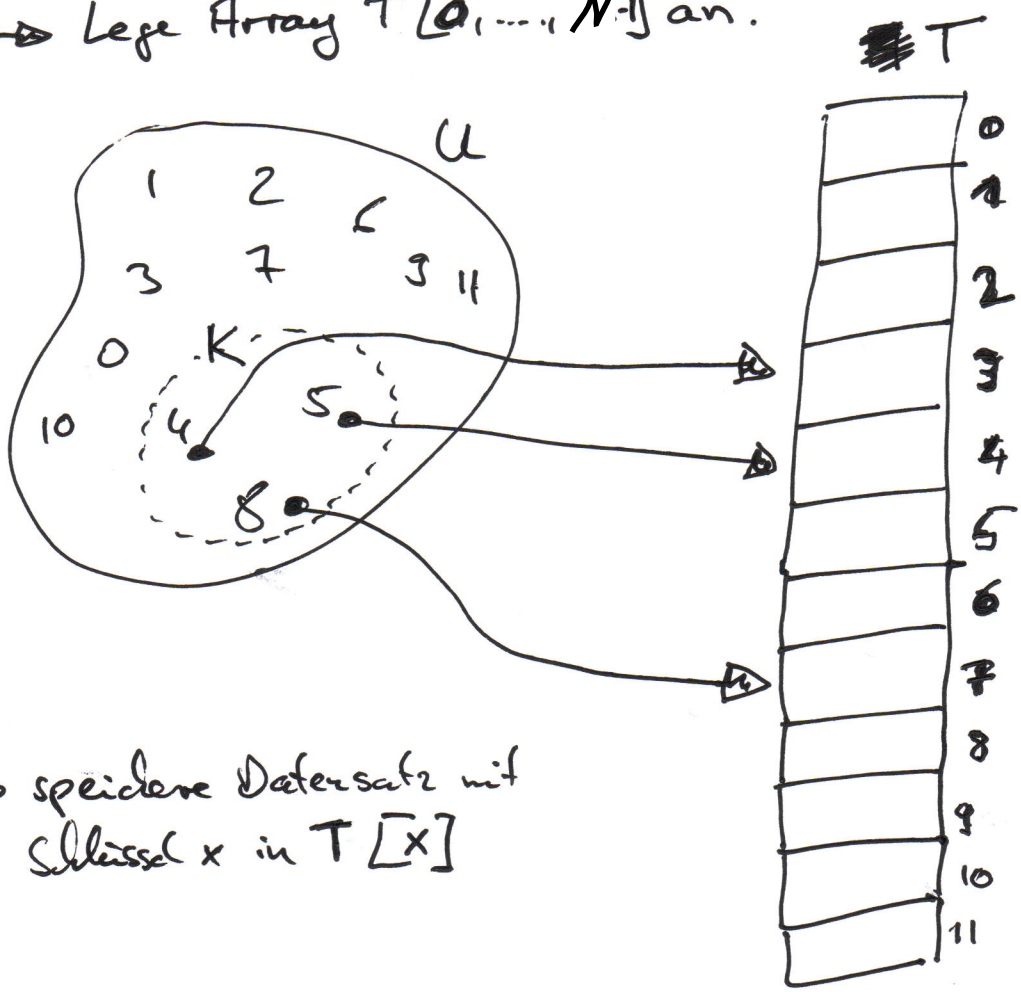
6. Hashing

Scenario:

- Datensätze mit eindeutigem Schlüssel
- Schlüssel $x \in U = \{0, \ldots, N-1\}$, K die Menge der tatsächlich verwendeten Schlüssel
- Funktionen
  - search(x)
  - insert(x)
  - delete(x)

6.1 Direkte Adressierung

↳ Lege Array $T[0, \ldots, N-1]$ an.



↳ speichere Datensatz mit Schlüssel x in $T[x]$

Funktionen

- search(x)
  Return T(x)
- insert(x)
  T(x) = Daten(x)
- delete(x)
  T(x) = NIL

} Worst-case-Laufzeit jeweils $O(1)$

Nachteil: Wenn U groß ist, reservieren wir viel Speicher!

6.2 Hashing

↳ Wir tauschen „worst-case $O(1)$" gegen „average-case $O(1)$" aber reduzieren den Speicherbedarf enorm.

↳ Speicherbedarf $\Theta(|K|)$ nicht mehr $\Theta(|U|)$

Das kann in praktischen Anwendungen entscheidend sein!

Idee: Datensatz mit Schlüssel x wird in

$T[h(x)]$ gespeichert.

Reduktion der (Anzahl der) tatsächlich benötigten Speicherzellen. T heißt Hashtabelle.

