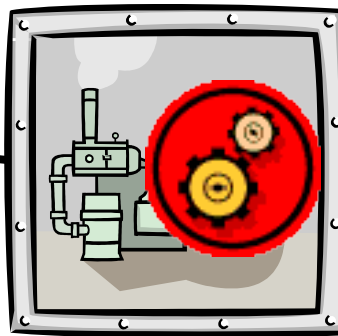
Nicht hier

04212183316 Martin

04212188797 Lothar

04212184039 Andree

04212187824?



Got it!

04212187824 Michael

04212187089 Otthein

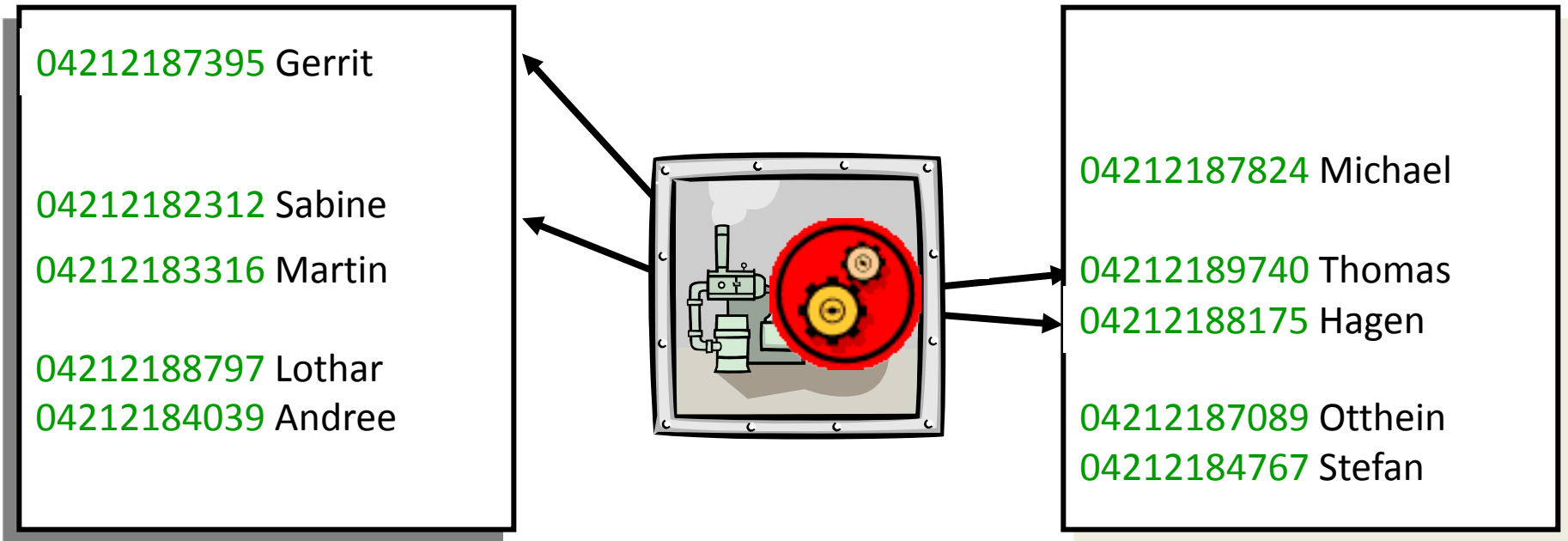
04212182312 Sabine

04212189740 Thomas

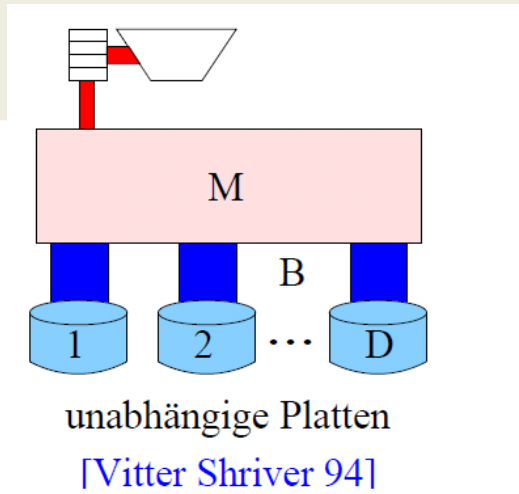
04212184767 Stefan

# Cuckoo Hashing

**Insert:** Neue Information wird eingefügt; falls notwendig, wird die alte Information herausgeschoben.



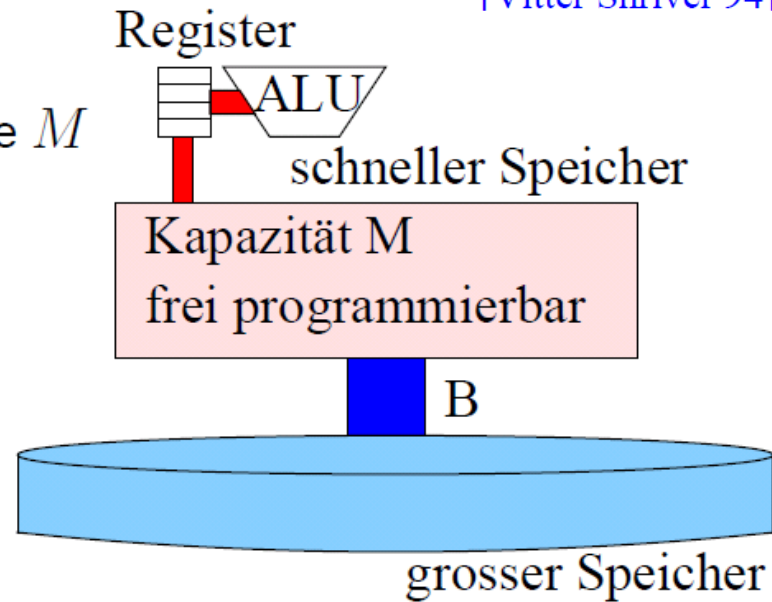
# Das Sekundärspeichermodell



$M$ : Schneller Speicher der Größe  $M$

$B$ : Blockgröße

**Analyse:** Blockzugriffe zählen

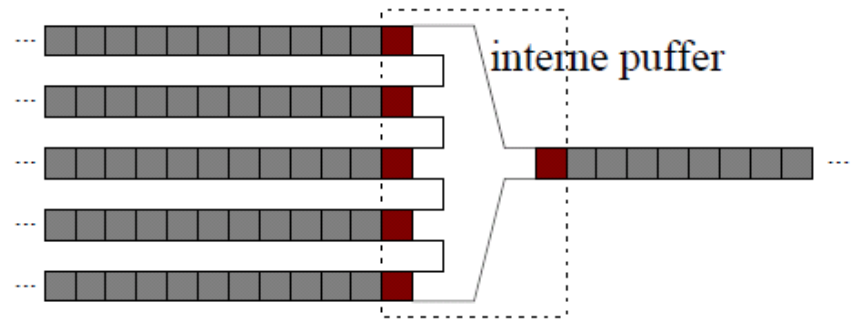
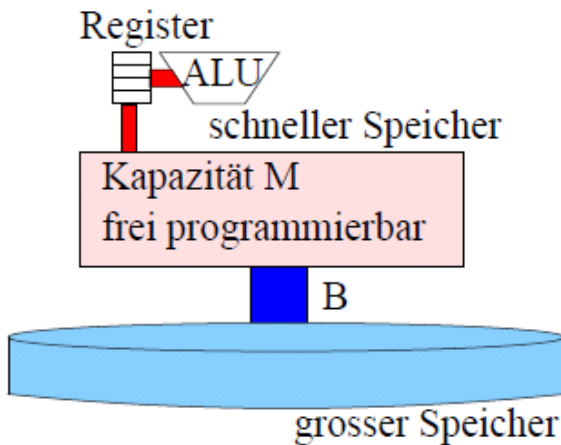


# Sortieren durch Mehrwegemischen

- Sortiere  $\lceil n/M \rceil$  runs mit je  $M$  Elementen  $2n/B$  I/Os
- Mische je  $M/B$  runs  $2n/B$  I/Os
- bis nur ein run übrig  $\times \lceil \log_{M/B} \frac{n}{M} \rceil$  Mischphasen

Insgesamt

$$\text{sort}(n) := \frac{2n}{B} \left( 1 + \left\lceil \log_{M/B} \frac{n}{M} \right\rceil \right) \text{ I/Os}$$



## **Literatur zur Vorlesung**

Skript ~ 3. Kapitel (Datenstrukturen, Externe & Parallele Suche) von 22 aus eigenem Buchprojekt  
