Technische Universität Braunschweig

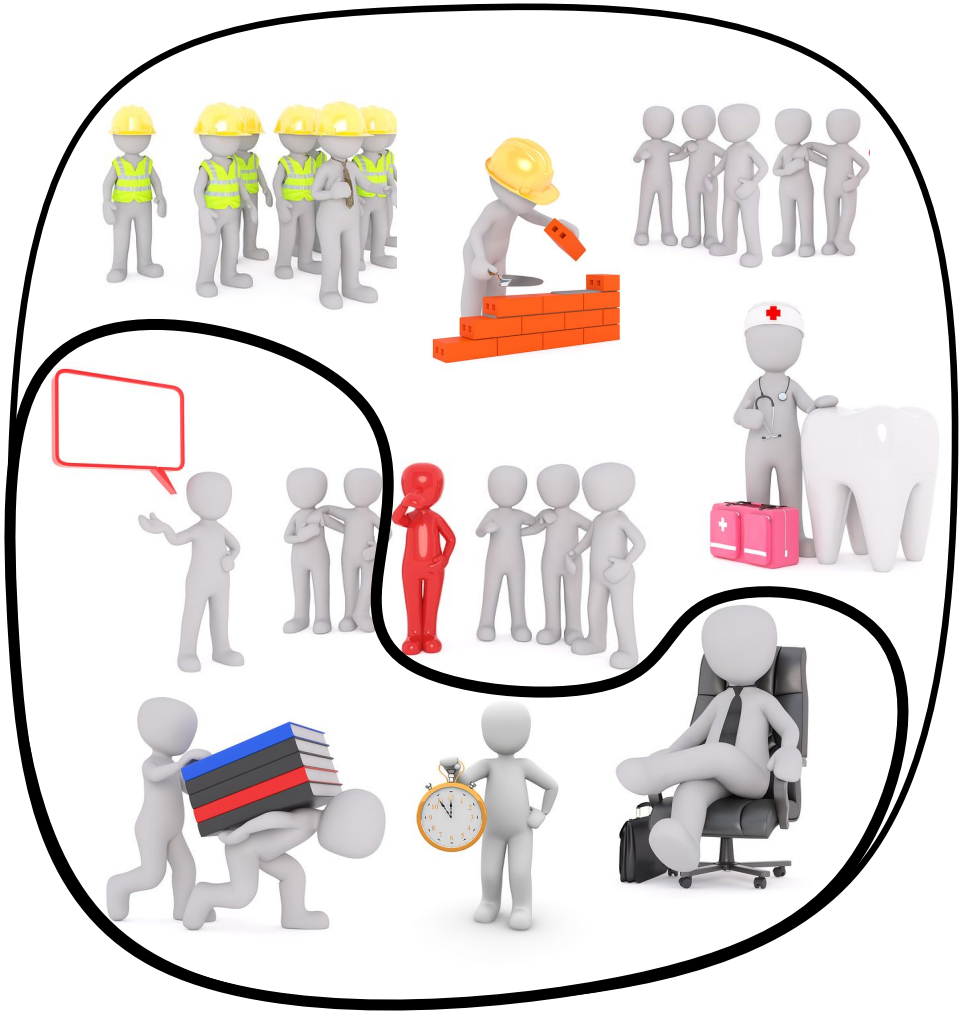
Algorithmen und Datenstrukturen II

10. Vorlesung

Linda Kleist, 03.07.2019

Motivation

Motivation



Motivation

Objekte bestehen aus

- Schlüssel: Identifikation
- Datensatz: weitere Infos

Beispiele:

- Personaldatenbank
  → Schlüssel: Personalnummer
- Online-Katalog
  → Schlüssel: Artikelnummer
- Cache im Browser
  → Schlüssel: URL

dynamische Datenverwaltung – Hashing

Ziel

Dynamische Datenverwaltung, wobei jeder Datensatz durch einen Schlüssel eindeutig charakterisiert.

