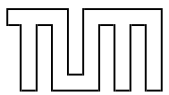
# Grundlagen: Algorithmen und Datenstrukturen

Prof. Dr. Hanjo Täubig

Lehrstuhl für Effiziente Algorithmen
(Prof. Dr. Ernst W. Mayr)
Institut für Informatik
Technische Universität München

Sommersemester 2010



# Übersicht

# Dynamisches Wörterbuch

Problem: Hashtabelle ist zu groß oder zu klein
(sollte nur um konstanten Faktor von Anzahl der Elemente abweichen)

Lösung: Reallokation

- Wähle geeignete Tabellengröße
- Wähle neue Hashfunktion
- Übertrage Elemente auf die neue Tabelle

# Dynamisches Wörterbuch

Problem: Tabellengröße $m$ sollte prim sein
(für eine gute Verteilung der Schlüssel)

Lösung:

- Für jedes $k$ gibt es eine Primzahl in $[k^3, (k+1)^3]$
- Jede Zahl $z \leq (k+1)^3$, die nicht prim ist, muss einen Teiler $t \leq \sqrt{(k+1)^3} = (k+1)^{3/2}$ haben.
- Bestimme $k$, so dass $k^3 \leq m \leq (k+1)^3$
- Streiche für jede Zahl $j = 2, \ldots, (k+1)^{3/2}$ die Vielfachen in $[k^3, (k+1)^3]$
- Für jedes $j$ kostet das Zeit $((k+1)^3 - k^3)/j = O(k^2/j)$

# Dynamisches Wörterbuch

- Hilfsmittel: Wachstum der harmonischen Reihe

$$\ln n \leq H_n = \sum_{i=1}^{n} \frac{1}{i} \leq \ln n + 1$$

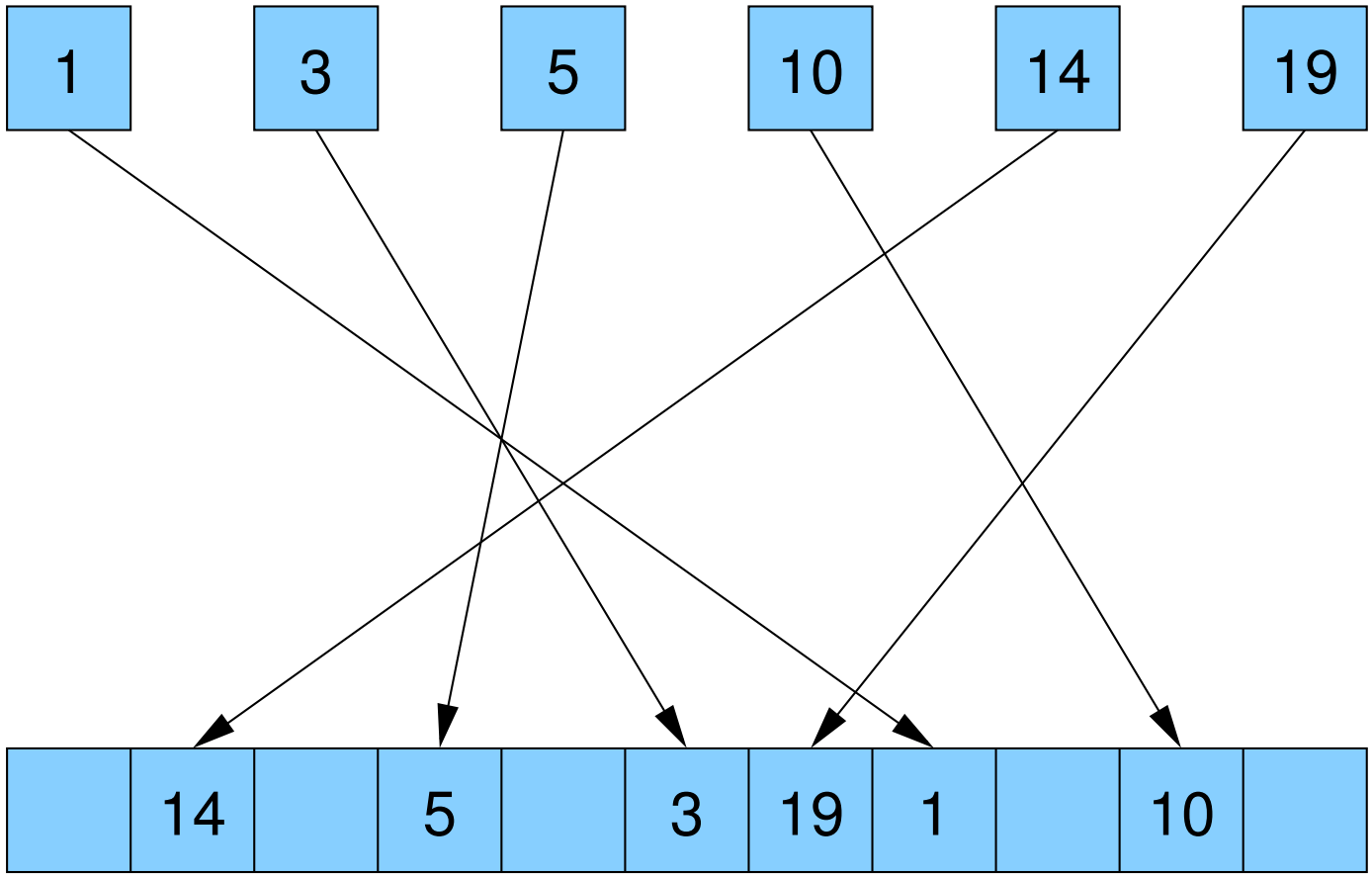
- insgesamt:

$$\begin{aligned}
\sum_{j \leq (k+1)^{3/2}} O\left(\frac{k^2}{j}\right) &= k^2 \sum_{j \leq (k+1)^{3/2}} O\left(\frac{1}{j}\right) \\
&\in O\left(k^2 \ln\left((k+1)^{3/2}\right)\right) \\
&\in O(k^2 \ln k) \\
&\in o(m)
\end{aligned}$$

$\Rightarrow$ Kosten zu vernachlässigen im Vergleich zur Initialisierung der Tabelle der Größe $m$

# Statisches Wörterbuch

Ziel: perfekte Hashtabelle



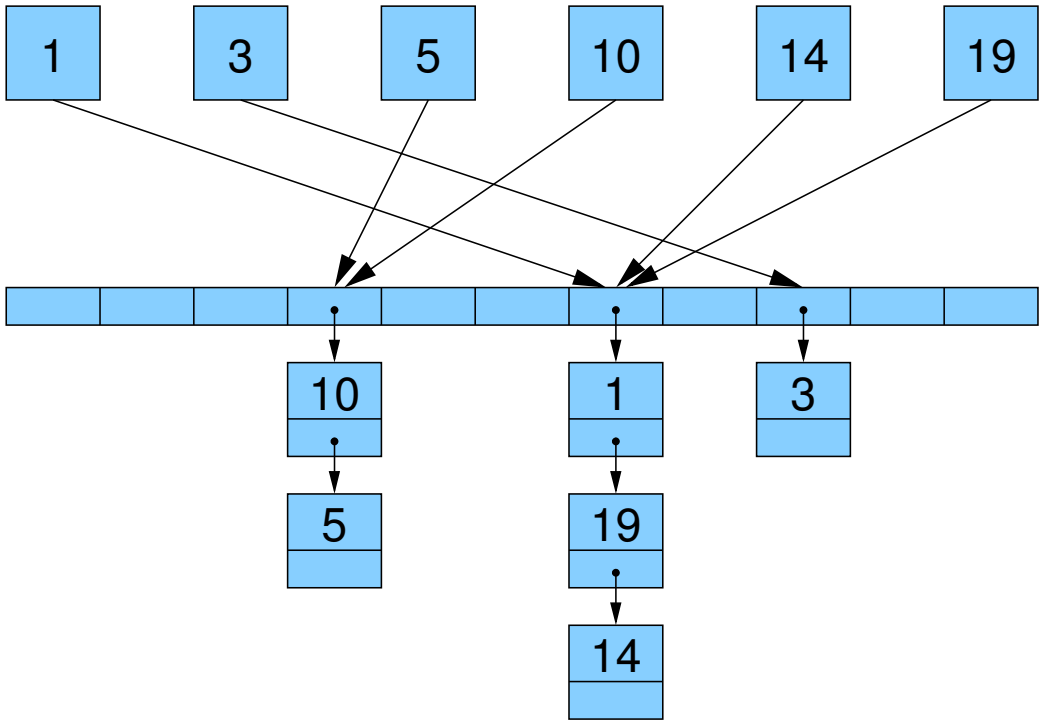
# Statisches Wörterbuch

- $S$: feste Menge von $n$ Elementen mit Schlüsseln $k_1$ bis $k_n$
- $H_m$: Familie $c$-universeller Hashfunktionen auf $\{0, \ldots, m-1\}$
- $C(h)$ für $h \in H_m$:
  Anzahl Kollisionen zwischen Elementen 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*

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

## 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. □

# Markov-Ungleichung

## Satz

*Für jede nichtnegative Zufallsvariable $X$ und Konstante $c$ gilt:*

$$\Pr[X \geq c \cdot \mathbb{E}[X]] \leq \frac{1}{c}$$

## Beweis.

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{z \in \mathbb{R}} z \cdot \Pr[X = z] \\
&\geq \sum_{z \geq c \cdot \mathbb{E}[X]} z \cdot \Pr[X = z] \geq \sum_{z \geq c \cdot \mathbb{E}[X]} c \cdot \mathbb{E}[X] \cdot \Pr[X = z] \\
&\geq c \cdot \mathbb{E}[X] \cdot \Pr[X \geq c \cdot \mathbb{E}[X]]
\end{aligned}$$

□

# Statisches Wörterbuch

## Lemma

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

