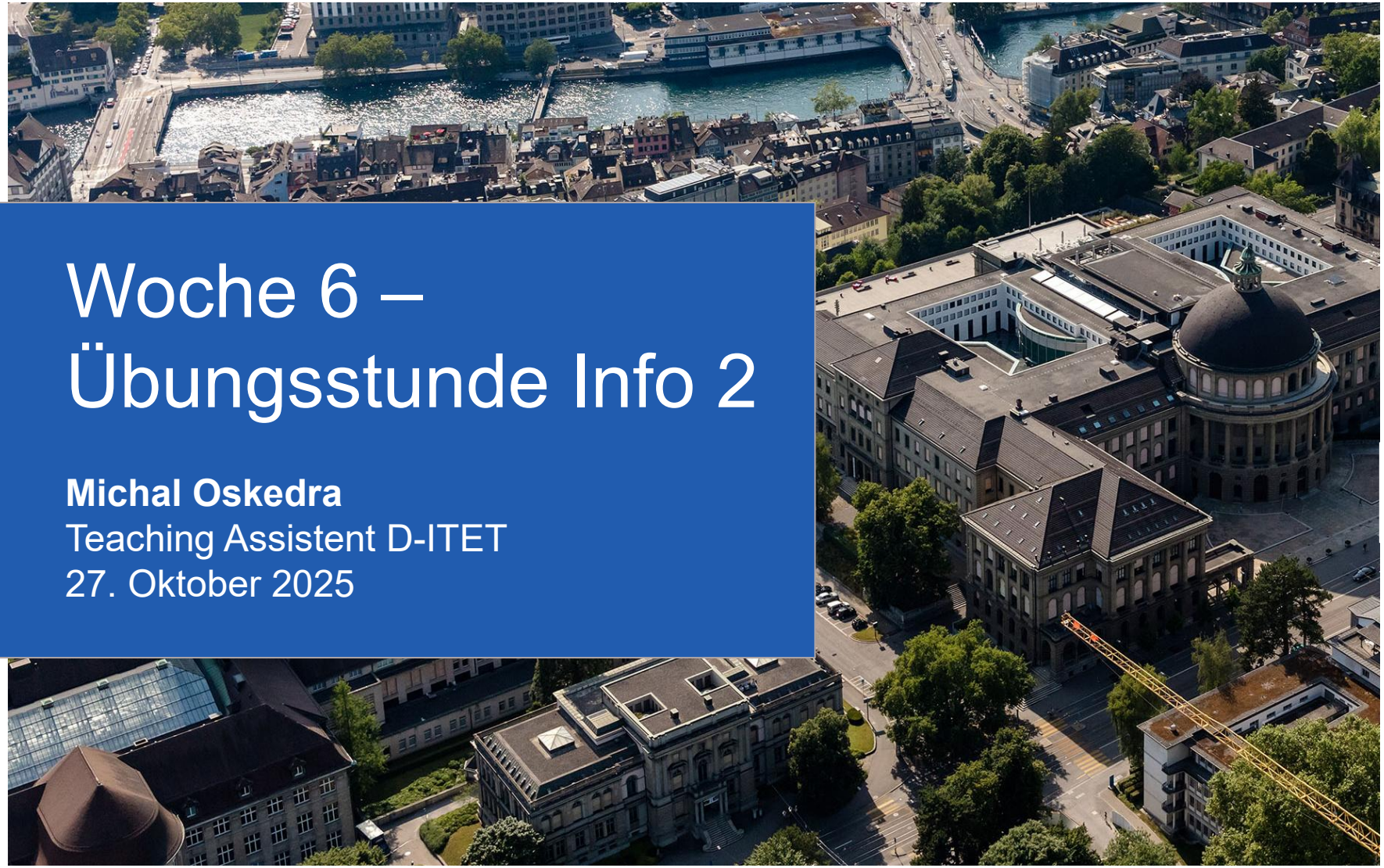
ETH zürich



Woche 6 – Übungsstunde Info 2

Michal Oskedra
Teaching Assistent D-ITET
27. Oktober 2025

Info2 - ML J 34.1 14-16
WhatsApp group

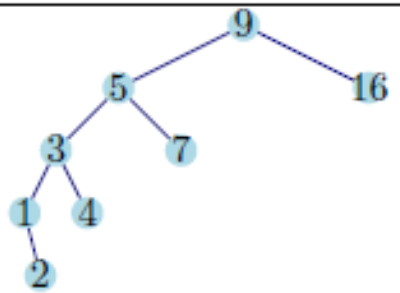
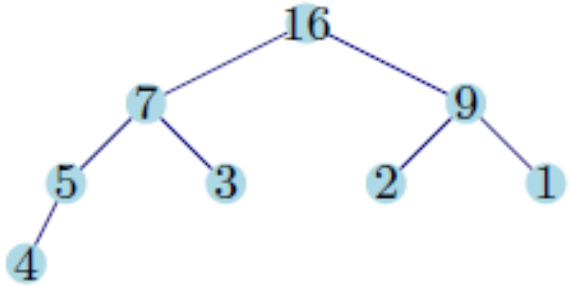
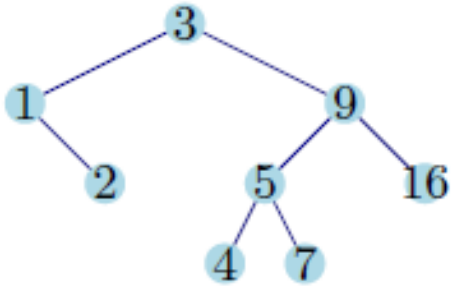
Webseite:
moskedra-web.ethz.ch

