# Algorithmen und Datenstrukturen

# Teil 3 - Hashverfahren

Prof. Dr. Peter Jüttner

# Hashverfahren

## Suchen in Feldern ➔ Hashverfahren

- **Ausgangsituation: Elemente eines Datentyps sind nach einem Schlüssel in einem Array gespeichert.*)**
- **Aufgabe: Berechnen des Arrayindex i zu einem gegebenen Schlüssel s als Funktion von s**)**

➔ **Lösung durch Anwendung einer Hashfunktion h(s)**

*) verallgemeinert kann statt eines Arrayindex einen Speicheradresse gesucht werden.

**) weitere Funktionen zum Aufbau („Füllen“) und Löschen des Arrays sind notwendig

# Hashverfahren

## Suchen in Feldern ➔ Hashverfahren h

Schlüssel s

$h(s) = 6$

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | $D_8$ | $D_9$ | $D_{10}$ |

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren**

**Beispiele:**

- **Verwaltung von Personendaten über das Geburtsdatum**
- **Verwaltung von Kfz-Daten über das Kennzeichen**
- **Symboltabellen in Compilern**

# Hashverfahren

Hochschule Deggendorf

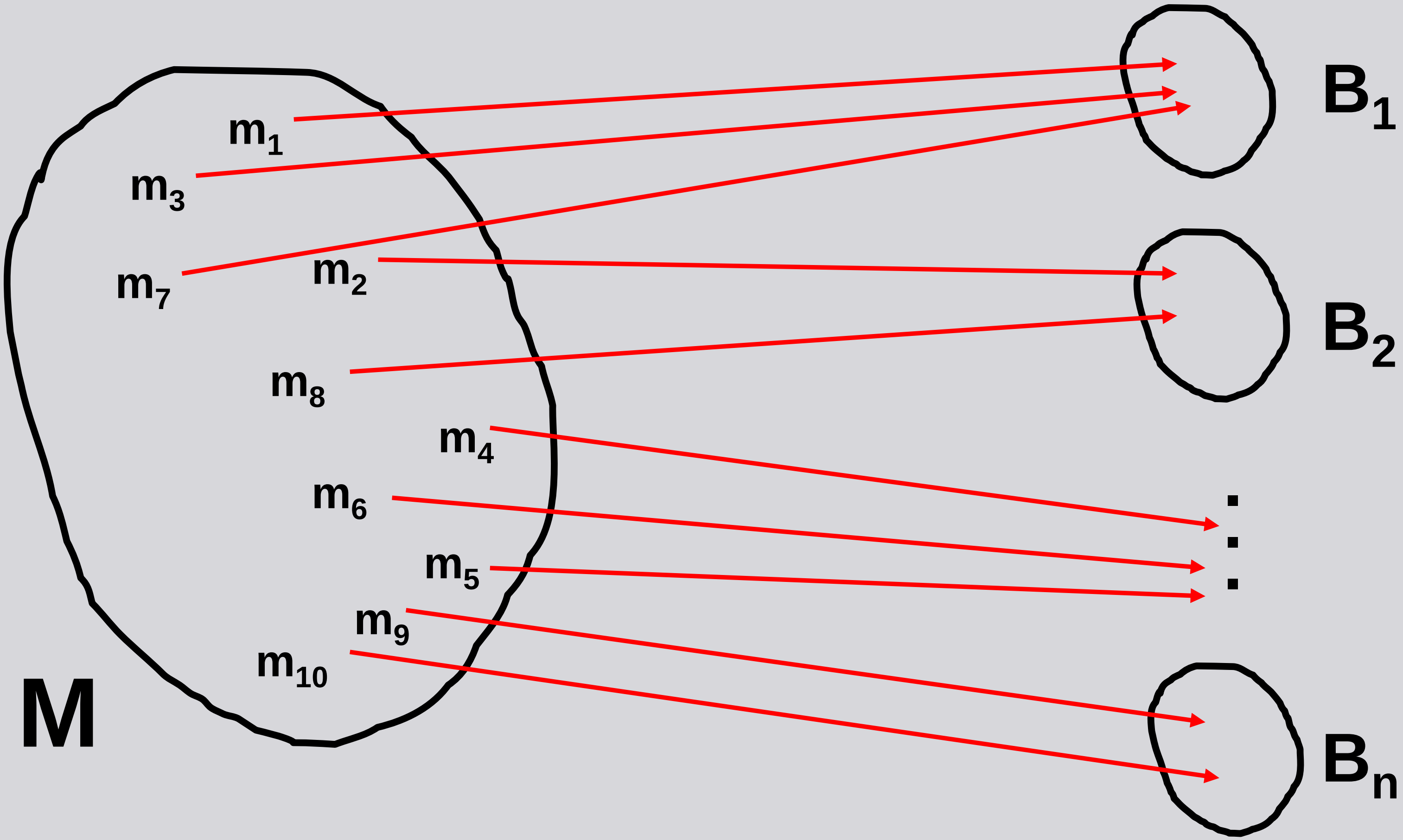
**Suchen in Feldern ➔ Hashverfahren**

***Definition Hash-Funktion*: Sei M eine Menge deren Elemente durch Schlüsselwerte S charakterisiert sind (d.h. jedes Element e aus M besitzt einen Schlüssel s). B sei eine endliche Menge von Behälten, in denen Elemente von M gespeichert werden sollen, mit |B| = n, n>0**

**Eine *Hash-Funktion* ist eine totale, d.h. überall definierte Funktion M → {1, ... n}**

# Hashverfahren

## Suchen in Feldern ➔ Hashverfahren



# Hashverfahren

Hochschule Deggendorf