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

## Beweis.

- Aus Markov-Ungleichung $\Pr[X \geq c \cdot \mathbb{E}[X]] \leq \frac{1}{c}$ folgt:

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

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

$\Rightarrow$ Für mindestens die Hälfte der Funktionen ist

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

□

# 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

- Für $\ell \in \{0, \ldots, m-1\}$ und $h \in H_m$ seien $B_\ell^h$ die Elemente in $S$, die durch $h$ auf $\ell$ abgebildet werden.
- $b_\ell^h$ sei die Kardinalität von $B_\ell^h$
- Für jedes $\ell$ führen die Schlüssel in $B_\ell^h$ zu $b_\ell^h(b_\ell^h - 1)$ Schlüsselpaaren, die auf die gleiche Position abgebildet werden.
- 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 $\alpha$ eine (später festzulegende) Konstante
- Wähle Funktion $h \in H_{\lceil \alpha n \rceil}$, um $S$ in Teilmengen $B_\ell$ 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 $\ell$ seien $B_\ell$ die Elemente, die auf $\ell$ abgebildet werden und $b_\ell = |B_\ell|$ deren Anzahl

# Perfektes statisches Hashing: 2. Stufe

- Für jedes $B_\ell$:

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

  Wähle eine Funktion $h_\ell \in H_{m_\ell}$, die $B_\ell$ injektiv in $\{0, \ldots, m_\ell - 1\}$ abbildet
  (mindestens die Hälfte der Funktionen in $H_{m_\ell}$ tun das)