Viele Anwendungen benötigen nur einfache Operationen:

- Suchen nach Datensatz mit Schlüssel $x \rightarrow \text{search}(x)$
- Einfügen eines neuen Datensatzes mit Schlüssel $x \rightarrow \text{insert}(x)$
- Löschen eines Datensatzes mit Schlüssel $x \rightarrow \text{delete}(x)$

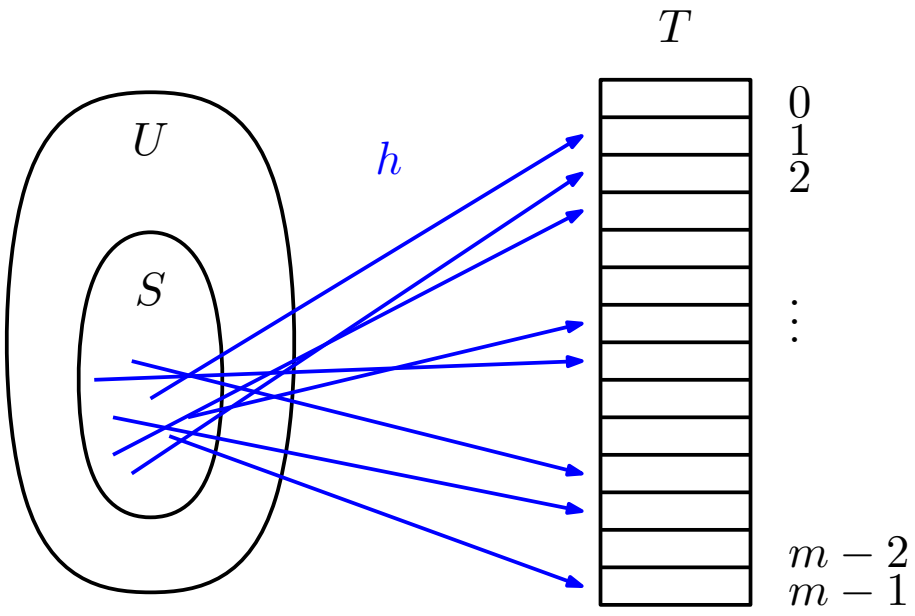
Idee

Berechne Speicherort des Datensatzes mit Schlüssel x

Hashverfahren

Universum $U = \{0, 1, \ldots, N-1\}$
Schlüsselmenge $S \subseteq U$, $n := |S|$

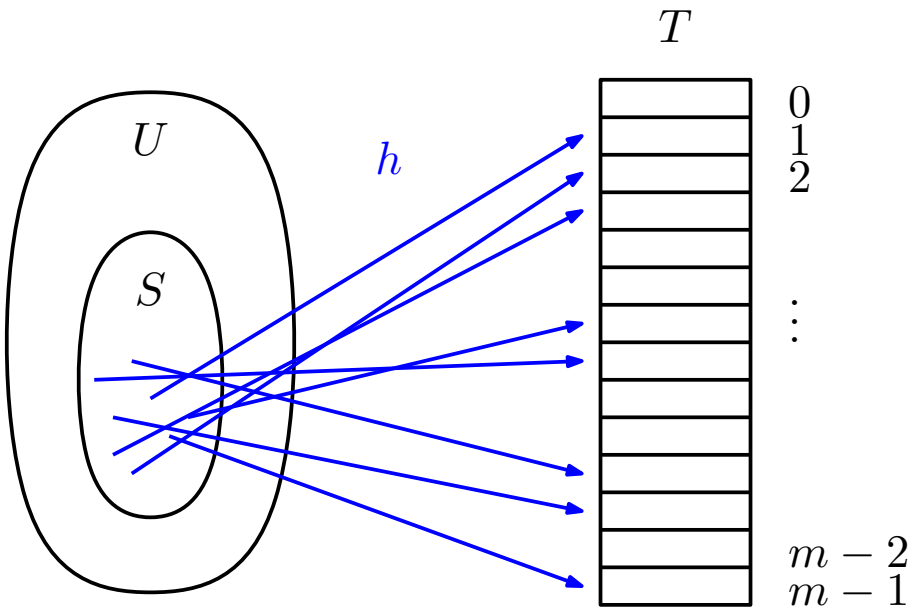
- Eine Hashtabelle der Größe m ist ein Array T mit den Zellen $T[0]$ bis $T[m-1]$ zur Speicherung der Datensätze.
- Eine Hashfunktion h liefert für jeden Schlüssel $x \in U$ eine Adresse in der Hashtabelle, d.h. $h : U \to \{0, \ldots, m-1\}$.



