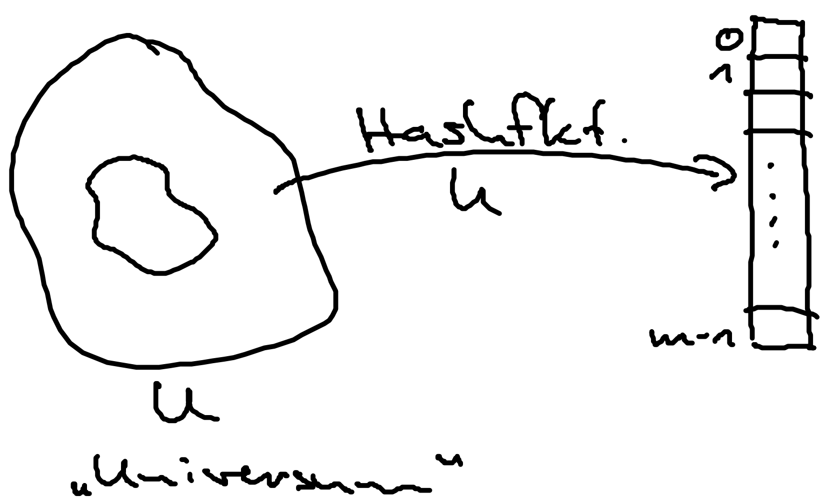
Hashing



$$h: U \to \{0, \dots, m-1\}$$

Problem: $|U|$ sehr groß, insbesondere
$m \ll |U|$

d.h. h nicht injektiv, also $h(k_1) = h(k_2)$
für $k_1 \neq k_2$.

Kollision

Umgang mit Kollisionen:

1) Hashing mit Überlauf („chaining")

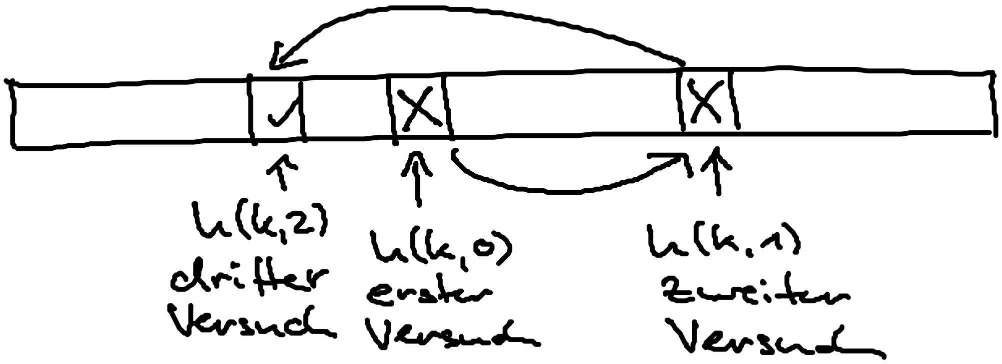
2) <u>Hashing mit Ersatzadressen</u>
(„open addressing")

<u>Idee</u>:

- speichere alle Datensätze in der Hashtabelle (d.h. keine Überlauf-Listen)
- $\Rightarrow m \geq n$
- Bei Kollisionen muss Ersatzadresse gesucht werden.

$$h : \underset{\substack{\uparrow \\ \text{Schlüssel}}}{U} \times \underset{\substack{\uparrow \\ \text{\# erfolgloser Versuche bei Suche nach freiem Eintrag}}}{\{0, 1, \ldots, m-1\}} \to \underset{\substack{\uparrow \\ \text{Adresse}}}{\{0, 1, \ldots, m-1\}}$$

$h(k,i) \hat{=} (i+1)$-ter Versuch, eine freie Adresse für k zu finden.



Dieses Verfahren berechnet also nacheinander die Adressen

$$h(k,0), h(k,1), \ldots, h(k,i), \ldots$$

bis eine freie Adresse gefunden wird.

