Script generated by TTT

Title: TÃ¤ubig: GAD (23.05.2013)

Date: Thu May 23 12:07:28 CEST 2013

Duration: 31:35 min

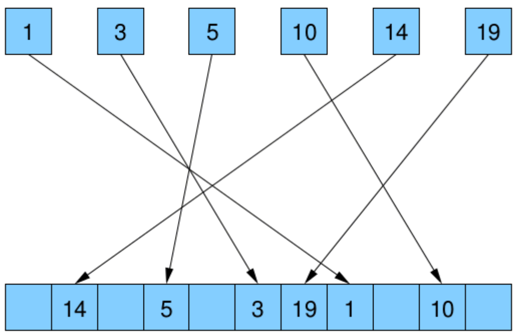
Pages: 17

Übersicht

5 Hashing
- Hashtabellen
- Hashing with Chaining
- Universelles Hashing
- Hashing with Linear Probing
- Anpassung der Tabellengröße
- Perfektes Hashing
- Diskussion / Alternativen

Perfektes Hashing für statisches Wörterbuch

- bisher: konstante *erwartete* Laufzeit
  falls m im Vergleich zu n genügend groß gewählt wird
  (nicht ausreichend für Real Time Scenario)
- Ziel: konstante Laufzeit im worst case für find()
  durch perfekte Hashtabelle ohne Kollisionen
- Annahme: statische Menge S von n Elementen



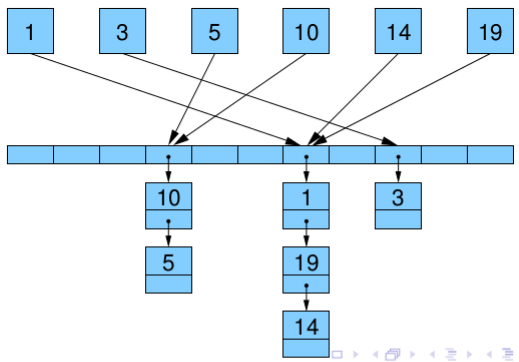
Statisches Wörterbuch

- S: feste Menge von n Elementen mit Schlüsseln k_1 bis k_n
- H_m: c-universelle Familie von Hashfunktionen auf $\{0, \ldots, m-1\}$
  (Hinweis: 2-universelle Familien existieren für alle m)
- $C(h)$ für $h \in H_m$: Anzahl Kollisionen in S für h, d.h.

$$C(h) = \left|\{(x, y) : x, y \in S,\ x \neq y,\ h(x) = h(y)\}\right|$$

Beispiel:

$C(h) = 2 + 6 = 8$



Statisches Wörterbuch

Lemma

Es gilt

$$\mathbb{E}[C(h)] \leq cn(n-1)/m$$

Statisches Wörterbuch

Lemma

Es gilt

$$\mathbb{E}[C(h)] \leq cn(n-1)/m$$

Beweis.

- Definiere $n(n-1)$ Indikator-Zufallsvariablen $X_{ij}(h)$:
  Für $i \neq j$ sei $X_{ij}(h) = 1 \;\Leftrightarrow\; h(k_i) = h(k_j)$.
- Dann ist $C(h) = \sum_{i \neq j} X_{ij}(h)$

$$\mathbb{E}[C] = \mathbb{E}\left[\sum_{i \neq j} X_{ij}\right] = \sum_{i \neq j} \mathbb{E}\left[X_{ij}\right] = \sum_{i \neq j} \Pr\left[X_{ij} = 1\right] \leq n(n-1) \cdot c/m$$

$\Rightarrow$ Für quadratische Tabellengröße ist die erwartete Anzahl von Kollisionen (und damit die erwartete worst-case-Laufzeit für find) eine Konstante. □

Statisches Wörterbuch

Lemma

Für mindestens die Hälfte der Funktionen $h \in H_m$ gilt:

$$C(h) \leq 2cn(n-1)/m$$

Statisches Wörterbuch

Lemma

Für mindestens die Hälfte der Funktionen $h \in H_m$ gilt:

$$C(h) \leq 2cn(n-1)/m$$

Beweis.

- Aus Lemma $\mathbb{E}[C(h)] \leq cn(n-1)/m$ und Markov-Ungleichung $\Pr[X \geq k \cdot \mathbb{E}[X]] \leq \frac{1}{k}$ folgt:

$$\Pr[C(h) \geq 2cn(n-1)/m] \quad \leq \quad \Pr[C(h) \geq 2\mathbb{E}[C(h)]] \quad \leq \quad \frac{1}{2}$$

$\Rightarrow$ Für höchstens die Hälfte der Funktionen ist $C(h) \geq \frac{2cn(n-1)}{m}$

$\Rightarrow$ Für mindestens die Hälfte der Funktionen ist $C(h) \leq \frac{2cn(n-1)}{m}$

□

Statisches Wörterbuch

Lemma

Wenn $m \geq cn(n-1)+1$, dann bildet mindestens die Hälfte der Funktionen $h \in H_m$ die Schlüssel injektiv in die Indexmenge der Hashtabelle ab.

