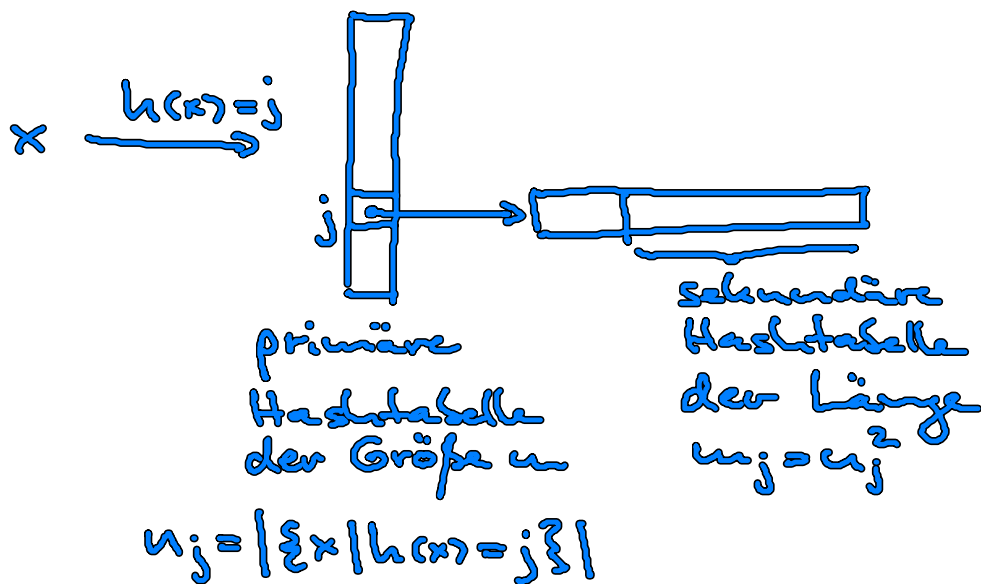


# Perfektes Hashing



$h(x) = ((a \cdot x + b) \bmod p) \bmod m$  primäre Hashfkt.

$h_j(x) = ((a_j \cdot x + b_j) \bmod p) \bmod m_j$  sekundäre Hashfkt.

Satz: Die Wahrscheinlichkeit für das Auftreten eines Konflikts in einer sekundären Hashtafel ist bei zufälliger Wahl von  $a_j$  und  $b_j$  kleiner als  $\frac{1}{2}$ .

Frage B: Wieviel Speicherplatz wird insgesamt benötigt?

Satz:  $n$  Schlüssel werden mit zufälliger Hashfkt.  $h \in \mathbb{H}_{p,m}$  in eine Hashtafel der Größe  $m = n$  gespeichert. Dann gilt:

$$E\left(\sum_{j=0}^{m-1} u_j^2\right) < 2 \cdot n$$

wobei  $u_j := |\{x \mid h(x) = j\}|$ .

Beweis: Für  $a \in \mathbb{N}$  gilt:  $a^2 = a + 2 \cdot \binom{a}{2}$

$$E\left(\sum_{j=0}^{m-1} u_j^2\right) = E\left(\sum_{j=0}^{m-1} \left(u_j + 2 \cdot \binom{u_j}{2}\right)\right)$$

$$= E\left(\underbrace{\sum_{j=0}^{u-1} u_j}_u\right) + 2 \cdot E\left(\sum_{j=0}^{u-1} \binom{u_j}{2}\right)$$

$$= u + 2 \cdot E\left(\sum_{j=0}^{u-1} \underbrace{\binom{u_j}{2}}_{\substack{= \# \text{ Kollisionen} \\ \text{an Stelle } j}}\right)$$

$\underbrace{\hspace{10em}}_{= \# \text{ Kollisionen insgesamt}}$

$$\leq u + 2 \cdot \binom{u}{2} \cdot \frac{1}{u}$$

$$= u + u \cdot (u-1) \cdot \frac{1}{u} = 2u - 1 < 2u. \quad \square$$

Korollar: Für das zweistufige Hash-Verf. mit primärer Länge  $u$  und sek. Länge  $u_j = u_j^2$  gilt: Der erwartete Speicherbedarf ist linear in  $u$ , d.h.  $O(u)$ .

Satz: Für das zweistufige Hash-Verf. gilt:

$$\Pr(\text{Speicherbedarf für sek. Hashstab.} \geq 4u) < \frac{1}{2}.$$

Beweis: Markov'sche Ungleichung.  $\square$

$\Rightarrow$  Durch wiederholtes Ziehen einer

Zufälligen primären Hashwert. In kann man "garantieren", dass der Speicherbedarf  $\leq 4n$  ist.

---

## Neues Kapitel:

### Berechenbarkeit und Komplexität

Literatur hierzu:

Ingo Wegener: Theoretische Informatik  
— eine algorithmorientierte Einführung  
(Teubner), Kap. 1-3

Frage: Welche Probleme lassen sich mit Computern lösen.

— Was ist ein Computer?

Turingmaschinen, Registermaschinen,  
Churchsche These

— Was verstehen wir unter einem "Problem"?

Eingabe  $\rightarrow$  Computer  $\rightarrow$  Ausgabe  
 $\in \Sigma^*$   $\in \Sigma^*$

$\Sigma$  ist endliches „Alphabet“, z.B.  $\Sigma = \{0,1\}$

$\Sigma^* := \bigcup_{i=1}^{\infty} \Sigma^i$  Menge der Strings endlicher Länge über  $\Sigma$ .

Problem: Relation  $R$  auf  $\Sigma^* \times \Sigma^*$

$(x,y) \in R \Leftrightarrow y \in \Sigma^*$  ist zulässige Ausgabe zur Eingabe  $x \in \Sigma^*$ .

Spezialfall davon: Funktion  $f: \Sigma^* \rightarrow \Sigma^*$   
gegeben  $x \in \Sigma^*$ , finde  $y := f(x) \in \Sigma^*$

Spezialfall hiervon:

Entscheidungsprobleme:  $f: \Sigma^* \rightarrow \{0,1\}$   
gegeben  $x \in \Sigma^*$ , entscheide ob  $f(x) = 1$

$\leftrightarrow$  „Sprache“  $L := f^{-1}(1) \subseteq \Sigma^*$

Es gibt unlösbare Probleme!

Denn: Computerprogramme  $\subseteq \{0,1\}^* (\cong \mathbb{Q})$

$\Rightarrow$  Es gibt „nur“ abzählbar unendlich viele Computerprogramme.

Probleme  $\supseteq \{0,1\}^* (\cong \mathbb{R})$

überabzählbar unendlich

(Einschub: „überabz.  $\infty > abz. \infty$ “)

0	0, <sup>0</sup> 1735489....
1	0, <sup>2</sup> 2385491....
2	0, <sup>4</sup> 8765432
3	0, <sup>6</sup> 3135724
4	0, <sup>7</sup> 2234331
⋮	⋮

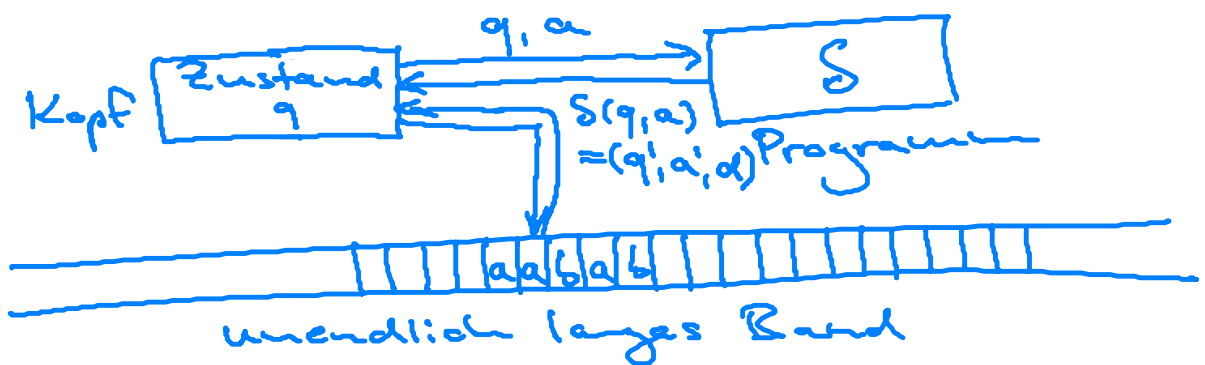
→ fast alle Probleme sind unlösbar!  
konkrete Beispiele später!

Weitere Frage: Welche Probleme lassen sich effizient lösen?

→ Komplexitätsklassen P und NP

## Turingmaschinen

DTM  $\hat{=}$  deterministische Turingmaschine



$Q$  endliche Zustandsmenge

$\Sigma$  endliches Eingabealphabet

$\Gamma \supseteq \Sigma$  endliches Bandalphabet

$B \in \Gamma \setminus \Sigma$  Leerzeichen

$q_0 \in Q$  Anfangszustand

$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{R, L, N\}$

Zustandsübergangsfkt.

$F \subseteq Q$  Menge der akzeptierenden  
Endzustände

$q \in Q$  heißt Endzustand, falls

$$\delta(q, a) = (q, a, N) \quad \forall a \in \Gamma$$

Def. 1: (i)  $f: \Sigma^* \rightarrow \Sigma^*$  heißt total rekursiv

(berechenbar), falls  $\Leftrightarrow$  eine DTM gibt,

die zur Eingabe  $x \in \Sigma^*$  die Ausgabe  
 $f(x) \in \Sigma^*$  erzeugt.

(ii)  $f: \mathbb{N}^k \rightarrow \mathbb{N}$  heißt total rekursiv, falls  
es eine DTM gibt, die zur Eingabe

$$\text{bin}(i_1) \# \text{bin}(i_2) \# \text{bin}(i_3) \# \dots \# \text{bin}(i_k) \#$$

die Ausgabe

$$\text{bin}(m)$$

erzeugt, wobei  $u = f(i_1, i_2, \dots, i_k)$ .

Erläuterung von (i)

Zu Beginn: 

B	B	B	$x_1$	$x_2$	$x_3$	...	$x_n$	B	B	...
---	---	---	-------	-------	-------	-----	-------	---	---	-----

am Ende: 

...	B	B	$f(x)$	B	B	...
-----	---	---	--------	---	---	-----