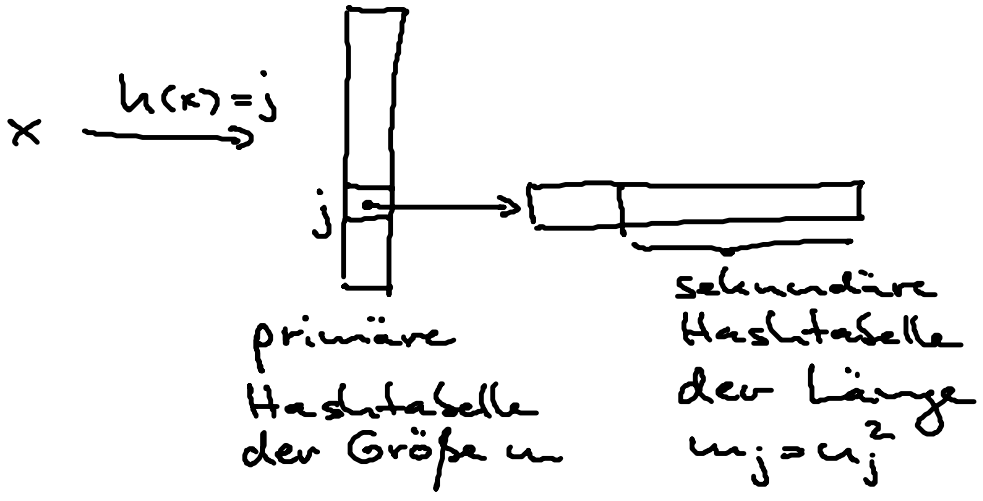
Perfektes Hashing



$$n_j = |\{x \mid h(x) = j\}|$$

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

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

<u>Satz:</u> Die Wahrscheinlichkeit für das Auftreten eines Konfliktes in einer sekundären Hashtabelle ist bei zufälliger Wahl von a_j und b_j kleiner als $\frac{1}{2}$.

Frage B: Wieviel Speicherplatz wird insgesamt benötigt?

<u>Satz:</u> n Schlüssel werden mit zufälliger Hashfkt. $h \in H_{p,m}$ in eine Hashtabelle der Größe $m = n$ gespeichert. Dann gilt:

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

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

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

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

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

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

$$\underbrace{\qquad\qquad}_{= \#\text{Kollisionen insgesamt}}$$

$$\leq n + 2 \cdot \binom{n}{2} \cdot \frac{1}{m}$$

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

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

<u>Satz:</u> Für das zweistufige Hash-Verf. gilt:

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

<u>Beweis:</u> Markov'sche Ungleichung. $\square$

$\Rightarrow$ Durch wiederholtes Ziehen einer

zufälligen primären Hashfkt. In kann man „garantieren", dass der Speicherbedarf $\leq 4n$ ist.

Neues Kapitel:

Berechenbarkeit und Komplexität

Literatur hierzu:

Ingo Wegener: Theoretische Informatik – eine algorithmenorientierte 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 $\in \Sigma^*$ → Computer → Ausgabe $\in \Sigma^*$

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

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

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 $f: \Sigma^* \to \Sigma^*$

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

Spezialfall hiervon:

Entscheidungsproblem: $f: \Sigma^* \to \{0,1\}$

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

überabzählbar unendlich

(Einschub: „überabz. ∞ > abz. ∞“

	$0,$
0	$0,2173548\overline{9}\ldots$
1	$0,23\overline{4}8549\overline{1}\ldots$
2	$0,876\overline{7}5432$
3	$0,3135\overline{6}724$
4	$0,22343\overline{4}31$
⋮	⋮

)

—> fast alle Probleme sind unlösbar!

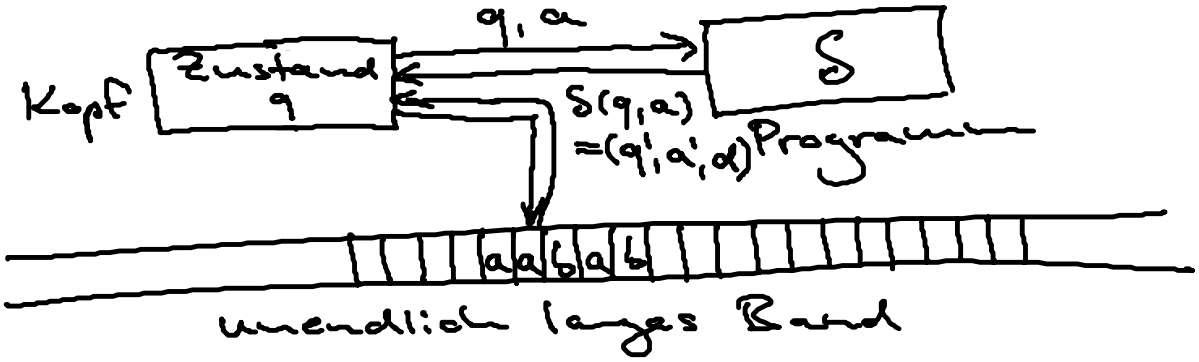
konkrete Beispiele später!

Weitere Frage: Welche Probleme lassen sich effizient lösen?

—> Komplexitätsklassen P und NP

Turingmaschinen

DTM ≙ deterministische Turingmaschine



Q endliche Zustandsmenge

Σ endliches Eingabealphabet

$\Gamma \supseteq \Sigma$ endliches Bandalphabet

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

$q_0 \in Q$ Anfangszustand

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L, N\}$
Zustandsüberführungsfkt.

$F \subseteq Q$ Menge der akzeptierenden Endzustände

$q \in Q$ heißt Endzustand, falls
$\delta(q, a) = (q, a, N) \quad \forall a \in \Gamma$

Def. 1: (i) $f : \Sigma^* \to \Sigma^*$ heißt **total rekursiv** (**berechenbar**), falls $\overset{\text{def}}{\Leftrightarrow}$ eine DTM gibt, die zur Eingabe $x \in \Sigma^*$ die Ausgabe $f(x) \in \Sigma^*$ erzeugt.

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

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

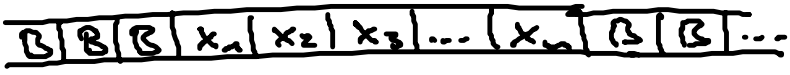
die Ausgabe

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

erzeugt, wobei $m = f(i_1, i_2, \ldots, i_k)$.

Erläuterung von $\langle i \rangle$

Zu Beginn:



am Ende: