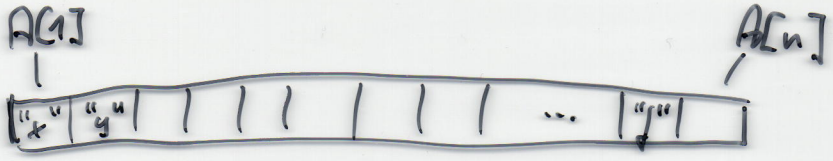
Hashing

Binärbäume brauchen $O(\log n)$ Zeit für

- INSERT (neues Element einfügen)
- DELETE (Element löschen)
- SEARCH (Element finden)

Arrays erlauben $O(1)$ Zugriff auf bekannten Index



Idee: Wenn alle Elemente einen Index „mitbringen", kann man sie effizient im Array verwalten!

Gesucht: Effiziente Datenstruktur auf dieser Basis

Genauer: DS, die im Mittel besser ist, als $O(\log n)$ aber dabei einen schlechteren Worst-Case haben darf.

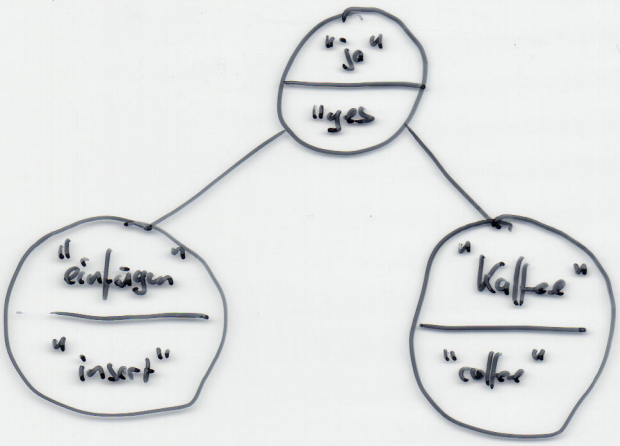
$\boxed{\text{Zu 3}}$ Exkurs: Was ist ein Dictionary?

Bsp.:

```
Dict d;
d.insert("ja", "yes");
d.insert("Kaffee", "coffee");
d.insert("einfügen", "insert");
d.search("ja");     // soll TRUE zurückgeben
d.search("no");     // " FALSE "
d.get("ja");        // " "yes" "
```

Wie setzt man sowas um?

AVL-Bäume



$\boxed{\text{Zu 5}}$ $|U| = m$: einfach

Aber: i. d. R. ist $|U| \gg m$

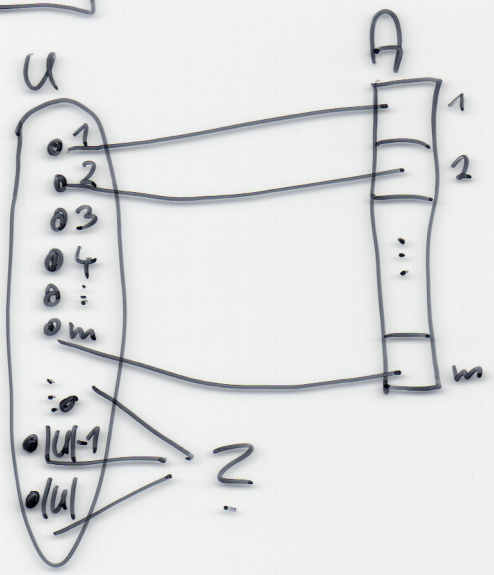
$\boxed{\text{Zu 6}}$ m: Länge des Arrays

n: #Elemente in Hashmap

Belegungsfaktor: $\beta = \frac{n}{m}$ wird i.d.R. in einem festen Intervall gehalten. Bsp. java.util.HashMap ist $0 \leq \beta \leq 0.75$ (per Default). Wird $\beta > 0.75$, wird Rehashing ausgeführt. D.h., alles wird in neue, größere, HashMap kopiert.