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

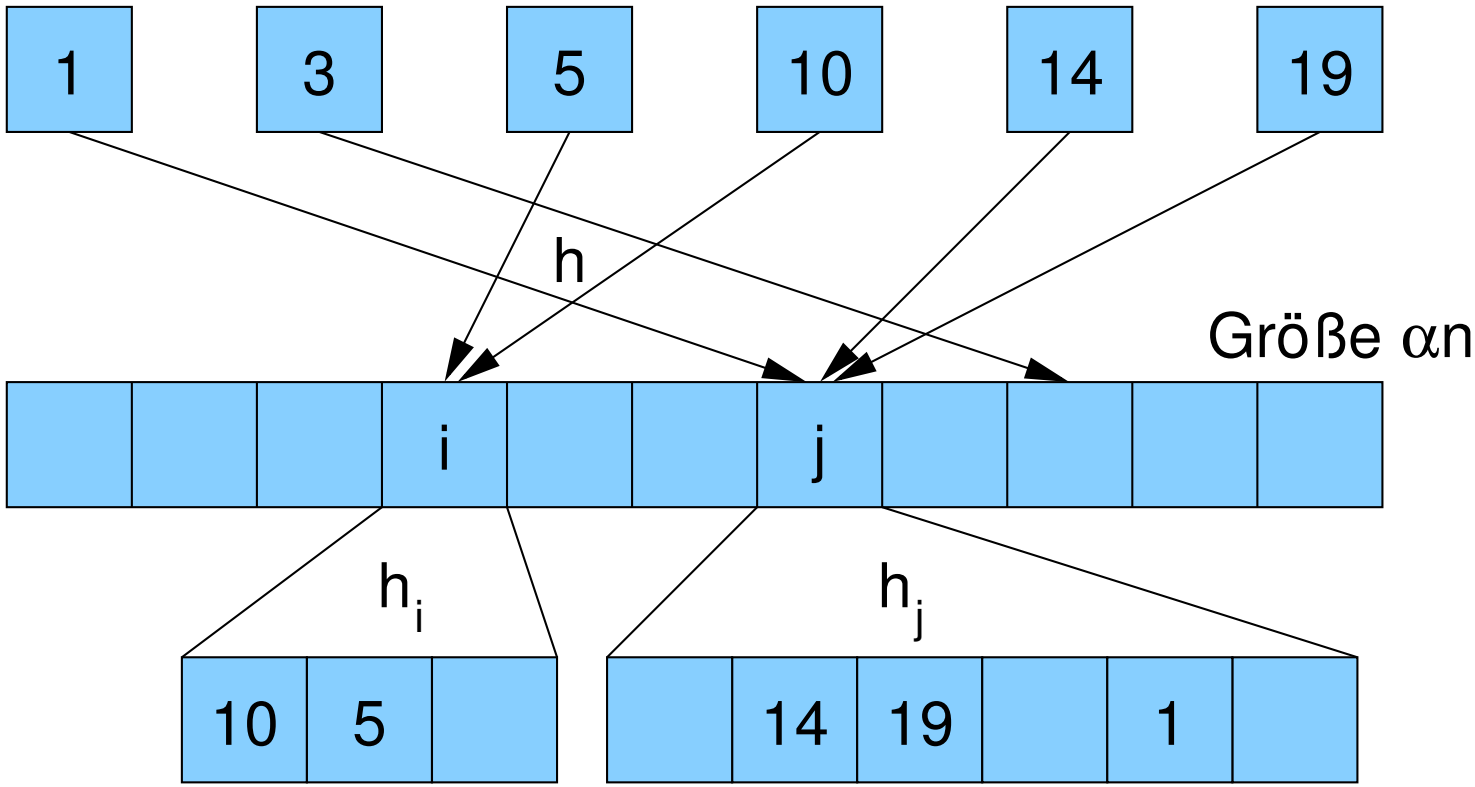
- Die Teiltabelle für $B_\ell$ beginnt dabei an Position
  $s_\ell = m_0 + m_1 + \ldots + m_{\ell-1}$