**Suchen in Feldern ➔ Hashverfahren**

**Der berechnete Wert (Nummer des Behälters für e) eines Elements s aus M der Hashfunktion h(e) wird als *Hashwert* bezeichnet**

**Die Gesamtzahl der Behälter $B_1, ..., B_n$ wird als *Hashtabelle* bezeichnet**

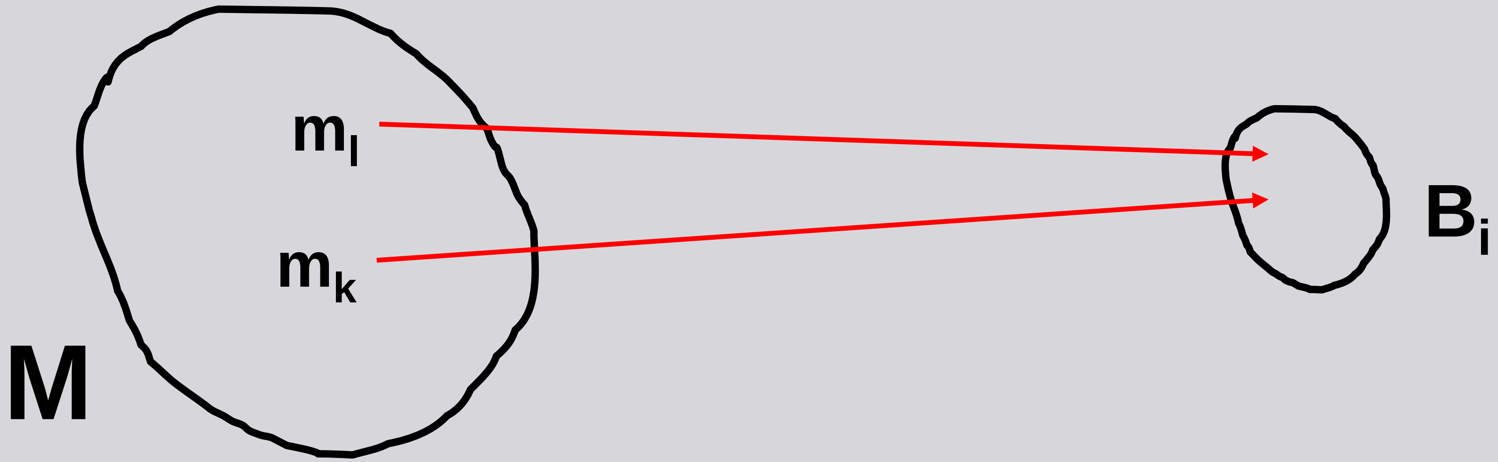
**Der Wert n/|M| definiert die *Schlüsseldichte***

**Der Wert n/m ist der *Belegungsfaktor* der Hash-Tabelle $B_0, . . . , B_{m-1}$.**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren**

**Haben unterschiedliche Elemente aus M den gleichen Hashwert, so wird dies als *Kollision* bezeichnet.**




**Kollisionen kommen häufig vor, da in der Regel |M| >> n**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren**

**Die Hash-Funktion h sollte die folgenden Eigenschaften haben:**

- **Sie sollte surjektiv sein, d.h. alle Behälter sollten belegt werden, d.h. es gibt zu jedem Behälter b ein x $\in$ M mit h(x) = b**
- **Die zu speichernden Schlüssel sollen möglichst gleichmäßig über alle Behälter verteilt werden. Jeder Behälter sollte möglichst mit gleicher Wahrscheinlichkeit belegt werden.**
- **Sie sollte „einfach“ zu berechnen sein.**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren**

**Beispiel:**

- **Die Wochentage Montag, ..., Sonntag sollen auf 7 Behälter $B_0$, ..., $B_6$ abgebildet werden. Als Hashfunktion eines Wochentags t wird folgende Funktion gewählt:**

  **h(t) = d(1. Buchstabe(t)) + d(2. Buchstabe(t))+ ... +d(6. Buchstabe(t)) mod 7,**

  **wobei d(c) := 1, falls c = ‘a‘ oder c = ‘A‘**
  **2, falls c = ‘b‘ oder c = ‘B‘**
  **...**
  **26, falls c = ‘z‘ oder c = ‘Z‘**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren**

**Beispiel:**

- **h(Montag) = 0**
- **h(Dienstag) = 1**
- **h(Mittwoch) = 2**
- **h(Donnerstag) = 0**
- **h(Freitag) = 3**
- **h(Samstag) = 3**
- **h(Sonntag) = 6**

**➔ Keine gleichmäßige Verteilung über die Behälter**

**➔ Kollisionen**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren**

**Wahrscheinlichkeit für Kollisionen?**

**Sei h : M $\rightarrow$ {1, . . . ,n} eine ideale Hash-Funktion, e $\in$ M.**

**Dann gilt zunächst : P(h(e) = i) = 1/n**

**und für eine Folge von k Schlüsseln, wobei k<n, gilt weiter:**

**P(Kollision(1, ... , k) = 1- P(keine Kollision(1, ... , k))**

(P steht hier für Wahrscheinlichkeit)

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren**

**P(keine Kollsision(1, ... , k)) = P(1) * P(2) * ... * P(k)**

**wobei P(i) die Wahrscheinlichkeit ist, dass der i-te Schlüssel einen freien Platz findet und alle vorherigen auch einen freien Platz gefunden haben (d.h. es sind keine Kollisionen aufgetreten)**

**Es gilt: P(1) = 1, P(2) = (n-1)/n, P(i) = (n-i+1)/n, daraus folgt**

$$P(\text{Kollision}(1, \ldots, k) = 1 - \frac{n(n-1) * \ldots (n-k+1)}{n^k}$$

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren**

**Beispiel:**

**Für die Anzahl der Tage eines Jahres n = 365 ergeben sich die folgenden Wahrscheinlichkeiten einer Kollision:**

**k = 22 : P(Kollision) ≈ 0, 475**

**k = 23 : P(Kollision) ≈ 0, 507**

**k = 50 : P(Kollision) ≈ 0, 970**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren**

**Dies ist das so genannte ”Geburtstagsparadoxon“:**

**Sind mehr als 23 Personen zusammen, so haben mit mehr als 50% Wahrscheinlichkeit mindestens zwei von ihnen am selben Tag Geburtstag.**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren**

**Für Hashverfahren bedeutet die obige Analyse, dass**

**1. Kollisionen praktisch nicht zu vermeiden sind!**
**2. Mit Kollisionen definiert umgegangen werden muss!**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren ➔ Behandlung von Kollisionen**

**Hashverfahren, bei denen ein Behälter (theoretisch) beliebig viele Elemente aufnehmen kann, heißen *<u>offene Hashverfahren</u>*.**

**(im Gegensatz zu *<u>geschlossenen Verfahren</u>*, bei denen jeder Behälter nur eine kleine feste Zahl von Elementen beherbergen kann.)**

# Hashverfahren

**Suchen in Feldern → Hashverfahren → Behandlung von Kollisionen, offene Verfahren**

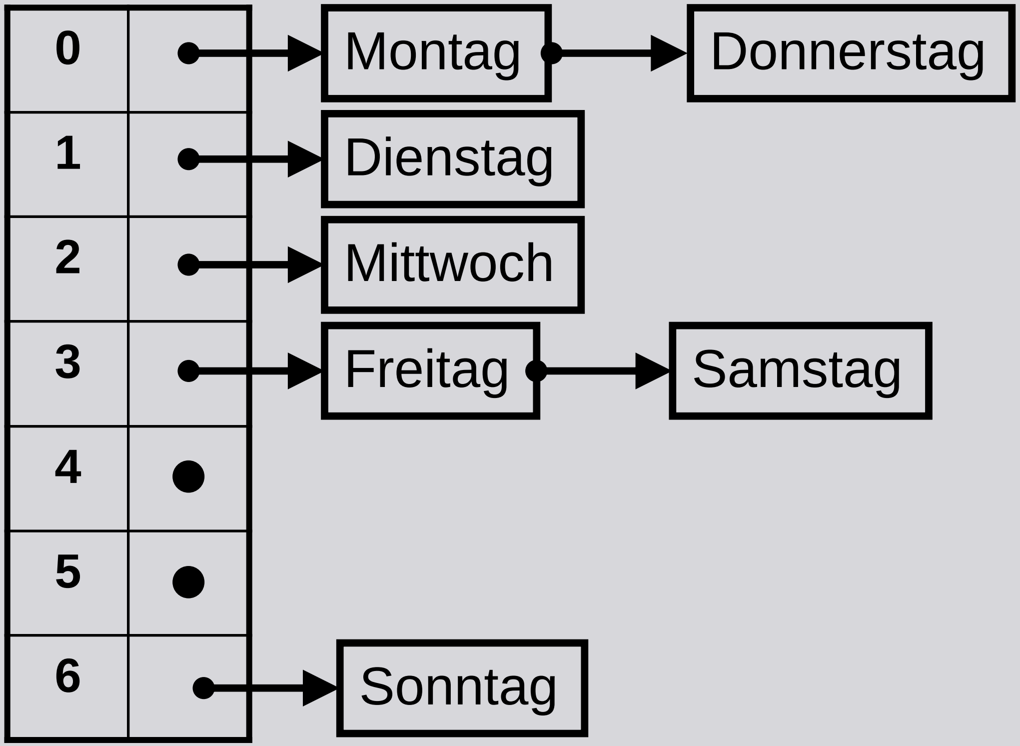
***Hashing mit Verkettung*: Als Behälter wird eine verkettete Liste verwendet, in die die Elemente gespeichert werden.**

# Hashverfahren

## Suchen in Feldern ➔ Hashverfahren ➔ Behandlung von Kollisionen, offene Verfahren

## ***Hashing mit Verkettung***: Beispiel Wochentage

h(Montag) = 0
h(Dienstag) = 1
h(Mittwoch) = 2
h(Donnerstag) = 0
h(Freitag) = 3
h(Samstag) = 3
h(Sonntag) = 6



# Hashverfahren

Hochschule Deggendorf

**Suchen in Feldern ➔ Hashverfahren ➔ offene Verfahren, Aufwand**

**Sei S = $\{x_1, \ldots, x_n\} \subseteq$ M eine zu speichernde Menge und sei HT eine offene Hash-Tabelle der Länge n mit Hash-Funktion h.**

**h(e) soll in konstanter Zeit ausgewertet werden für alle e $\in$ M.**

**Im Folgenden soll die Rechenzeit wird dann für Operationen Suchen, Einfügen und Löschen in der Hashtabelle betrachtet werden.**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren ➔ offene Verfahren, Aufwand**

**Für $0 \leq i < n$ sei HT[i] die Liste der Schlüssel $x_j$ , für die $h(x_j) = i$ gilt.**
**|HT[i]| sei die Länge der i-ten Liste.**

**Dann kostet jede Operation im schlechtesten Fall**
**max {O(|HT[h(x)]|) } viele Schritte (über alle möglichen x).**

**$O(|HT[h(x)]|) \leq n$, da maximal n Elemente in einer Liste gespeichert werden**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren ➔ offene Verfahren, Aufwand**

**Damit gilt, dass die Ausführung der Operationen Suchen, Einfügen und Löschen in einer Hash-Tabelle, in der eine Menge S mit n Elementen abgespeichert werden soll, im schlechtesten Fall einen Aufwand O(n) erfordert.**

**Anmerkung: In der Realität ist er Aufwand geringer, da die wahrscheinlichkeit, dass alle Elemente in einer Liste „landen", gering ist.**

# Hashverfahren

## Zum Schluss dieses Abschnitts …

**Noch Fragen ??**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren**

**Bei geschlossenen Hasverfahren kann jeder Behälter nur eine konstante Anzahl a ≥ 1 von Schlüsseln aufnehmen.**

**Daher ist die Behandlung von Kollisionen in der Regel komplexer (und daher wichtiger) als bei offenen Verfahren**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren**