$h: U \longrightarrow \{0, 1, \ldots, m-1\}$ heißt Hash-Funktion

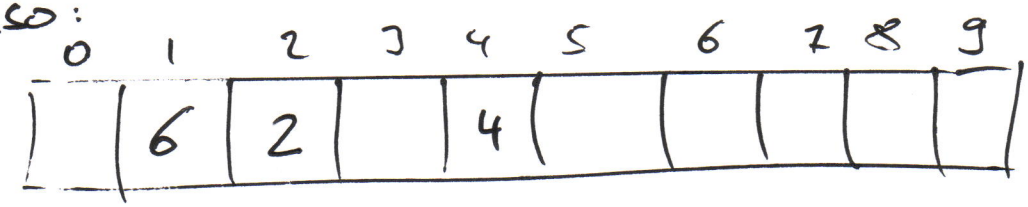
Beispiel

Schlüssel 2, 4, 6 einfügen.

Bereich der Schlüssel $0, \ldots, 9$

$h(x) = x \bmod 5$

Also:



$h(2) = 2 \bmod 5 = 2$

$h(4) = 4 \bmod 5 = 4$

$h(6) = 6 \bmod 5 = 1$

Wir nutzen also nur die Adressen $0, \ldots, 4$.

Noch zu klären

- Kollisionen: Füge z.B. 7 mit $h(7) = 2$ ein, dann Kollision mit der 2
- Suchen? } Wir müssen die Elemente wiederfinden können
- Löschen? }
- Erwartete Laufzeit

Kollisionen

↳ Wie wahrscheinlich ist eine Kollision überhaupt?

Verwandtes Problem: Geburtstagsparadoxon!

Bei wie vielen zufällig ausgewählten Personen, ist es wahrscheinlich, dass zwei von ihnen am selben Tag Geburtstag haben?

Annahme: $P(h(x) = j) = \frac{1}{m}$

Wahrscheinlichkeit dafür, dass Person x an Tag j Geburtstag hat, $m = 365$.

$P(i\text{-tes Datum kollidiert nicht mit den ersten } i-1 \text{ Daten, wenn diese kollisionsfrei sind}) = \frac{m-(i-1)}{m}$

m Möglichkeiten, $i-1$ schon benutzt.

$$P(n \text{ Daten kollisionsfrei}) = \frac{m-1}{m} \cdot \frac{m-2}{m} \cdot \ldots \cdot \frac{m-n+1}{m}$$

(intuitiv: egal welche Tage die ersten $i-1$ Geburtstage belegen, $m-i+1$ der m Möglichkeiten sind gut)

Mit $m = 365$ und $n = 23$:

$P(23 \text{ Daten kollisionsfrei}) = 0{,}49$

Mit $n = 50$

$P(50 \text{ Daten kollisionsfrei}) = 0{,}03$.

Mit $n = 2 \cdot m^{1/2}$:

$$P(2m^{1/2} \text{ Daten kollisionsfrei}) = \frac{m-1}{m} \cdot \ldots \cdot \frac{m-2m^{1/2}+1}{m}$$