01 – Theorie

01.1 AVL Bäume

01.2 Hashing

Wiederholung Baumstrukturen

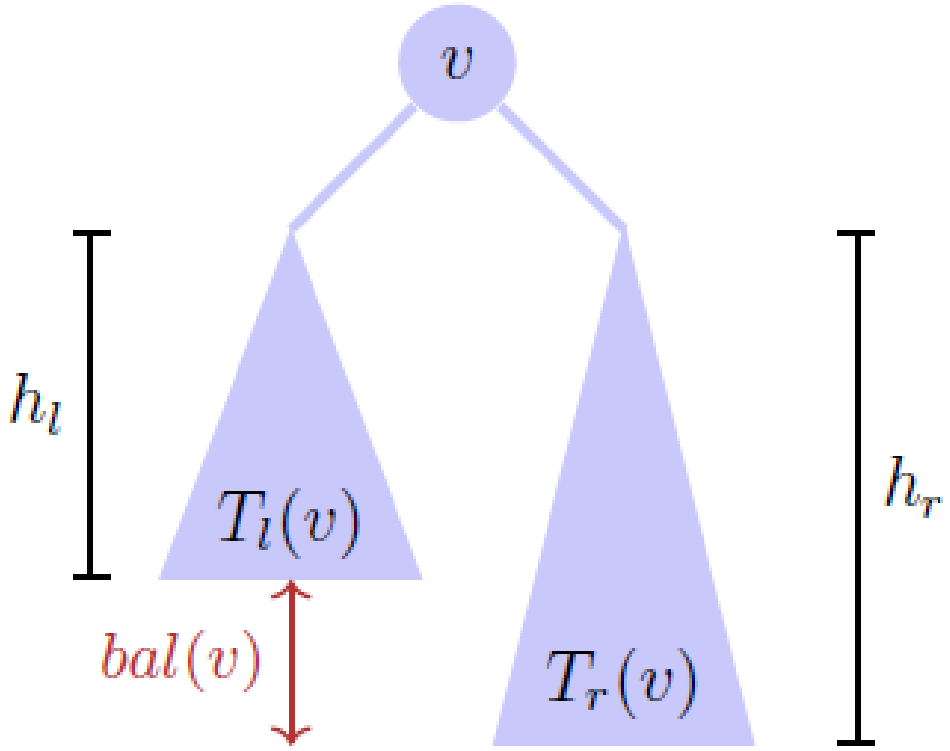
	Suchbäume	**Heaps** Min- / Max- Heap	**Balancierte Bäume** AVL, red-black tree
in C++:		`std::make_heap`	`std::map`
			
Einfügen	$\Theta(h(T))$	$\Theta(\log n)$	$\Theta(\log n)$
Suchen	$\Theta(h(T))$	$\Theta(n)$ (!!)	$\Theta(\log n)$
Löschen	$\Theta(h(T))$	Suchen + $\Theta(\log n)$	$\Theta(\log n)$

Bemerke: $\Theta(\log n) \leq \Theta(h(T)) \leq \Theta(n)$

Balance

Balance eines Knoten ist die **Höhendifferenz** seiner **linken** und **rechten** Teilbäume:

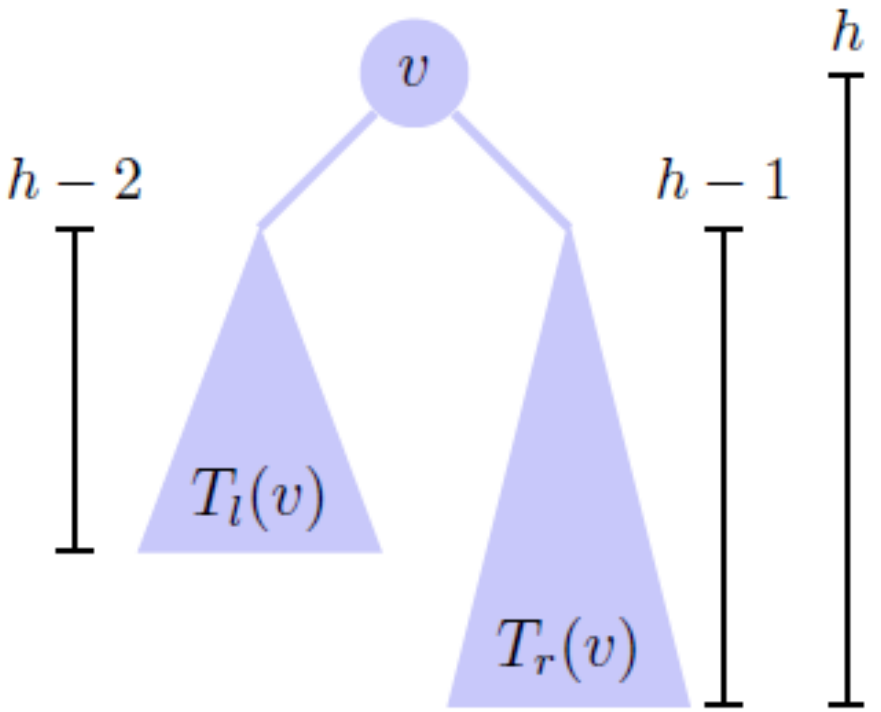
$$\mathrm{bal}(v) := h(T_r(v)) - h(T_l(v))$$



AVL Bedinung

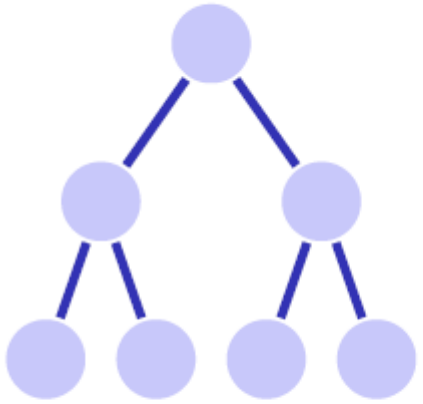
Balance jedes Knoten darf nur **-1, 0, 1** betragen

$$\mathrm{bal}(v) \in \{-1, 0, 1\}$$

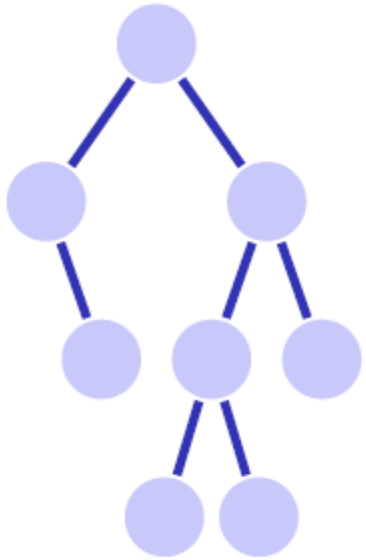


Jedes Teilbaum von einem AVL Baum ist ebenfalls ein AVL Baum!

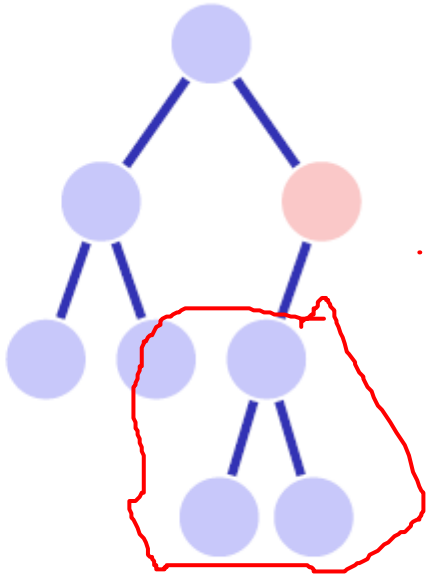
Beispiele



AVL-Baum der Höhe 3

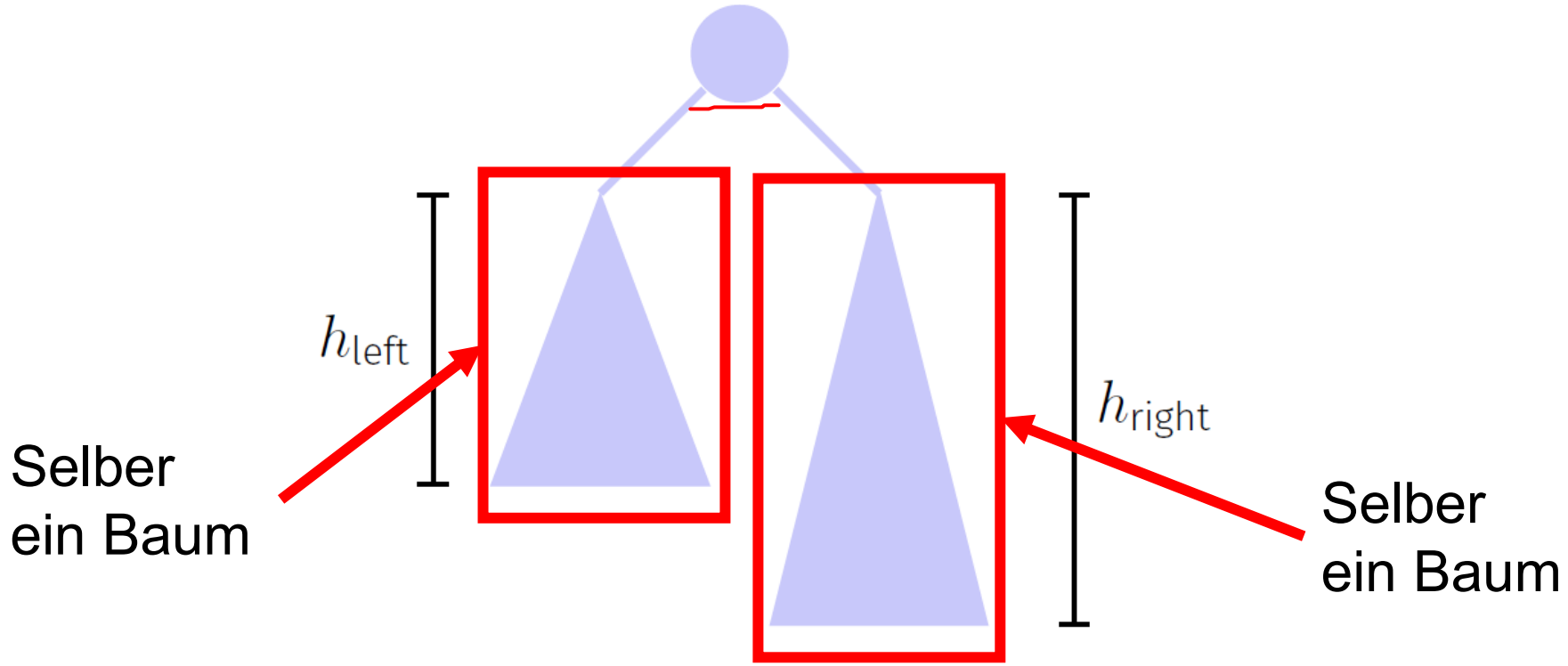


AVL-Baum der Höhe 4



Kein AVL-Baum

Erinnerung: Höhe eines Baumes



$$h(\texttt{r}) = \begin{cases} 0 & \text{falls } \texttt{r} = \textbf{null} \\ 1 + \max\{\underbrace{h(\texttt{r.left})}_{h_{\text{left}}}, \underbrace{h(\texttt{r.right})}_{h_{\text{right}}}\} & \text{sonst.} \end{cases}$$

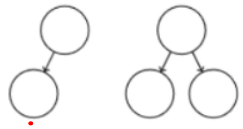
Min/Max-Anzahl Blätter

Beispiele:

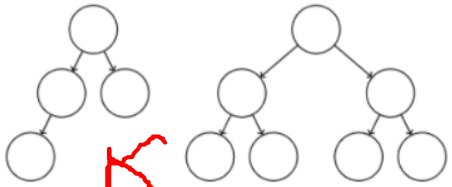
$h = 1 \longrightarrow k_{min}(1) = k_{max}(1) = 1$



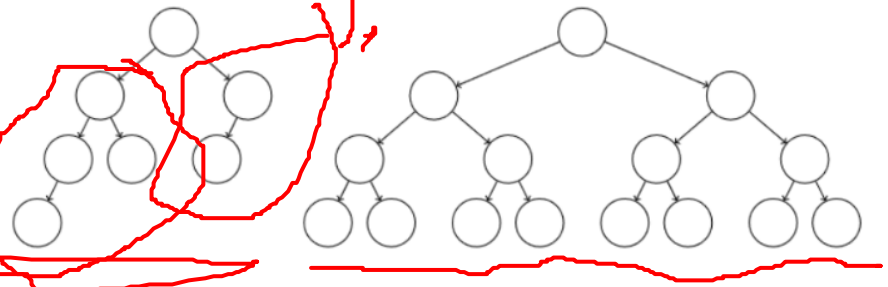
$h = 2 \longrightarrow k_{min}(2) = 1, k_{max}(2) = 2$



$h = 3 \longrightarrow k_{min}(3) = 2, k_{max}(3) = 4$



$h = 4 \longrightarrow k_{min}(4) = 3, k_{max}(4) = 8$



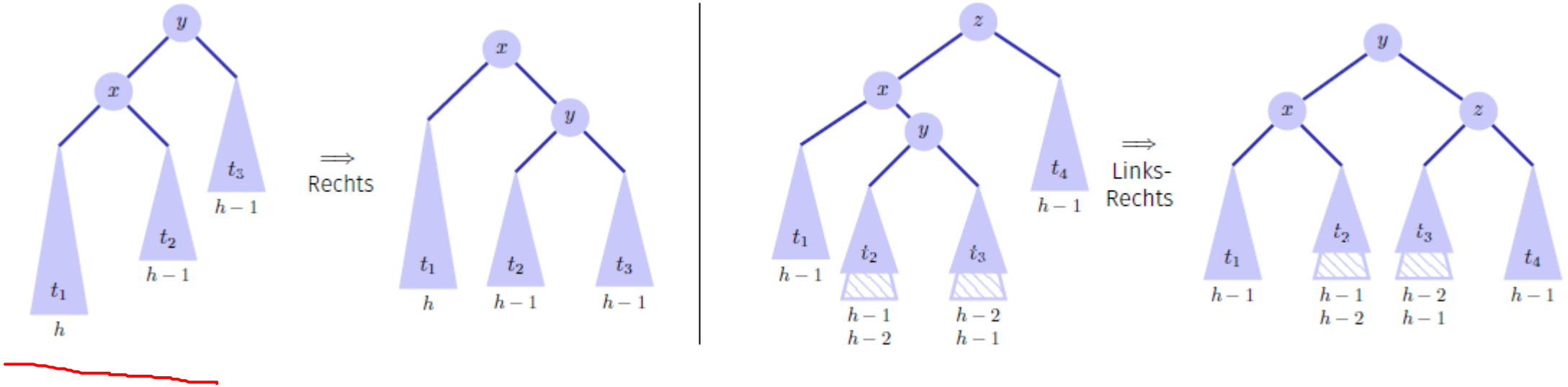
Frage: Allgemeine Berechnung $k_{min}(h)$, $k_{max}(h)$?

$k_n = k_{n-1} + k_{n-2}$

$k = 2^{n-1}$

AVL Rotationen

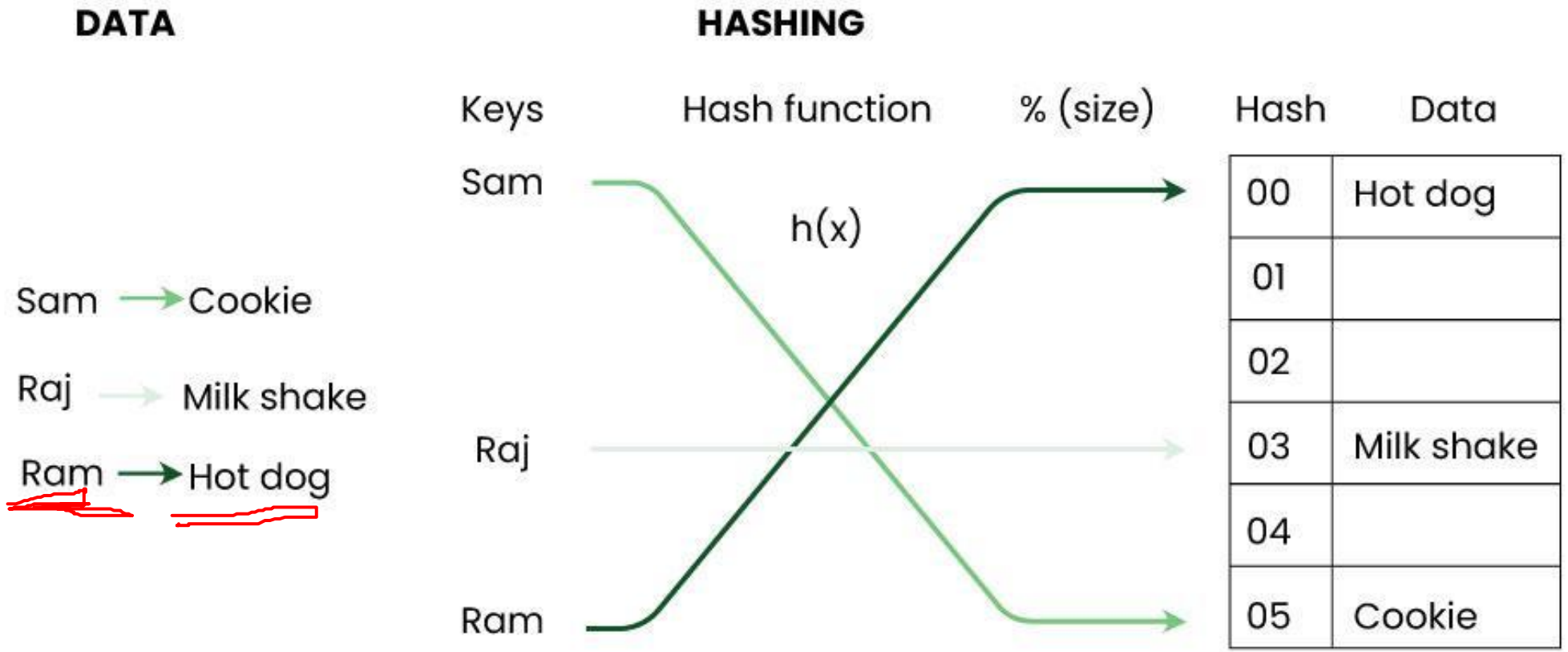
Vier Optionen: L, R, RL, LR



Mehr dazu:
https://www.tutorialspoint.com/data_structures_algorithms/avl_tree_algorithm.htm

01 – Theorie
01.1 AVL Bäume
01.2 Hashing

Hashing Grobidee



Einweg – Funktion!

