Effiziente Algorithmen

Sommersemester 2019

Effiziente Algorithmen

Sommersemester 2019

Dozent: Martin Hoefer (Raum 115)

Übungen: Daniel Schmand (Raum 114)

Webseite: http://algo.cs.uni-frankfurt.de/
→ Lehre - Sommer 2019
→ Effiziente Algorithmen
(→ hier)

Übungen

- Ausgabe Blatt x: Dienstag in Woche i auf der Webseite
- Abgabe Blatt x: Bis Dienstag in Woche $i + 1$ **bis 10:15 Uhr** entweder **in der Vorlesung** oder **vorher im Briefkasten im 1. Stock**, RMS 11-15.

Übungen

- Ausgabe Blatt x: Dienstag in Woche i auf der Webseite
- Abgabe Blatt x: Bis Dienstag in Woche $i+1$ **bis 10:15 Uhr** entweder **in der Vorlesung** oder **vorher im Briefkasten im 1. Stock**, RMS 11-15.

- Korrektur, Rückgabe, Besprechung Blatt x in Woche $i+2$
- Tutoren und Termine:
  Martin Ludwig (martin.j.ludwig@gmx.de)
  Di 14-16h, SR 011

  Conrad Schecker (conrad.schecker@gmail.com)
  Mi 14-16h, NM 113

Übungen

- Ausgabe Blatt x: Dienstag in Woche i auf der Webseite
- Abgabe Blatt x: Bis Dienstag in Woche $i+1$ **bis 10:15 Uhr** entweder **in der Vorlesung** oder **vorher im Briefkasten im 1. Stock**, RMS 11-15.

- Korrektur, Rückgabe, Besprechung Blatt x in Woche $i+2$
- Tutoren und Termine:
  Martin Ludwig (martin.j.ludwig@gmx.de)
  Di 14-16h, SR 011

  Conrad Schecker (conrad.schecker@gmail.com)
  Mi 14-16h, NM 113

- Bonifikation in der Klausur bzgl. erreichbarer Gesamtpunktzahl:
  $\geq 50\%$ erreicht $\rightarrow$ 1 Schritt (z.B. 2.0 $\rightarrow$ 1.7, 3.7 $\rightarrow$ 3.3)
  $\geq 75\%$ erreicht $\rightarrow$ 2 Schritte (z.B. 2.0 $\rightarrow$ 1.3, 3.3 $\rightarrow$ 2.7)

Die Themen (vorläufig)

- Randomisierte Algorithmen I.
  - Entwurfsmethoden I. (Vermeidung von Worst-Case, Fingerprinting)
  - Random Walks (Markoff-Ketten)
- Online Algorithmen
  - Der Wettbewerbsfaktor (Scheduling, Paging)

Die Themen (vorläufig)

- Randomisierte Algorithmen I.
  - Entwurfsmethoden I. (Vermeidung von Worst-Case, Fingerprinting)
  - Random Walks (Markoff-Ketten)
- Online Algorithmen
  - Der Wettbewerbsfaktor (Scheduling, Paging)

  - (Ski-Problem, k-Server-Problem, Auswahl von Experten)
  - Amortisierte Laufzeit (Binomische Heaps, Fibonacci-Heaps)
- Randomisierte Algorithmen II.
  - Entwurfsmethoden II. (Konfliktsituationen, Symmetry-Breaking, die probabilistische Methode, Stichproben)
  - Pseudo-Random Generatoren

Bücher

1. Hromkovic: Design and Analysis of Randomized Algorithms

   `https://link.springer.com/book/10.1007%2F3-540-27903-2`

2. Moore, Mertens: The Nature of Computation

   Chapter 10: Randomized Algorithms

   Informatik Bibliothek

Bücher

1. Hromkovic: Design and Analysis of Randomized Algorithms

   `https://link.springer.com/book/10.1007%2F3-540-27903-2`

2. Moore, Mertens: The Nature of Computation
   Chapter 10: Randomized Algorithms

   Informatik Bibliothek

3. Motwani, Raghavan: Randomized Agorithms

   Informatik Bibliothek

4. Borodin, El-Yaniv: Online Computation and Competitive Analysis

   Informatik Bibliothek

5. Mitzenmacher, Upfal: Probability and Computing

   Informatik Bibliothek

Stochastik Wiederholung – Donnerstag 10-12 Uhr

- Grundlagen
- (Un)abhängigkeit, Bedingte Wahrscheinlichkeit, Formel für die totale Wahrscheinlichkeit
- Zufallsvariable, wichtigste Verteilungen
- Erwartungswert, Varianz, Markoff-, Tschebischeff-, Chernoff-Ungleichungen