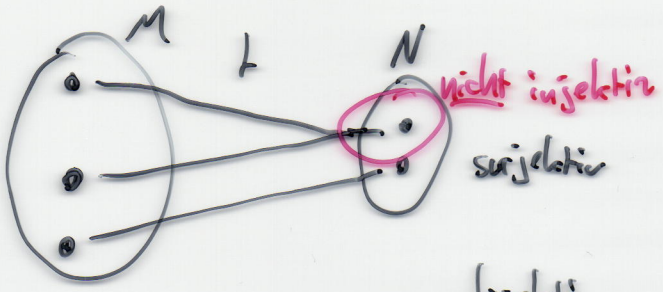
Idee dabei: Teure ($O(n)$) Operationen sind „selten" (nur jedes $\Omega(n)$-te Element) $\rightarrow$ amortisiert konstante Zeit.

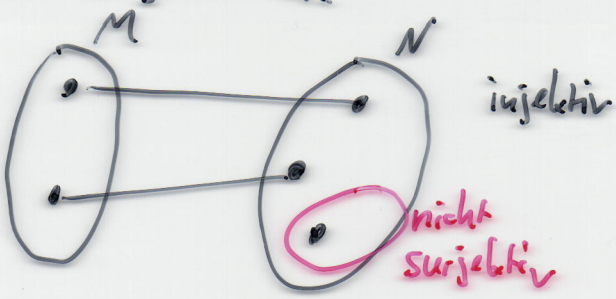
Zu 7

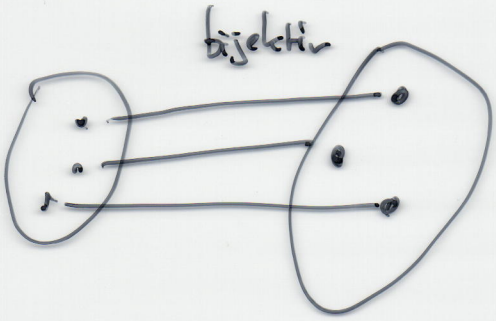


<u>Exkurs</u>: Sei $f: M \to N$ eine Funktion.

- Sei $m_1, m_2 \in M$. f heißt <u>injektiv</u>, wenn: $m_1 \neq m_2 \Rightarrow f(m_1) \neq f(m_2)$ und zwar für <u>alle</u> $m_1, m_2 \in M$
- f heißt <u>surjektiv</u>, wenn $\forall n \in N: \exists m \in M$ mit $f(m) = n$
- f heißt <u>bijektiv</u> wenn f injektiv und surjektiv ist.



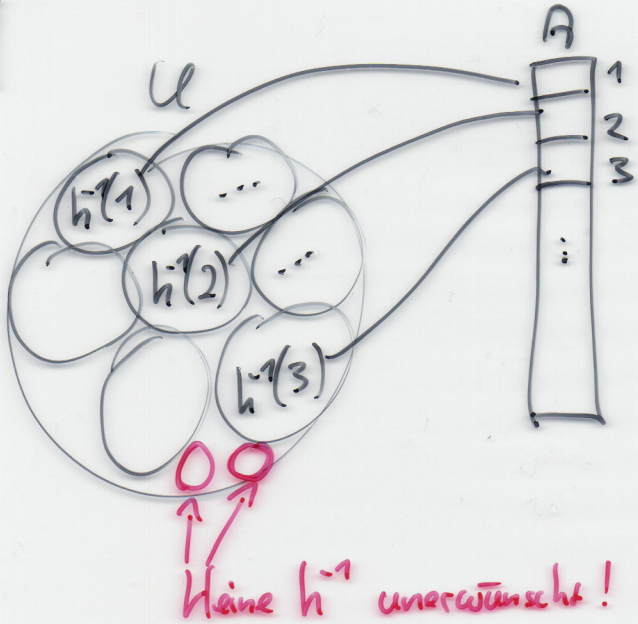




	1	2	3	4	5	6	7	8	9	0
10										
20		X								
30			X							X
40										
50										
60				X	X					
70										
80										
90					X					
00			X							X

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1	X X					X						
2		X										X
3			X									
4												
5										X		
6				X								
7				X								
8							X					
9	X										X	
10												
11							X					
12												
13				X								
14			X							X		
15												
16												
17										X		
18			X	X								
19												
20							X					
21												
22												
23												
24												
25												X
26	X	X				X						
27					X				X		X	
28												
29												
30											X	
31												

Zu 8



Zu 9

$m = 2^k$ ist nicht gut:

$\parallel$

$4 = 2^2$

$h(5) = 1$

$h(9) = 1$

$h(13) = 1$

$\vdots$