Aktuelle Veröffentlichung in Konferenzbänden und Zeitschriften der KI, der Verifikation und des Algorithm Engineering.

## **Protagonisten der Szene**

Ulrich Meyer (Univ. Frankfurt), Peter Sanders (Univ. Karlsruhe), Lars Arge (Univ. Aarhus), Kurt Mehlhorn (MPI Saarbrücken), Eric Demaine (MIT)

...

- ▶ Baumann, Hannes 8746 Bayro Kaiser, Esteban 8920 Berndt, Jan Ole 8177 Boronowsky, Michael 7272 Breckenfelder, Christof 3837 Edelkamp, Stefan 4676 Elfers, Carsten 7618 Gehrke, Jan 7828 Gottfried, Björn 7832 Herzog, Otthein 7089 Hoffmann, Peter 7281 Horstmann, Mirko 7835 Iben, Hendrik 2447 Jacobs, Arne 9135 Kalkbrenner, Gerrit 7395 Kemnade, Andreas 8919 Kissmann, Peter 7695 Landau, Veronika 2894 Langer, Hagen 8175 Lawo, Michael 7824 Leibrandt, Rüdiger 8919 Lüdtker, Andree 4039 Mania, Patrick 7282 Messerschmidt, Hartmut 7840 Mathews, Antje 7090 Meyer-Lerbs, Lothar 8797 Modzelewski, Markus Möhlmann, Daniel 9135 Nadin, Michai 4311 Nicolai, Tom 8172 Pantke, Florian 7475 Schröder, Marcus 8950 Schuldt, Arne 8176 Sohr, Karsten 7618 Sprado, Jörn 4781 Stommel, Martin 3316 Veit, Sabine 2312 Wagner, Thomas 9740 Warden, Tobias 8614 Wewetzer, David 3579 Wojtusiak, Janusz 8175 Woronowicz, Tanja 7829 Xing, Xin 3035