**Im folgenden wird der Fall a = 1 betrachtet und Hash-Tabellen, die Schlüssel/Wert-Paare mit Schlüsseln vom Typ String und Werten vom Typ Object speichern. Dabei soll jedem Schlüssel höchstens ein Wert zugeordnet sein.**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren**

**Kennzeichnung der Felder der Hashtabelle mit einem booleschen Wert.**

**Unterscheidung :**

- **Behälter wurde noch nie getroffen (true)**
- **Ein Behälter wurde schon benutzt, ist aber wegen einer vorhergehenden Löschoperation leer. (false)**

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| | $O_2$ | $O_3$ | | | $O_6$ | $O_7$ | $O_8$ | $O_9$ | |
| t | f | f | t | f | f | f | f | f | t |

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren, Kollisionsbehandlung**

**Grundlegende Idee der Kollisionsbehandlung: *Rehashing*:**

- **Neben der „Haupt-“Hashfunktion $h_0$ werden weitere Hashfunktionen $h_1$, ... ,$h_l$ benutzt.**
- **Für einen Schlüssel x werden dann nacheinander die Behälter $h_0(x)$, $h_1(x)$, . . . , $h_l(x)$ angeschaut. Sobald ein freier Behälter gefunden wird, kann das Element gespeichert werden.**

# Hashverfahren



**Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren, Kollisionsbehandlung**

**Problem: Das Auftreten einer freien Zelle für $h_i(x)$ besagt nicht, dass x nicht in schon in der Hash-Tabelle enthalten gewesen ist. ➔ Markieren der gelöschten Paare**

# Hashverfahren

## Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren, Kollisionsbehandlung

Schlüssel s

$h_2(s) = 1$
➔ freies Feld, noch nicht belegt
➔ Speichern

$h_0(s) = 6$
➔ 1. Kollision

$h_1(s) = 9$ ➔ 2. Kollision

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
|  | f | e | n | k | y |  | w | z | a |
| t | f | f | f | f | f | f | f | f | f |

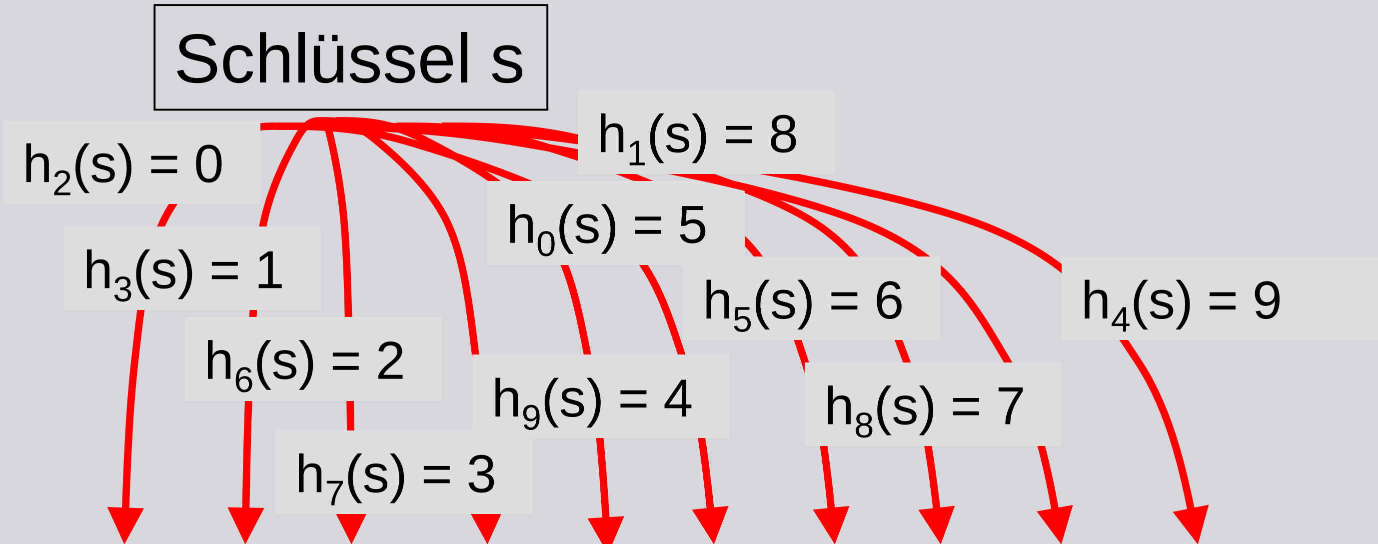
# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren, Kollisionsbehandlung**