Hashverfahren

Universum $U = \{0, 1, \ldots, N-1\}$
Schlüsselmenge $S \subseteq U$, $n := |S|$

- Der Belegungsfaktor einer Hashtabelle der Größe m ist $\beta := \frac{n}{m}$.
- Bei einer Kollision erhalten verschiede Schlüssel x_1 und x_2 den selben Hashwert $h(x_1) = h(x_2)$ (unvermeidbar wenn $|U| > m$).

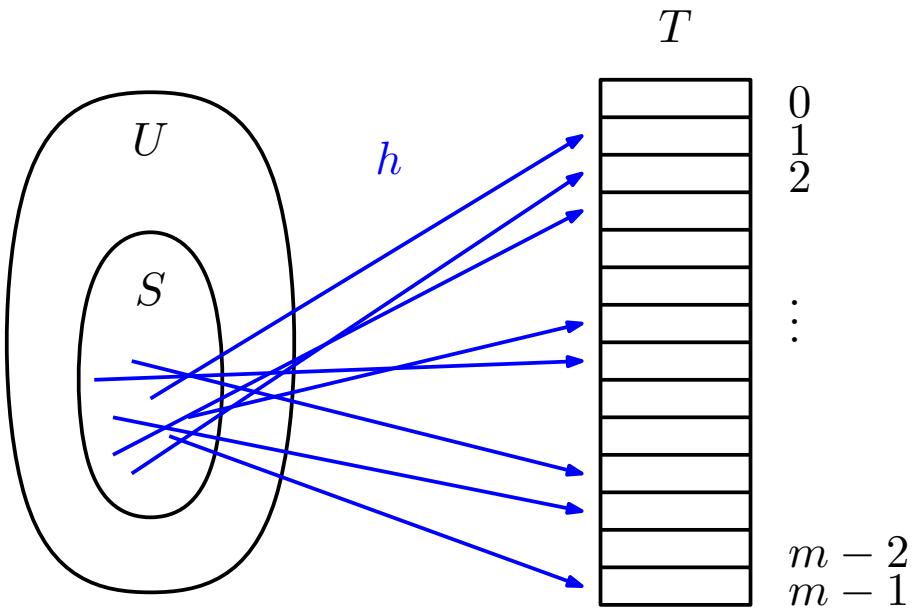


Hashverfahren

Universum $U = \{0, 1, \ldots, N-1\}$
Schlüsselmenge $S \subseteq U$, $n := |S|$

- Ein Hashverfahren ist durch
  - eine Hashtabelle,
  - eine Hashfunktion, und
  - eine Strategie zur Auflösung von Kollisionen

  gegeben.
- Herausforderungen
  - $|U| \gg |S|$
  - S ist a priori unbekannt



Hashfunktionen

Sei $U = \{0, 1, \ldots, N-1\}$ und $h\colon U \to \{0, 1, \ldots, m-1\}$.

Eine gute Hashfunktion h sollte
- den ganzen Wertebereich umfassen (surjektiv),
- die Schlüssel möglichst gleichmäßig verteilen,
- effizient berechenbar sein.

Beim idealen Hashing wählt die Hashfunktion jede Position in $\{0, 1, \ldots, m-1\}$ mit Wahrscheinlichkeit $\frac{1}{m}$.

Divisions-Rest-Methode

$$h(x) = x \bmod m := x - \left\lfloor \frac{x}{m} \right\rfloor \cdot m$$

Hashfunktionen

Sei $U = \{0, 1, \ldots, N-1\}$ und $h\colon U \to \{0, 1, \ldots, m-1\}$.

Divisions-Rest-Methode

$$h(x) = x \bmod m := x - \left\lfloor \frac{x}{m} \right\rfloor \cdot m$$

Beispiel

Sei $m = 11$ und $S = \{49, 22, 6, 52, 76, 34, 13, 29\}$.
$h(49) = 49 \bmod 11 = 5$,
$h(22) = 22 \bmod 11 = 0$, und
$h(6) = 6, h(52) = 8, h(76) = 10, h(34) = 1, h(13) = 2, h(29) = 7$.

Hashfunktionen

Sei $U = \{0, 1, \ldots, N-1\}$ und $h\colon U \to \{0, 1, \ldots, m-1\}$.