Prehashing

Jedem Objekt, z.B. Zeichen wird ein Wert zugeteilt

ASCII Table

Dec	Hex	Char	Name	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	null	32	20	Space	64	40	@	96	60	`
1	1	SOH	start of heading	33	21	!	65	41	A	97	61	a
2	2	STX	start of text	34	22	"	66	42	B	98	62	b
3	3	ETX	end of text	35	23	#	67	43	C	99	63	c
4	4	EOT	end of transmission	36	24	$	68	44	D	100	64	d
5	5	ENQ	enquiry	37	25	%	69	45	E	101	65	e
6	6	ACK	acknowledge	38	26	&	70	46	F	102	66	f
7	7	BEL	bell	39	27	'	71	47	G	103	67	g
8	8	BS	backspace	40	28	(	72	48	H	104	68	h
9	9	HT	horizontal tab	41	29	)	73	49	I	105	69	i
10	A	LF	line feed	42	2A	*	74	4A	J	106	6A	j
11	B	VT	vertical tab	43	2B	+	75	4B	K	107	6B	k
12	C	FF	form feed	44	2C	,	76	4C	L	108	6C	l
13	D	CR	carriage return	45	2D	-	77	4D	M	109	6D	m
14	E	SO	shift out	46	2E	.	78	4E	N	110	6E	n
15	F	SI	shift in	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	data link escape	48	30	0	80	50	P	112	70	p
17	11	DC1	device control 1	49	31	1	81	51	Q	113	71	q
18	12	DC2	device control 2	50	32	2	82	52	R	114	72	r
19	13	DC3	device control 3	51	33	3	83	53	S	115	73	s
20	14	DC4	device control 4	52	34	4	84	54	T	116	74	t
21	15	NAK	negative acknowledge	53	35	5	85	55	U	117	75	u
22	16	SYN	synchronous idle	54	36	6	86	56	V	118	76	v
23	17	ETB	end of transmission block	55	37	7	87	57	W	119	77	w
24	18	CAN	cancel	56	38	8	88	58	X	120	78	x
25	19	EM	end of medium	57	39	9	89	59	Y	121	79	y
26	1A	SUB	substitute	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	escape	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	file separator	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	US	unit separator	63	3F	?	95	5F	_	127	7F	DEL

From https://ajsmith.org/tools/ascii-table/

Prehashing bei einem String

$$s = s_1 s_2 \dots s_{l_s}$$

$$ph(s) = \left(\sum_{i=1}^{l_s} s_i \cdot b^i \right) \bmod 2^w$$

Peturbation

Wortlänge

Beispiel mit $b = 31$, $w = 32$, ASCII-Werte s_i

Anna $\mapsto 92966272$
Anne $\mapsto 96660356$
Heinz-Harald $\mapsto 8159299669930423653 3 \bmod 2^{32} = 631641589$

ASCII Table

Dec	Hex	Char	Name	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	null	32	20	Space	64	40	@	96	60	`
1	1	SOH	start of heading	33	21	!	65	41	A	97	61	a
2	2	STX	start of text	34	22	"	66	42	B	98	62	b
3	3	ETX	end of text	35	23	#	67	43	C	99	63	c
4	4	EOT	end of transmission	36	24	$	68	44	D	100	64	d
5	5	ENQ	enquiry	37	25	%	69	45	E	101	65	e
6	6	ACK	acknowledge	38	26	&	70	46	F	102	66	f
7	7	BEL	bell	39	27	'	71	47	G	103	67	g
8	8	BS	backspace	40	28	(	72	48	H	104	68	h
9	9	HT	horizontal tab	41	29	)	73	49	I	105	69	i
10	A	LF	line feed	42	2A	*	74	4A	J	106	6A	j
11	B	VT	vertical tab	43	2B	+	75	4B	K	107	6B	k
12	C	FF	form feed	44	2C	,	76	4C	L	108	6C	l
13	D	CR	carriage return	45	2D	-	77	4D	M	109	6D	m
14	E	SO	shift out	46	2E	.	78	4E	N	110	6E	n
15	F	SI	shift in	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	data link escape	48	30	0	80	50	P	112	70	p
17	11	DC1	device control 1	49	31	1	81	51	Q	113	71	q
18	12	DC2	device control 2	50	32	2	82	52	R	114	72	r
19	13	DC3	device control 3	51	33	3	83	53	S	115	73	s
20	14	DC4	device control 4	52	34	4	84	54	T	116	74	t
21	15	NAK	negative acknowledge	53	35	5	85	55	U	117	75	u
22	16	SYN	synchronous idle	54	36	6	86	56	V	118	76	v
23	17	ETB	end of transmission block	55	37	7	87	57	W	119	77	w
24	18	CAN	cancel	56	38	8	88	58	X	120	78	x
25	19	EM	end of medium	57	39	9	89	59	Y	121	79	y
26	1A	SUB	substitute	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	escape	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	file separator	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	US	unit separator	63	3F	?	95	5F	_	127	7F	DEL

From https://ajsmith.org/tools/ascii-table/

Also genügt Prehashing?