Beweis.

- Für mindestens die Hälfte der Funktionen $h \in H_m$ gilt $C(h) < 2$.
- Da $C(h)$ immer eine gerade Zahl sein muss, folgt aus $C(h) < 2$ direkt $C(h) = 0$

$\Rightarrow$ keine Kollisionen (bzw. injektive Abbildung)

□

- Wähle zufällig $h \in H_m$ mit $m \geq cn(n-1)+1$
- Prüfe, ob injektiv $\Rightarrow$ behalten oder erneut wählen

$\Rightarrow$ Nach durchschnittlich 2 Versuchen erfolgreich

Statisches Wörterbuch

Ziel: ~~lineare Tabellengröße~~

Idee: zweistufige Abbildung der Schlüssel

- 1. Stufe bildet Schlüssel auf Buckets von konstanter durchschnittlicher Größe ab
- 2. Stufe benutzt quadratisch viel Platz für jedes Bucket
- Benutze Information über $C(h)$, um die Anzahl Schlüssel zu beschränken, die auf jede Tabellenposition abgebildet werden

Statisches Wörterbuch

- B_ℓ^h: Menge der Elemente in S, die h auf ℓ abbildet, $\ell \in \{0, \ldots, m-1\}$ und $h \in H_m$
- b_ℓ^h: Kardinalität von B_ℓ^h, also $b_\ell^h := \left|B_\ell^h\right|$
- Für jedes ℓ führen die Schlüssel in B_ℓ^h zu $b_\ell^h(b_\ell^h - 1)$ Kollisionen
- Also gilt:

$$C(h) = \sum_{\ell \in \{0,\ldots,m-1\}} b_\ell^h(b_\ell^h - 1)$$

Perfektes statisches Hashing: 1. Stufe

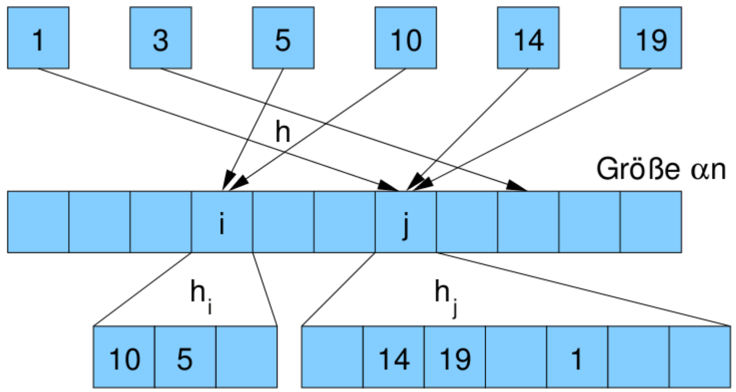
Konstruktion der Hashtabelle:

- Sei α eine (später festzulegende) Konstante
- Wähle Funktion $h \in H_{\lceil \alpha n \rceil}$, um S in Teilmengen B_ℓ aufzuspalten.
- Wähle h dabei aus dem Teil von $H_{\lceil \alpha n \rceil}$, für den gilt

$$C(h) \leq \frac{2cn(n-1)}{\lceil \alpha n \rceil} \leq \frac{2cn}{\alpha} \qquad \text{(vorletztes Lemma)}$$

- Für jedes $\ell \in \{0, \ldots, \lceil \alpha n \rceil - 1\}$ seien B_ℓ die Elemente, die auf ℓ abgebildet werden und $b_\ell = |B_\ell|$ deren Anzahl

Perfektes statisches Hashing



Perfektes statisches Hashing: 2. Stufe

- Für jedes B_ℓ:

  Sei $\boxed{m_\ell} = cb_\ell(b_\ell - 1) + 1$

  Wähle eine Funktion $h_\ell \in H_{m_\ell}$, die B_ℓ injektiv in $\{0, \ldots, m_\ell - 1\}$ abbildet

  (mindestens die Hälfte der Funktionen in H_{m_ℓ} tun das)

- Hintereinanderreihung der einzelnen Tabellen ergibt eine Gesamtgröße der Tabelle von $\sum_\ell m_\ell$