**Analog zu den offenen Hashverfahren soll die Folge der Hashfunktionen $h_0, \ldots, h_{n-1}$ so festgelegt werden, dass für jeden Schlüsselwert s sämtliche Behälter HT[i] ($0 \leq i < n$) erreicht werden. D.h. es gibt eine „gleichmäßige“ Verteilung über alle Behälter.**

# Hashverfahren

## Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren, Kollisionsbehandlung




| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| | f | e | n | k | y | | w | z | a |
| t | f | f | f | f | f | f | f | f | f |

# Hashverfahren

Hochschule Deggendorf

**Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren, Kollisionsbehandlung**

**Wahl der Hashfunktionen $h_i(x)$:**

**$h_i(x) := (h(x) + i) \bmod n$**

**Diese einfachste Art der Festlegung wird als *<u>lineares Sondieren</u>* (*<u>linear probing</u>*) bezeichnet**

# Hashverfahren

Hochschule Deggendorf

## Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren, Beispiel Wochentage mit linearem Sondieren:

$h_0$(Montag) = 0
$h_0$ (Dienstag) = 1
$h_0$ (Mittwoch) = 2
$h_0$ (Donnerstag) = 0 ➔ Kollision
$h_1$(Donnerstag = 1 ➔ Kollision
$h_2$(Donnerstag) = 2 ➔ Kollision
$h_3$(Donnerstag) = 3
$h_0$ (Freitag) = 3 ➔ Kollision
$h_1$(Freitag) = 4
$h_0$ (Samstag) = 3 ➔ Kollision
$h_1$(Samstag) = 4 ➔ Kollision
$h_2$(Samstag) = 5
$h_0$ (Sonntag) = 6

| | |
|---|---|
| **0** | Montag |
| **1** | Dienstag |
| **2** | Mittwoch |
| **3** | Donnerstag |
| **4** | Freitag |
| **5** | Samstag |
| **6** | Sonntag |

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren, Kollisionsbehandlung**

**Lineares Sondieren, Eigenschaften**

- **Verschieben „kollidierender“ Elemente auf den nächsten freien Behälter.**
- **Bei k hintereinander belegten Behältern gilt:**
  **Die Wahrscheinlichkeit, dass der erste freie Behälter nach diesen k Behältern belegt wird, ist mit (k + 1)/m wesentlich größer als die Wahrscheinlichkeit, dass ein Behälter im nächsten Schritt belegt wird, dessen Vorgänger noch frei ist. Dadurch entstehen beim linearen Sondieren „Ketten“ belegter Behälter.**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren, Kollisionsbehandlung**

**Lineares Sondieren, Eigenschaften**

- **Sei $\alpha := n/m$ der Belegungsfaktor einer Hash-Tabelle mit m Behältern, von denen n belegt sind. Beim Hashing mit linearem Sondieren entstehen für eine Suchoperation durchschnittlich folgende Kosten:**
  - **$(1 + 1/(1 - \alpha))/2$ beim erfolgreichen Suchen**
  - **$1 + 1/(1 - \alpha)^2)/2$ beim erfolglosen Suchen.**