Ja...
+ Immer anwendbar
+ Schnelles Nachschlagen, Einfügen, Löschen

Aber...
- Platzineffizient!!! → Deswegen wird Hashing angewendet

Definition

Hashing:
Reduktion der Hashtabellengrösse

Voraussetzungen
- Eindeutig
- Ermöglicht schnelles Nachschlagen, Einfügen und Löschen

Definition

Erinnerung Modulo

$$a \bmod n = r \quad \text{where } a = qn + r \text{ and } 0 \leq r < n$$

Where:

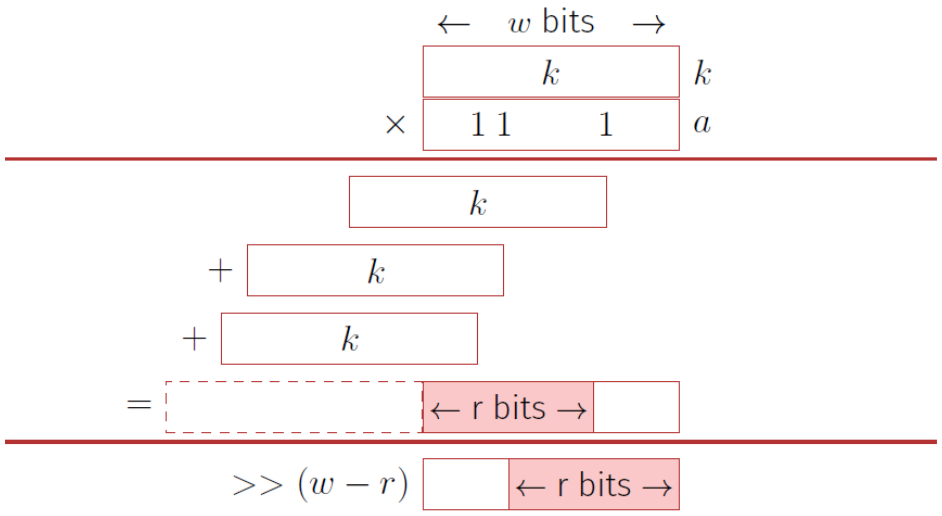
a = The number being divided (the dividend)

n = The number you divide by (the modulus), must be positive

q = The quotient (how many whole times n fits into a)

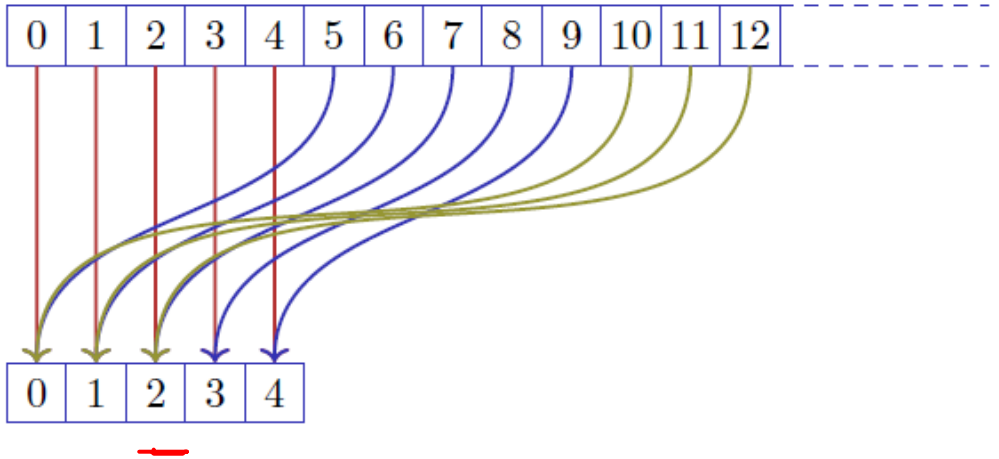
r = The remainder, which is the result of the modulo operation