Randomisierter QUICKSORT

- Randomisierter QUICKSORT arbeitet mit zufälligen Pivotelementen
- Pivot wird **in jedem Schritt** gemäß **Gleichverteilung** gewählt.
- Betrachte die **erwartete Laufzeit** für **jede fixierte Eingabe** S.

Randomisierter QUICKSORT

- Randomisierter QUICKSORT arbeitet mit zufälligen Pivotelementen
- Pivot wird **in jedem Schritt** gemäß **Gleichverteilung** gewählt.
- Betrachte die **erwartete Laufzeit** für **jede fixierte Eingabe** S.

Theorem: Im randomisierten QUICKSORT mit uniform zufälliger Pivotwahl ist die **erwartete Laufzeit für jede fixierte Eingabe** S $2n \ln n + \mathcal{O}(n)$.

Randomisierter QUICKSORT

- Randomisierter QUICKSORT arbeitet mit zufälligen Pivotelementen
- Pivot wird **in jedem Schritt** gemäß **Gleichverteilung** gewählt.
- Betrachte die **erwartete Laufzeit** für **jede fixierte Eingabe** S.

Theorem: Im randomisierten QUICKSORT mit uniform zufälliger Pivotwahl ist die **erwartete Laufzeit für jede fixierte Eingabe** S $2n \ln n + \mathcal{O}(n)$.

Die Erwartung wird über die Zufallsentscheidungen des Algorithmus genommen.

Was ist ein randomisierter Algorithmus?

Was ist ein randomisierter Algorithmus?

Ein **randomisierter Algorithmus** verfügt über alle Fähigkeiten eines deterministischen Algorithmus.

Er kann zusätzlich auch Zufallsbits anfordern: Er erhält das Bit 0 oder 1 jeweils mit Wahrscheinlichkeit $1/2$.

Was ist ein randomisierter Algorithmus?

Ein **randomisierter Algorithmus** verfügt über alle Fähigkeiten eines deterministischen Algorithmus.

Er kann zusätzlich auch Zufallsbits anfordern: Er erhält das Bit 0 oder 1 jeweils mit Wahrscheinlichkeit $1/2$.

Der Algorithmus wird verschiedenste Berechnungen B in Abhängigkeit von den erhaltenen Zufallsbits ausführen.

Die erwartete Laufzeit

Definition:

Sei A ein randomisierter Algorithmus und w eine fixierte Eingabe.

Die erwartete Laufzeit

Definition:

Sei A ein randomisierter Algorithmus und w eine fixierte Eingabe.

- die Wahrscheinlichkeit einer Berechnung B von A ist $p_B = 1/2^k$, falls die Berechnung B k Zufallsbits anfordert;

Die erwartete Laufzeit

Definition:

Sei A ein randomisierter Algorithmus und w eine fixierte Eingabe.

- die Wahrscheinlichkeit einer Berechnung B von A ist $p_B = 1/2^k$, falls die Berechnung B k Zufallsbits anfordert;
- sei Zeit_B die Anzahl der Schritte einer Berechnung B;

Die erwartete Laufzeit

Definition:

Sei A ein randomisierter Algorithmus und w eine fixierte Eingabe.

- die Wahrscheinlichkeit einer Berechnung B von A ist $p_B = 1/2^k$, falls die Berechnung B k Zufallsbits anfordert;
- sei Zeit_B die Anzahl der Schritte einer Berechnung B;

Die **erwartete Laufzeit** von A auf Eingabe w ist

$$\sum_{B \text{Berechnung}} p_B \cdot \mathrm{Zeit}_B$$

Was ist ein Online-Algorithmus?

Was ist ein Online-Algorithmus?

Ein Online-Algorithmus erhält eine **Folge von Eingaben**.

Er berechnet die i-te Ausgabe (trifft eine Entscheidung über die i-ten Eingabe) **direkt nachdem** die i-te Eingabe erhalten wurde, **ohne die zukünftigen** Eingaben gesehen zu haben.