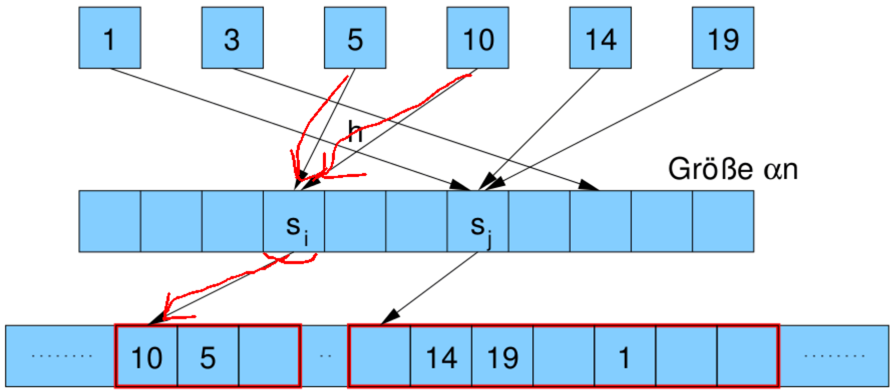
- Die Teiltabelle für B_ℓ beginnt dabei an Position $s_\ell = m_0 + m_1 + \ldots + m_{\ell-1}$

- Die Anweisungen

$$\ell = h(x); \quad \text{return } s_\ell + h_\ell(x);$$

  berechnen dann eine injektive Funktion auf S.

Perfektes statisches Hashing



Perfektes statisches Hashing

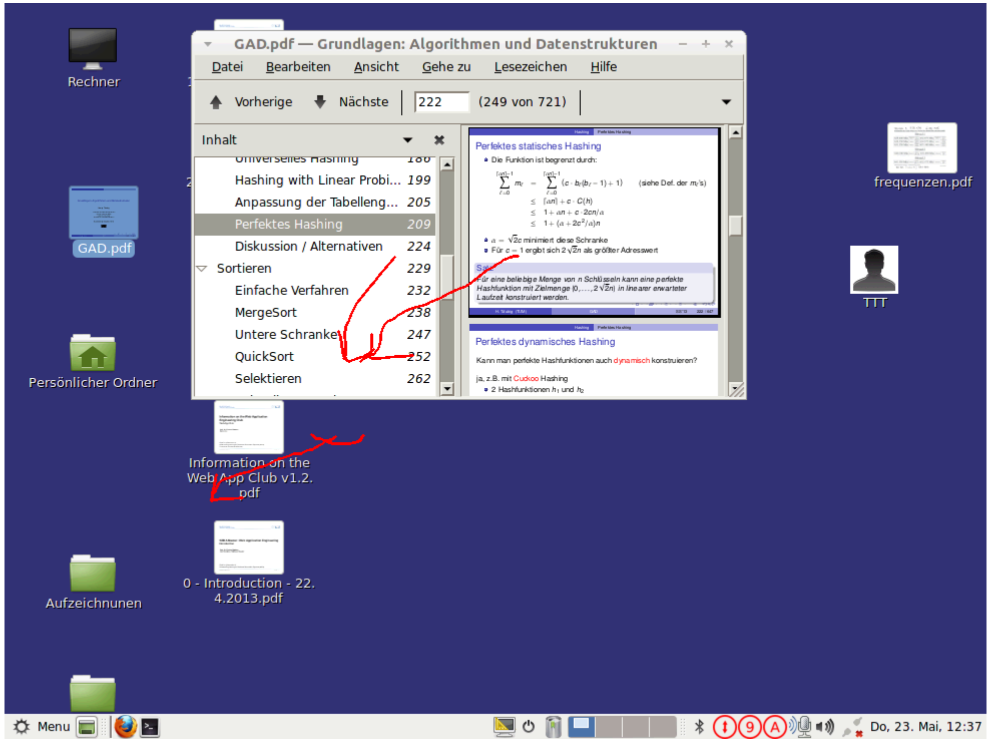
- Die Funktion ist begrenzt durch:

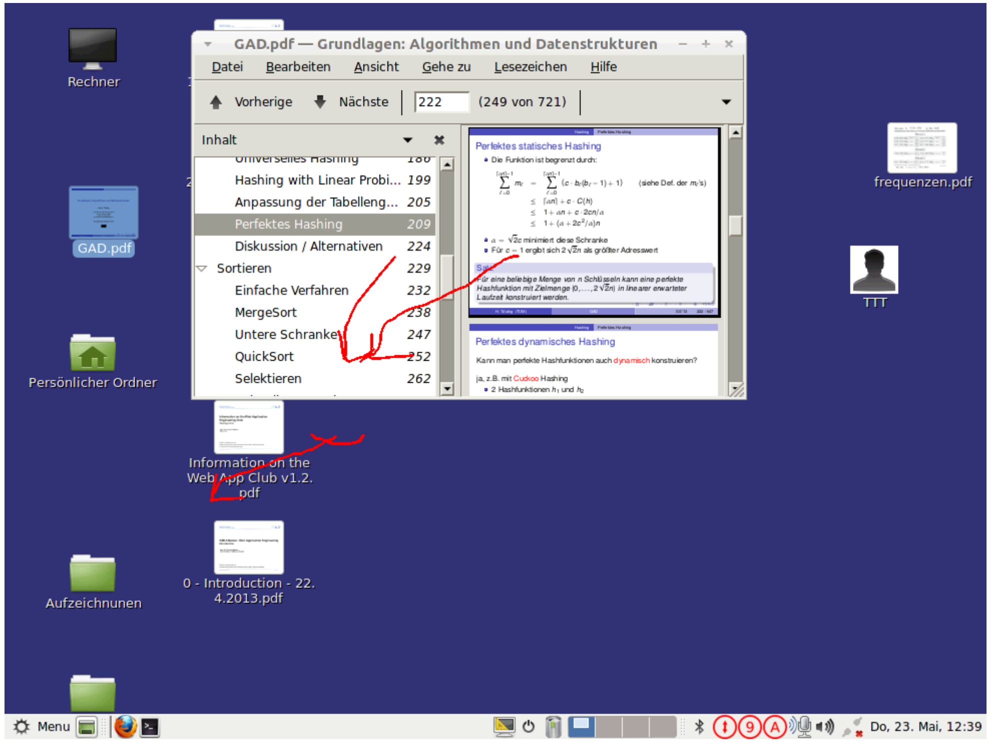
$$\begin{aligned}
\sum_{\ell=0}^{\lceil \alpha n \rceil - 1} m_\ell &= \sum_{\ell=0}^{\lceil \alpha n \rceil - 1} (c \cdot b_\ell(b_\ell - 1) + 1) \qquad \text{(siehe Def. der } m_\ell\text{'s)} \\
&\le \lceil \alpha n \rceil + c \cdot C(h) \\
&\le 1 + \alpha n + c \cdot 2cn/\alpha \\
&\le 1 + (\alpha + 2c^2/\alpha)n
\end{aligned}$$

- $\alpha = \sqrt{2}c$ minimiert diese Schranke
- Für $c = 1$ ergibt sich $2\sqrt{2}n$ als größter Adresswert

Satz

Für eine beliebige Menge von n Schlüsseln kann eine perfekte Hashfunktion mit Zielmenge $\{0, \ldots, 2\sqrt{2}n\}$ in linearer erwarteter Laufzeit konstruiert werden.


GAD.pdf — Grundlagen: Algorithmen und Datenstrukturen
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe
Vorherige Nächste 222 (249 von 721)
Inhalt
Hashing with Linear Probi... 199
Anpassung der Tabelleng... 205
Perfektes Hashing 209
Diskussion / Alternativen 224
Sortieren 229
Einfache Verfahren 232
MergeSort 238
Untere Schranke 247
QuickSort 252
Selektieren 262
Perfektes statisches Hashing
Perfektes dynamisches Hashing
Kann man perfekte Hashfunktionen auch dynamisch konstruieren?
ja, z.B. mit Cuckoo Hashing
Rechner
GAD.pdf
Persönlicher Ordner
Aufzeichnunen
Information on the Web App Club v1.2.pdf
0 - Introduction - 22. 4.2013.pdf
frequenzen.pdf
TTT
Menu
Do, 23. Mai, 12:37


GAD.pdf — Grundlagen: Algorithmen und Datenstrukturen
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe
Vorherige Nächste 222 (249 von 721)
Inhalt
Hashing with Linear Probi... 199
Anpassung der Tabelleng... 205
Perfektes Hashing 209
Diskussion / Alternativen 224
Sortieren 229
Einfache Verfahren 232
MergeSort 238
Untere Schranke 247
QuickSort 252
Selektieren 262
Perfektes statisches Hashing
Perfektes dynamisches Hashing
Kann man perfekte Hashfunktionen auch dynamisch konstruieren?
ja, z.B. mit Cuckoo Hashing
Rechner
GAD.pdf
Persönlicher Ordner
Aufzeichnunen
Information on the Web App Club v1.2.pdf
0 - Introduction - 22. 4.2013.pdf
frequenzen.pdf
TTT
Menu
Do, 23. Mai, 12:39