**(Knuth 1973)**

# Hashverfahren

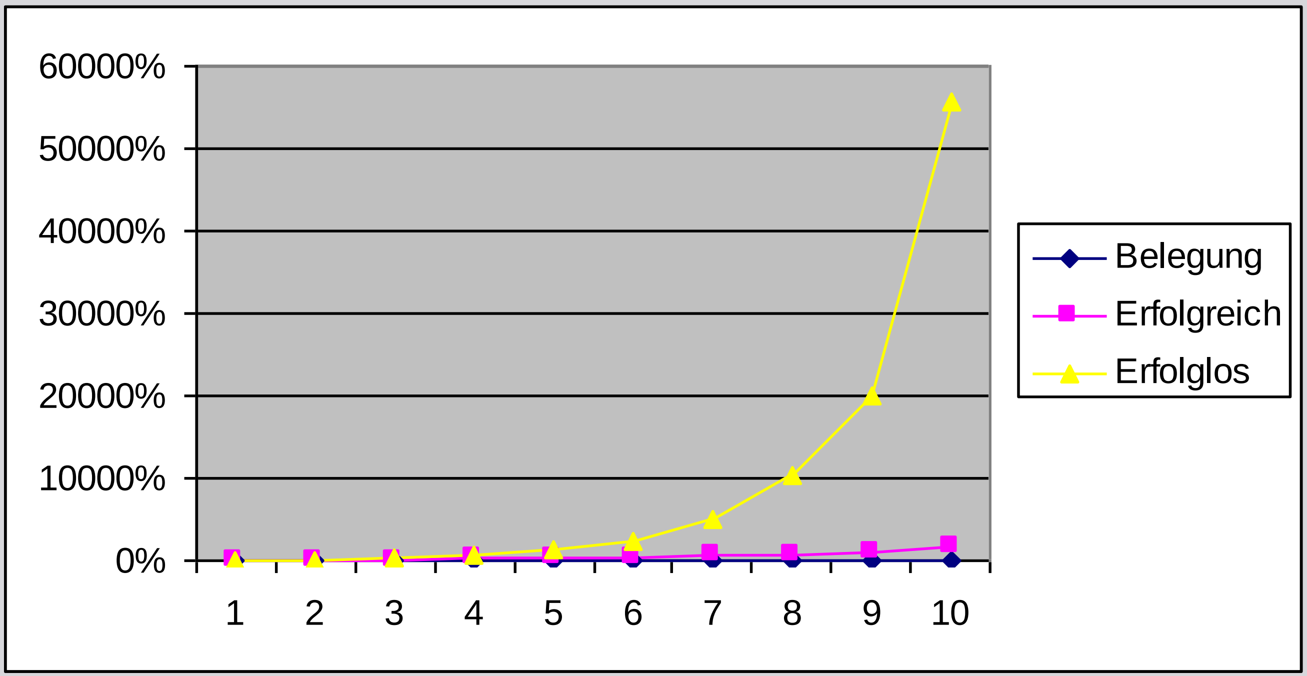
## Suchen in Feldern → Hashverfahren → geschlossene Verfahren, Kollisionsbehandlung
## Lineares Sondieren, Beispiel für Aufwand für Suchen

| Belegung | Aufwand erfolgreiches Suchen | Aufwand erfolgloses Suchen |
|---|---|---|
| 10% | 1,06 | 1,12 |
| 25% | 1,17 | 1,39 |
| 50% | 1,5 | 2,5 |
| 70% | 2,17 | 6,06 |
| 80% | 3,00 | 13,00 |
| 85% | 3,83 | 22,72 |
| 90% | 5,50 | 50,50 |
| 93% | 7,64 | 102,54 |
| 95% | 10,50 | 200,50 |
| 97% | 17,17 | 556,06 |

# Hashverfahren

## Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren, Kollisionsbehandlung

## Lineares Sondieren, Beispiel für Aufwand für Suchen



# Hashverfahren

**Suchen in Feldern → Hashverfahren → geschlossene Verfahren, Kollisionsbehandlung**

**Lineares Sondieren, Folgerung:**

- **Bei Belegung einer Hashtabelle wird Suchen mittels linearem Sondieren ineffizient.**

**→ Alternative Hashverfahren betrachten!**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren, Kollisionsbehandlung**

***Verallgemeinertes lineares Sondieren*:**

**$h_i(x) = (h(x) + c \cdot i) \bmod m$.**

**c ist dabei eine ganzzahlige Konstante (>0), die zu m teilerfremd sein muss, um alle Behälter zu erreichen.**

# Hashverfahren

**Suchen in Feldern → Hashverfahren → geschlossene Verfahren, Kollisionsbehandlung**

**<u>Quadratisches Sondieren</u>:**

$$h_i(x) = (h(x) + i^2) \bmod m$$

**oder für $1 \leq i \leq (m - 1)/2$**

$$h_{2i-1}(x) = (h(x) + i^2) \bmod m,$$

$$h_{2i}(x) = (h(x) - i^2) \bmod m.$$

**(Wählt man bei dieser Variante $m = 4j + 3$ als Primzahl, so wird jeder Behälter getroffen.)**

# Hashverfahren

**Suchen in Feldern ➔ Hashverfahren ➔ geschlossene Verfahren, Kollisionsbehandlung**

***<u>Doppel-Hashing</u>*:**

**Seien h, $h^*$ : M → {0, . . . ,n− 1} zwei Hash-Funktionen.**

**Dabei seien h und $h^*$ so definiert, dass für beide eine Kollision nur mit Wahrscheinlichkeit 1/n auftritt,**

**d.h. $P(h(x) = h(y)) = P(h^*(x) = h^*(y)) = 1/n$**

# Hashverfahren

**Suchen in Feldern → Hashverfahren → geschlossene Verfahren, Kollisionsbehandlung**

***Doppel-Hashing*:**

**Die Funktionen h und $h^*$ heißen *unabhängig*, wenn eine Doppelkollision nur mit Wahrscheinlichkeit $1/n^2$ auftritt, d.h. $P(h(x) = h(y) \text{ und } h^*(x) = h^*(y)) = 1/n^2$**