Damit auf diese Art immer eine freie Adresse gefunden wird (unter der Annahme, dass $n \leq m$), muss gelten:

$$\boxed{\begin{array}{l} h(k,0), h(k,1), \ldots, h(k,m-1) \\ \text{ist Permutation von } 0,1,\ldots,m-1 \end{array}}$$

„Permutationsbedingung"

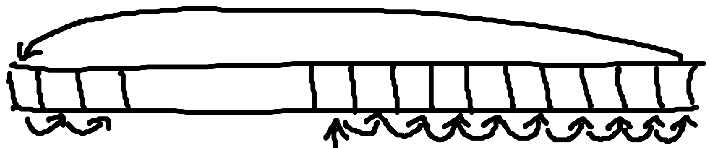
<u>Achtung</u>: Dies erschwert das Löschen!

<u>Beispiele</u> für open addressing:

- Lineares Sondieren

$h(k,i) := (h'(k) + i) \bmod m$

$h'(k)$: gewöhnliche Hashfkt. $\widehat{=} h(k,0)$



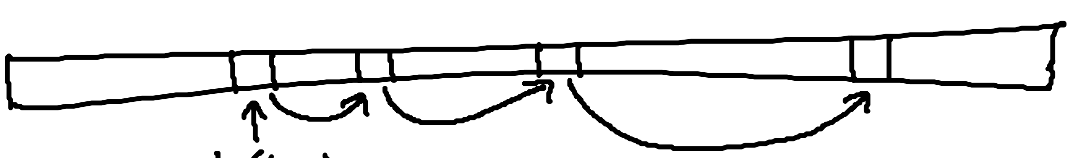
Permutationsbedingung offensichtlich erfüllt!

<u>Nachteil:</u> „primäres Clustering", d.h. es bilden sich immer größere Blöcke konsekutiver belegter Adressen, denn:

Falls die Adressen $a, a+1, a+2, \ldots, a+i-1$ bereits belegt sind und als nächstes ein neues Datenelement mit zufälligem Schlüssel eingefügt wird, so landet dieses mit W'keit $\frac{i+1}{m}$ an der Stelle $a+i$.

<u>Quadratisches Sondieren:</u>

$h(k,i) = (h'(k) + c_1 \cdot i + c_2 \cdot i^2) \bmod m$



Sprungweite nimmt quadratisch zu!

<u>Beachte:</u> Permutationsbedingung hängt von m, c_1, c_2 ab und muss im Einzelfall überprüft werden.

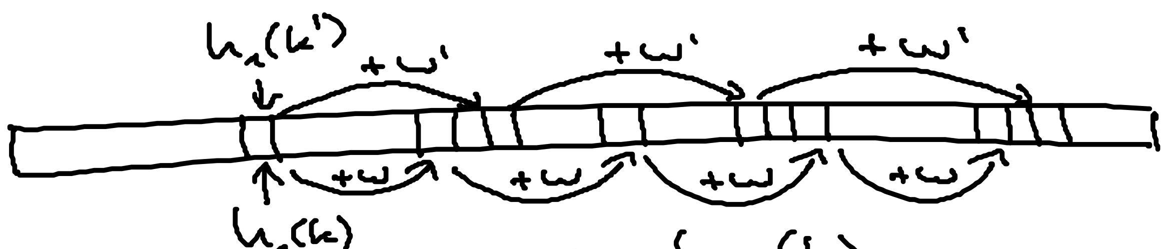
<u>Vorteil:</u> kein primäres Clustering!

<u>Nachteil:</u> sekundäres Clustering:
Schlüssel, die auf dieselbe erste Adresse abgebildet werden haben auch dieselbe Folge von Ersatzadressen.

<u>Doppel-Hash:</u>

$$h(k,i) = (h_1(k) + i \cdot h_2(k)) \bmod m$$

h_1, h_2 herkömmliche Hashfkt.

