

# Hashing

## Weiterführende Literatur:

- Schneider, *Taschenbuch der Informatik*, Kapitel 6.3.3 Streutabellen (Hashing), Seite 188-189
- *Algorithmen und Datenstrukturen: Tafelübung 10*, WS 2018/19, Seite 22-39
- Saake und Sattler, *Algorithmen und Datenstrukturen*, Seite 419-439
- Wikipedia-Artikel „Hashtabelle“

## Grundlagen

Das Ziel von Hashing ist einerseits einen *extrem großen Schlüsselraum* auf einen vernünftig *kleinen Bereich von ganzen Zahlen* abzubilden und andererseits dass *zwei Schlüssel auf die selbe Zahl* abgebildet werden, soll möglichst *unwahrscheinlich* sein.<sup>1</sup>

extrem großen Schlüsselraum  
kleinen Bereich von ganzen Zahlen  
zwei Schlüssel auf die selbe Zahl  
unwahrscheinlich

Daten werden in einer *Streu(wert)tabelle* (hash table) abgelegt. Aufgrund des wahlfreien Zugriffs eignen sich Felder zum Abspeichern der Daten. Eine *Hashfunktion*  $h$  bildet ein Datenelement auf einen *Hashwert* ab. Das Datenelement benötigt dazu einen *Schlüssel* (key), der das Element eindeutig identifiziert. Der Hashwert wird als Index in dem Feld verwendet. Das Datenelement wird im entsprechenden *Bucket* der Tabelle gespeichert. Bei der *Suche* nach einem Element mit *bekanntem Schlüssel* wird der Index mittels der Hashfunktion bestimmt. Dies geschieht mit *konstantem Aufwand*. Der Aufwand des Nachschlagens an entsprechender Stelle ist abhängig von der *Organisationsform*.<sup>2</sup>

Streu(wert)tabelle  
Hashfunktion  
Hashwert  
Schlüssel  
Bucket  
Suche  
bekanntem Schlüssel  
konstantem Aufwand

## Kollisionen

Da Hashfunktionen im Allgemeinen *nicht eindeutig (injektiv)* sind, können zwei unterschiedliche Schlüssel zum selben Hash-Wert, also zum selben Feld in der Tabelle, führen. Dieses Ereignis wird als *Kollision* bezeichnet. In diesem Fall muss die Hashtabelle mehrere Werte in demselben Bucket aufnehmen.

nicht eindeutig (injektiv)  
Kollision

Zur Behandlung von Kollisionen werden kollidierte Daten nach einer *Ausweichstrategie in alternativen Feldern* oder in einer *Liste* gespeichert. Schlimmstenfalls können Kollisionen zu einer *Entartung der Hashtabelle* führen, wenn wenige Hashwerte sehr vielen Objekten zugewiesen wurden, während andere Hashwerte unbenutzt bleiben.<sup>3</sup>

Ausweichstrategie in alternativen Feldern  
Liste  
Entartung der Hashtabelle

Um das Kollisions-Problem zu handhaben, gibt es diverse Kollisionsauflösungsstrategien.

## geschlossenes Hashing mit offener Adressierung

Wenn dabei ein Eintrag an einer schon belegten Stelle in der Tabelle abgelegt werden soll, wird stattdessen eine *andere freie Stelle genommen*. Häufig werden

andere freie Stelle genommen

<sup>1</sup>Seite 4 <https://moves.rwth-aachen.de/wp-content/uploads/SS15/dsal/lec13.pdf>

<sup>2</sup>*Algorithmen und Datenstrukturen: Tafelübung 10*, WS 2018/19, Seite 25.

<sup>3</sup>Wikipedia-Artikel „Hashtabelle“.

drei Varianten unterschieden:

### lineares Sondieren

konstantes Intervall

es wird um ein *konstantes Intervall* verschoben nach einer freien Stelle gesucht. Meistens wird die Intervallgröße auf 1 festgelegt.

### quadratisches Sondieren

Intervall quadriert

Nach jedem erfolglosen Suchschritt wird das *Intervall quadriert*.

### Beispiele

#### - nach Foliensatz der TU Braunschweig <sup>4</sup>

##### Formel

$$h(k, i) := h'(k) + (-1)^{i+1} \cdot \left\lfloor \frac{i+1}{2} \right\rfloor^2 \bmod m$$

$$k, k + 1^2, k - 1^2, k + 2^2, k - 2^2, \dots, k + \left(\frac{m-1}{2}\right)^2, k - \left(\frac{m-1}{2}\right)^2 \bmod m$$

##### Werte

$m = 19$ , d. h. das Feld (die Tabelle) hat die Index-Nummern 0 bis 18.