Divisions-Rest-Methode

$$h(x) = x \bmod m := x - \left\lfloor \frac{x}{m} \right\rfloor \cdot m$$

Es gilt:

- $|h^{-1}(i)| \in \left\{ \left\lfloor \frac{N}{m} \right\rfloor, \left\lceil \frac{N}{m} \right\rceil \right\}$.
- Problem: Die Daten sind oft nicht gleichverteilt!
- geeignete Wahl von m ist wichtig
  - m Zweierpotenz: $x \bmod m$ wählt die letzten $\log m$ Bits.
  - $m > 2$ Primzahl: $x \bmod m$ beeinflusst alle Bits.
  - empirisch gut: Primzahl, die weit von Zweierpotenz entfernt ist.

Kollisionen

- Kollisionen sind unvermeidbar wenn $N \gg m$!
- Wie wahrscheinlich sind Kollisionen bei $n \ll m$?

Geburtstagsparadoxon

Bei wie vielen (zufällig gewählten) Person ist es wahrscheinlich, dass mindestens zwei am selben Datum (Tag und Monat) Geburtstag haben?

$$\text{Prob}[n \text{ Daten kollisionsfrei}] = \frac{m}{m} \cdot \frac{m-1}{m} \cdot \frac{m-2}{m} \cdot \ldots \cdot \frac{m-(n-1)}{m}$$

Kollisionen

- Kollisionen sind unvermeidbar wenn $N \gg m$!
- Wie wahrscheinlich sind Kollisionen bei $n \ll m$?

Geburtstagsparadoxon

Bei wie vielen (zufällig gewählten) Person ist es wahrscheinlich, dass mindestens zwei am selben Datum (Tag und Monat) Geburtstag haben?

$$\text{Prob}[n \text{ Daten kollisionsfrei}] = \frac{m}{m} \cdot \frac{m-1}{m} \cdot \frac{m-2}{m} \cdot \ldots \cdot \frac{m-(n-1)}{m}$$

Beispiel

Für $m = 365$ ist
Prob[23 Daten kollisionsfrei] ≈ 0.49 und
Prob[50 Daten kollisionsfrei] ≈ 0.03.

Kollisionen

- Kollisionen sind unvermeidbar wenn $N \gg m$!
- Wie wahrscheinlich sind Kollisionen bei $n \ll m$?

Geburtstagsparadoxon

Bei wie vielen (zufällig gewählten) Person ist es wahrscheinlich, dass mindestens zwei am selben Datum (Tag und Monat) Geburtstag haben?

$$\text{Prob}[n \text{ Daten kollisionsfrei}] = \frac{m}{m} \cdot \frac{m-1}{m} \cdot \frac{m-2}{m} \cdot \ldots \cdot \frac{m-(n-1)}{m}$$

Satz 16

$$\text{Prob}[2m^{1/2} \text{ Daten kollisionsfrei}] \leq \left(1 - \frac{1}{m^{1/2}}\right)^{m^{1/2}} \approx \frac{1}{e} \approx 0.3679$$

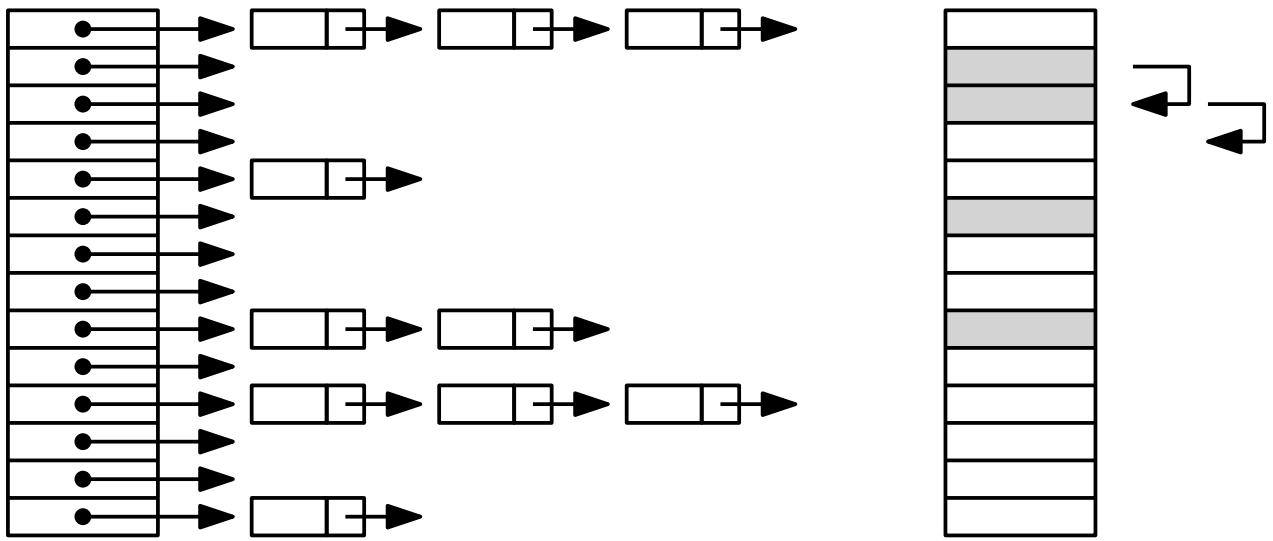
Beweis.