# Hashverfahren

**Suchen in Feldern → Hashverfahren → geschlossene Verfahren, Kollisionsbehandlung**

***Doppel-Hashing*:**

**Eine Folge von Hash-Funktionen wird wie folgt definiert:**

**Sei $i \geq 1$, dann ist**

$$h_i(x) = (h(x) + h^*(x) \cdot i^2) \bmod n.$$

**Problem: Paare von Funktionen finden, die unabhängig sind.**

# Hashverfahren

## Zum Schluss dieses Abschnitts …

**Noch Fragen ??**

# Hashverfahren

## Hashfunktionen

**In folgenden sollen verschiedene Hashfunktionen vorgestellt werden.**

**Sei M eine Menge und nat: M → N eine Funktion von M in die Menge der natürlichen Zahlen. Dann ist**

**h(m) = nat(m) mod n eine Hashfunktion (für n Behälter)**

# Hashverfahren

**Hashfunktionen**

**1.) Bildung von Hashwerten aus Wörtern:**

**Sei W die Menge der Wörter aus dem Alphabet A = {a, b, ... , z}.**

**Dann kann ein Wort $w = a_1 a_2 \ldots a_n$ ($a_i \in A$ ) als eine Zahl**

**$nat(w) = wert(a_1) * 26^{n-1} + wert(a_2) * 26^{n-2} + \cdots + wert(a_{n-1}) * 26 + wert(a_n)$**

**aufgefasst werden, wenn man wert(a) = 0, wert(b) = 1, . . . , wert(z) = 25 setzt.**

# Hashverfahren

Hochschule Deggendorf

## Hashfunktionen

**1.) Einfache Bildung von Hashwerten aus Zahlen:**
**Sei M $\subseteq \mathbb{N}$, n eine Menge von Behältern, dann sei**

**h(x) = x mod n**

- **Alle Behälter werden erfaßt**
- **Nacheinander folgende Schlüssel landen in aufeinanderfolgende Behälter → Probleme beim Sondieren**

# Hashverfahren

**Hashfunktionen**

**2.) Bildung von Hashwerten aus Wörtern:**

**Sei W die Menge der Wörter aus dem Alphabet A = {a, b, ... , z}.**

**Dann kann ein Wort $w = a_1 a_2 \ldots a_n$ ($a_i \in A$ ) als eine Zahl**

**$nat(w) = wert(a_1) * 26^{n-1} + wert(a_2) * 26^{n-2} + \cdot \cdot \cdot + wert(a_{n-1}) *\cdot\ 26 + wert(a_n)$**

**aufgefasst werden, wenn man wert(a) = 0, wert(b) = 1, . . . , wert(z) = 25 setzt.**

# Hashverfahren

**Hashfunktionen**

**3.) Bildung von Hashwerten aus Zahlen:**

**Die *Mittel-Quadrat-Methode*: Sei $U \subseteq N$ und sei**

$$k = \sum_{i=0}^{l} z_i * 10^i$$

**k wird durch die Ziffernfolge $z_l z_{l-1} \ldots z_0$ beschrieben.**
**Den Wert h(k) erhält man dadurch „Herausgreifen" eines hinreichend großen Blocks aus der Mitte der Ziffernfolge von $k^2$.**

**Die mittleren Ziffern von $k^2$ hängen von allen Ziffern von k ab → gute Streuung von aufeinander folgenden Werten von k.**

# Hashverfahren

## Hashfunktionen

**3.) Bildung von Hashwerten aus Zahlen:**

**Die *Mittel-Quadrat-Methode*: Beispiel**

**Sei n = 100 (Anzahl der Behälter)**

| k | K mod 100 | $k^2$ | H(k) |
|---|---|---|---|
| 130 | 30 | 16<u>90</u>0 | 90 |
| 131 | 31 | 17<u>16</u>1 | 16 |
| 132 | 32 | 17<u>42</u>4 | 42 |

## Hashverfahren

Hochschule Deggendorf

# Zum Schluss dieses Abschnitts …

**Noch Fragen ??**