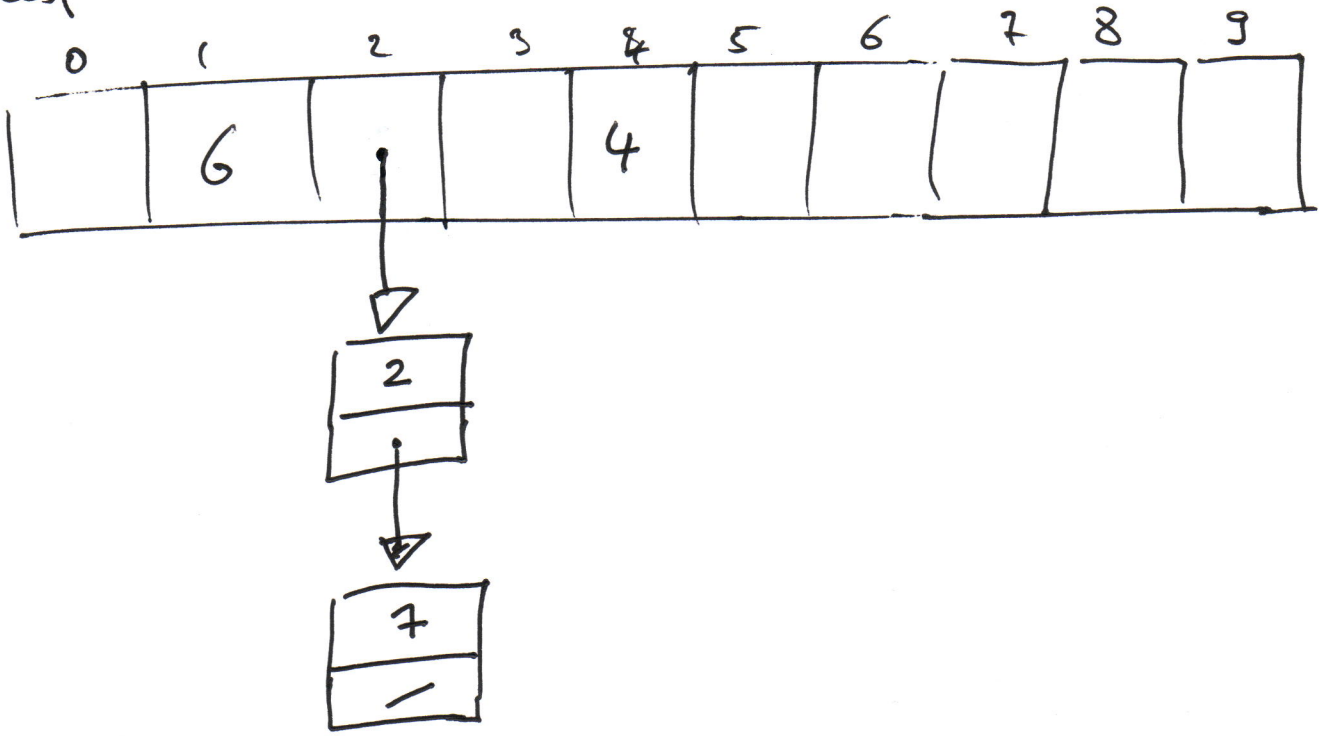
$$\leq \left(\frac{m-m^{1/2}}{m}\right)^{m^{1/2}} = \left(1 - \frac{1}{m^{1/2}}\right)^{m^{1/2}} \approx \frac{1}{e} \approx 0{,}3678\ldots$$

$\Rightarrow$ Kollisionen bei Hashing sind wahrscheinlich!

Kollisionsbehandlung mittels verketteter Listen

Idee: Jede Komponente k in der Hashtabelle enthält Zeiger auf Liste $L(k)$.

Im Beispiele hätten wir also:



Man sieht: Im Worst-Case-Fall kann man eine Liste der Länge n (bei n einzufügenden Elementen) erhalten.

Funktionen

- search(x): Berechne $h(x)$ und suche in Liste $L(h(x))$
- insert(x): Nach erfolgloser Suche, füge x in Liste $L(h(x))$ ein.

• delete(x): Nach erfolgreicher Suche, lösche x aus $L(h(x))$.

Kollisionsbehandlung mittels offener Adressierung

↳ **Idee:** Hashfunktion, die die Nummer des Einfügeversuchs als Parameter hat.

Also: $h : U \times \{0, \ldots, m-1\} \longrightarrow \{0, 1, \ldots, m-1\}$

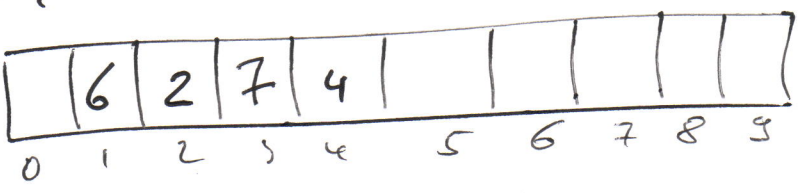
z.B. $t(h(x)) = (h(x) + i) \bmod m$, $i = 0, 1, \ldots$

Beispiel:

$t(i, h(x)) = (x + i) \bmod 5$ d.h. $h(x) = x$

Schlüssel: 2, 4, 6, 7

$t(0,2) = (2+0) \bmod 5 = 2$



$t(0,4) = 4$

$t(0,6) = 1$

$t(0,7) = (7+0) \bmod 5 = 2$ Kollision!

zähle i hoch

$t(1,7) = (7+1) \bmod 5 = 3$ ⟶ 7 nach T[3]

. i solange Hochzählen bis freier Platz gefunden

. i startet bei jedem Elt bei 0.

```
insert (T, x)
    i = 0
    repeat  j = t(x, i)
        if T[j] = NIL
            T[j] = x
            return
        else
            i = i+1
    until i = m
    error „Hash table overflow"
```

```
search(T, x)
    i = 0
    repeat j = t(x, i)
        if T[j] = x
            return j
        i = i + 1
    until T[j] = NIL or i = m
    return NIL
```

Was macht eine gute Hashfunktion aus?

(1) $h : U \to \{0, \ldots, m-1\}$ sollte surjektiv sein, d.h. den gesamten Wertebereich umfassen

(2) Die berechneten Werte sollten sich „gleichmäßig" über $\{0, \ldots, m-1\}$ verteilen

(3) Sie sollte effizient berechenbar sein.