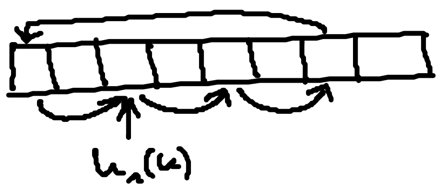


$w = h_2(k)$
$w' = h_2(k')$

vermeidet primäres und sekundäres Clustering, d.h.
$h_1(k) = h_1(k')$ führt zu unterschiedl. Ersatzadressen, falls $h_2(k) \neq h_2(k')$

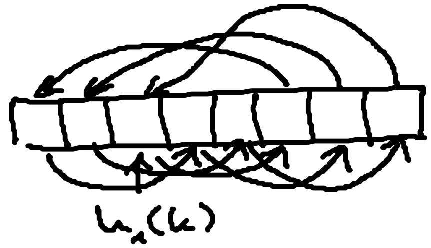
<u>Frage:</u> Wie erfüllt man die Permutationsbedingung?

Bsp.: $m = 8$



$h_2(k) = 2$

Perm. Bed. verletzt



$h_2(k) = 3$

Perm. Bed. erfüllt

Satz: Doppelhash erfüllt die Permutations-Bedingung $\Longleftrightarrow$

$\forall k$ sind m und $h_2(k)$ relativ prim

d.h. $ggT(m, h_2(k)) = 1$.

Bemerkung:

- Diese Bedingung ist immer erfüllt, wenn m Primzahl ist.
- Sie ist z.B. auch erfüllt, wenn $m = 2^q$ und $h_2(k)$ ungerade für alle k.

Beweis: „$\Longrightarrow$": Ann.: $\exists k$, so dass

$$ggT(m, h_2(k)) = d > 1$$

$$\Rightarrow \quad m = p \cdot d, \quad h_2(k) = q \cdot d$$

$$\Rightarrow \quad p \cdot h_2(k) = q \cdot p \cdot d = q \cdot m \equiv 0 \mod m$$

d.h. die p-te Ersatzadresse für k stimmt überein mit $h_1(k)$. Beachte, dass $p < m$

$\Rightarrow$ nicht alle Adressen besucht

$\Rightarrow$ Perm. bed. verletzt $\lightning$

„$\Leftarrow$" Ann: Perm. bed. nicht erfüllt, d.h.:

$\exists k$ und $0 \le i < j \le m-1$ mit

$$h_1(k) + i \cdot h_2(k) \equiv h_1(k) + j \cdot h_2(k) \mod m$$

$$\Leftrightarrow \quad (j-i) \cdot h_2(k) \equiv 0 \mod m$$

$$\Leftrightarrow \quad (j-i) \cdot h_2(k) = q \cdot m \quad , q \in \mathbb{N}$$

$$\Leftrightarrow \quad h_2(k) = \frac{q \cdot m}{j-i} \quad , \quad q \in \mathbb{N}$$

Da $j-i < m$ gilt also $ggT(h_2(k), m) > 1$ $\lightning$

$\square$

Analyse des „open addressing"

Einfügen: Wie viele Sondierungen müssen durchgeführt werden bis eine freie Adresse gefunden wird?

Beantworte diese Frage unter der Gleichverteilungsannahme

d.h.: nächste Sondierung wählt immer gleichverteilt unter den m Adressen, also wird jede Adresse mit W'keit $\frac{1}{m}$ gewählt.

Satz: Bei Auslastung $\alpha = \frac{n}{m} < 1$ ist die erwartete Anzahl Sondierungen beim Einfügen höchstens $\frac{1}{1-\alpha}$.

Bsp.: $\alpha = \frac{1}{2} \Rightarrow \leq 2$ Sondierungen im Erw. wert

$\alpha = 0{,}8 \Rightarrow \leq 5$ " " "

Beweis beruht auf etwas W'-Theorie...