$$k = h(x) = 7$$

##### Sondierungsfolgen

i	Rechnung	Ergebnis	Index in der Tabelle
0	$7 + 0^2$	7	7
1	$7 + 1^2$	8	8
1	$7 - 1^2$	6	6
2	$7 + 2^2$	11	11
2	$7 - 2^2$	3	2
3	$7 + 3^2 = 7 + 9$	16	16
3	$7 - 3^2 = 7 - 9$	-2	17 <small>(<math>19 - 2 = 10</math>) oder (<math>0 \rightarrow 0, -1 \rightarrow 18, -2 \rightarrow 17</math>)</small>
4	$7 + 4^2 = 7 + 16$	23	4 <small>(<math>23 - 19 = 4</math>) oder (<math>19 \rightarrow 0, 20 \rightarrow 1, 21 \rightarrow 2, 22 \rightarrow 3, 23 \rightarrow 4</math>)</small>
4	$7 - 4^2 = 7 - 16$	-9	10 <small>(<math>19 - 9 = 10</math>) oder (<math>0 \rightarrow 0, -1 \rightarrow 18, -2 \rightarrow 17, \dots, -9 \rightarrow 10</math>)</small>
5	$7 + 5^2 = 7 + 25$	32	13 <small>(<math>32 - 19 = 13</math>)</small>
5	$7 - 5^2 = 7 - 25$	-18	1 <small>(<math>19 - 18 = 1</math>)</small>

#### - nach Foliensatz der RWTH Aachen <sup>5</sup>

$$h'(k) = k \bmod 11$$

$$h(k, i) = (h'(k) + i + 3i^2) \bmod 11$$

$$h'(17) = 17 \bmod 11 = 6$$

<sup>4</sup>Seite 25 <https://www.ibr.cs.tu-bs.de/courses/ws0708/aud/skript/hash.np.pdf>

<sup>5</sup>Seite 19 <https://moves.rwth-aachen.de/wp-content/uploads/SS15/dsal/lec13.pdf>

**Sondierungsfolgen**

$$h(17,0) = (17 + 0 + 3 \cdot 0^2) \bmod 11 = 6$$

$$h(17,1) = (17 + 1 + 3 \cdot 1^2) \bmod 11 = 21 \bmod 11 = 10$$

$$h(17,2) = (17 + 2 + 3 \cdot 2^2) \bmod 11 = 31 \bmod 11 = 31 - 2 \cdot 11 = 9$$

$$h(17,3) = (17 + 3 + 3 \cdot 3^2) \bmod 11 = 47 \bmod 11 = 47 - 4 \cdot 11 = 3$$

**doppeltes Hashen**

eine *weitere Hash-Funktion* liefert das Intervall.

weitere Hash-Funktion

**offenes Hashing mit geschlossener Adressierung**

Anstelle der gesuchten Daten enthält die Hashtabelle hier *Behälter* (englisch *Buckets*), die alle Daten mit gleichem Hash-Wert aufnehmen. Es müssen die *Elemente im Behälter durchsucht werden*. Oft wird die Verkettung durch eine lineare Liste pro Behälter realisiert.<sup>6</sup>

Behälter

Buckets

Elemente im Behälter durchsucht werden

**Die Divisionsrestmethode<sup>7</sup>**

Die Divisionsrestmethode - auch Modulo genannt liefert eine Hashfunktion. Die Funktion lautet:  $h(k) = k \bmod m$ .  $m$  ist die Größe der Hashtabelle. Die Hash-Funktion kann sehr schnell berechnet werden. Die Wahl der Tabellengröße  $m$  beeinflusst die Kollisionswahrscheinlichkeit der Funktionswerte von  $h$ . Für praxisrelevante Anwendungen liefert die Wahl einer Primzahl für  $m$ .

**Belegungsfaktor<sup>8</sup>**

$$\text{Belegungsfaktor} = \frac{\text{Anzahl tatsächlich eingetragener Schlüssel}}{\text{Anzahl Hashwerte}}$$

**Literatur**

- [1] *Algorithmen und Datenstrukturen: Tafelübung 10, WS 2018/19*. [https://www.studon.fau.de/file2619757\\_download.html](https://www.studon.fau.de/file2619757_download.html). FAU: Lehrstuhl für Informatik 2 (Programmiersysteme).
- [2] Gunter Saake und Kai-Uwe Sattler. *Algorithmen und Datenstrukturen. Eine Einführung in Java*. 2014.
- [3] Uwe Schneider. *Taschenbuch der Informatik*. 7. Aufl. Hanser, 2012. ISBN: 9783446426382.
- [4] Wikipedia-Artikel „Divisionsrestmethode“. <https://de.wikipedia.org/wiki/Divisionsrestmethode>.
- [5] Wikipedia-Artikel „Hashtabelle“. <https://de.wikipedia.org/wiki/Hashtabelle>.

<sup>6</sup>Wikipedia-Artikel „Hashtabelle“.

<sup>7</sup>Wikipedia-Artikel „Divisionsrestmethode“.

<sup>8</sup>Algorithmen und Datenstrukturen: Tafelübung 10, WS 2018/19, Seite 29.