- Die Anweisungen

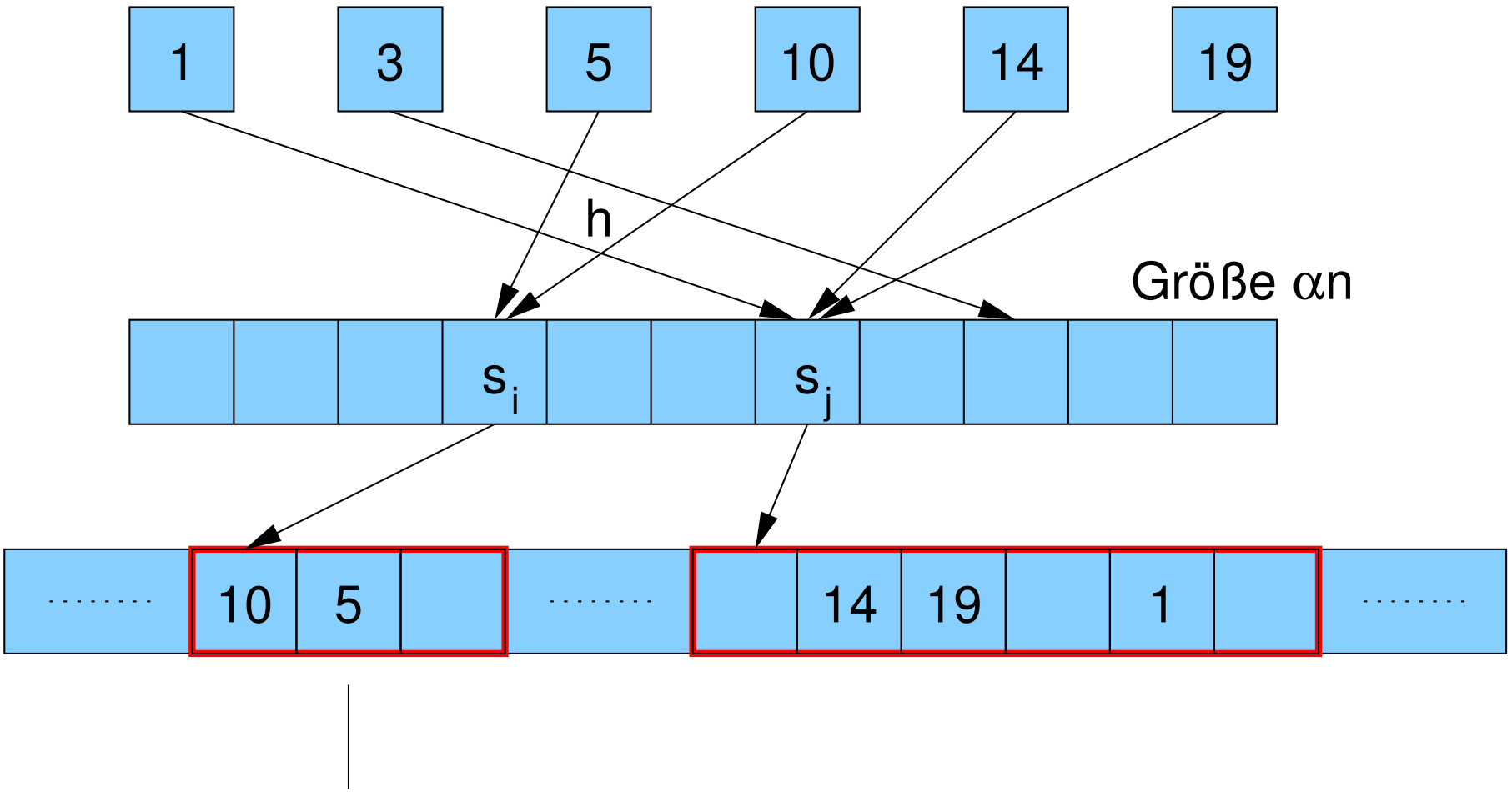
$$\ell = h(x); \quad \text{return } s_\ell + h_\ell(x);$$

  berechnen dann eine injektive Funktion auf $S$.

# Perfektes statisches Hashing



# Perfektes statisches Hashing



# Perfektes statisches Hashing

- Die Funktion ist begrenzt durch:

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

- $\alpha = \sqrt{2}c$ 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.*