Tafel... □

Kollisionsbehandlung

Möglichkeiten zur Kollisionsbehandlung:

- Verkettete Listen: Jede Zelle der Hashtabelle enthält einen Zeiger auf eine Überlaufliste.
- Offene Adressierung: Bei Adresskollision nach fester Regel alternativen freien Platz suchen (Sondierungsfolge).

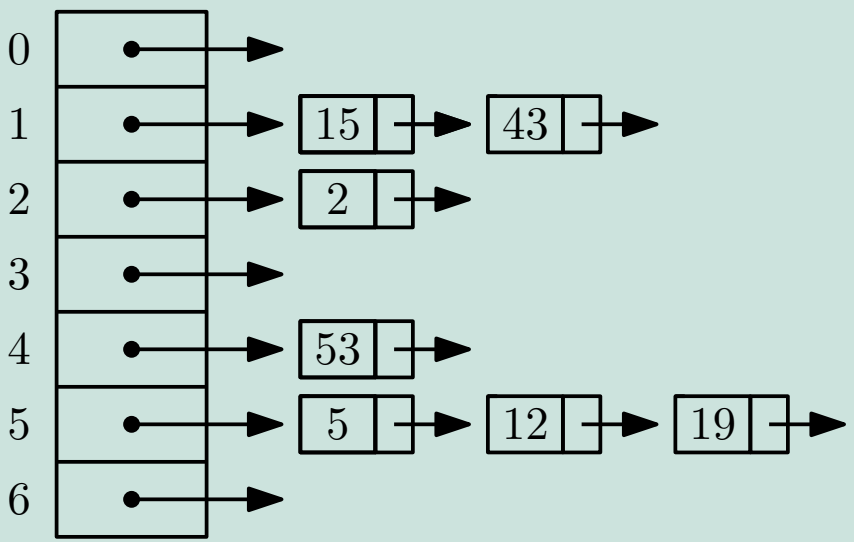


Verkette Listen

Jede Speicherzelle $T[i]$ enthält einen Zeiger auf eine Liste $L(i)$, die alle Schlüssel $x \in S$ mit $h(x) = i$ enthält.

Beispiel

Sei $m = 7$ und $h(x) = x \bmod m$. $S = \{2, 5, 12, 15, 19, 43, 53\}$ ergibt die Hashtabelle:



- Operationen:
  - search(x): suche in Liste $L(h(x))$
  - insert(x): nach erfolgloser Suche, füge x in Liste $L(h(x))$ ein.
  - delete(x): nach erfolgreicher Suche von x in Liste $L(h(x))$, entferne x aus $L(h(x))$.
- Vorteile:
  - unterstützt alle Operationen
  - $n > m$ möglich
- Nachteil:
  - Speicherplatz für Zeiger

Verkette Listen – Analyse

Satz 17

Bei zufälligen Daten und ideal streuenden Hashfunktionen enthält eine Liste $L(j)$ im Erwartungswert $\frac{n}{m} =: \beta$ Elemente.

Beweis.

Tafel... □

Bemerkungen

- Erfolglose Suche in $L(j)$ betrachtet im EW $1 + \beta$ Objekte.
- Erfolgreiche Suche in $L(j)$ betrachtet im EW $\approx 1 + \frac{\beta}{2}$ Objekte.

Korollar

- Wählt man $m \in \Theta(n)$, dann ist $\beta = \frac{n}{m} \in O(1)$ und alle Operationen haben im Mittel eine Laufzeit von $O(1)$.
- Die worst-case Laufzeit für search-Operation ist allerdings $\Omega(n)$.