Multiplikationsmethode



$$h(k) = \left\lfloor (a \cdot k \bmod 2^w)/2^{w-r} \right\rfloor \bmod m$$

Divisionsmethode



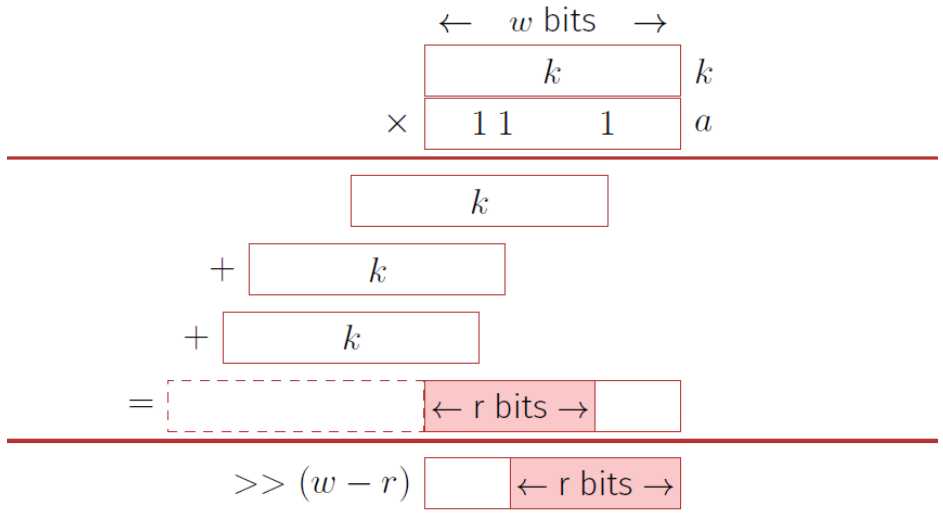
$$h(k) = k \bmod m$$

m – eine Primzahl

Definition

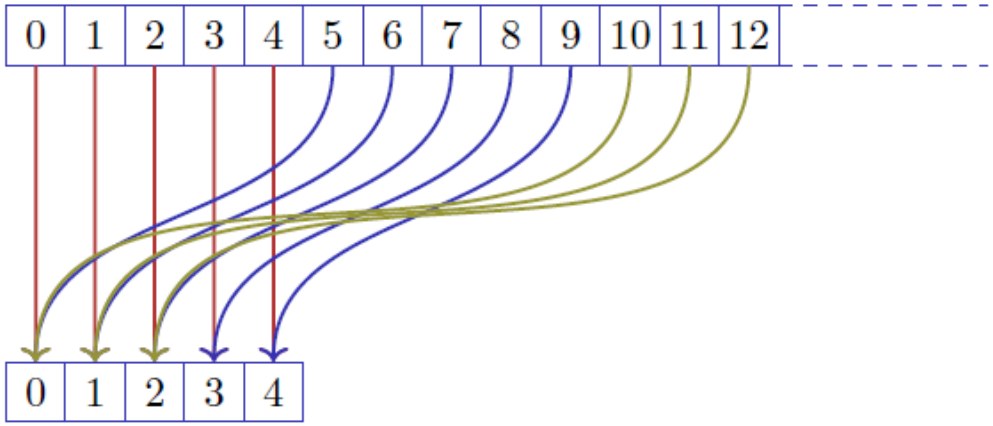
Hier im Kurs

Multiplikationsmethode



$$h(k) = \left\lfloor (a \cdot k \bmod 2^w)/2^{w-r} \right\rfloor \bmod m$$

Divisionsmethode



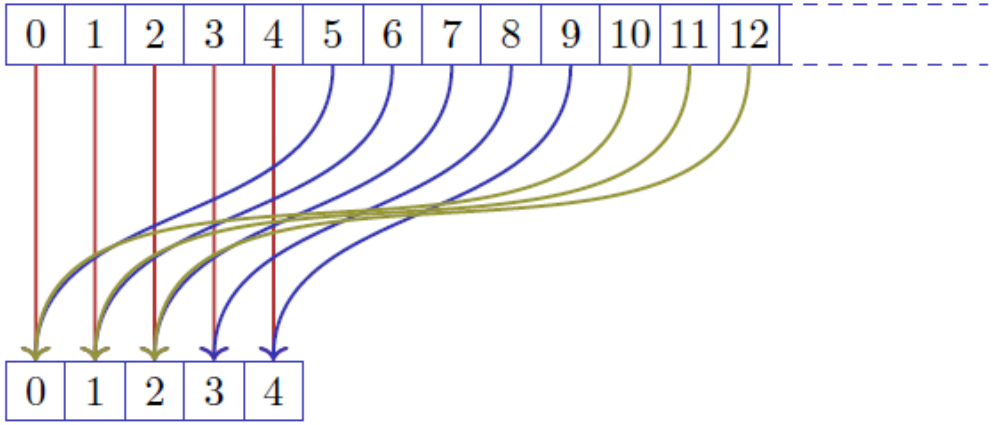
$$h(k) = k \bmod m$$

Wieso wird für m eine Primzahl gewählt?

Divisionsmethode

Schlüssel (k)	(h(k)) bei m=10	(h(k)) bei m=11
4	4	4
8	8	8
12	2	1
16	6	5
20	0	9
24	4	2
28	8	6
32	2	10
36	6	3
40	0	7

Divisionsmethode



$$h(k) = k \bmod m$$

Wieso wird für m eine Primzahl gewählt?

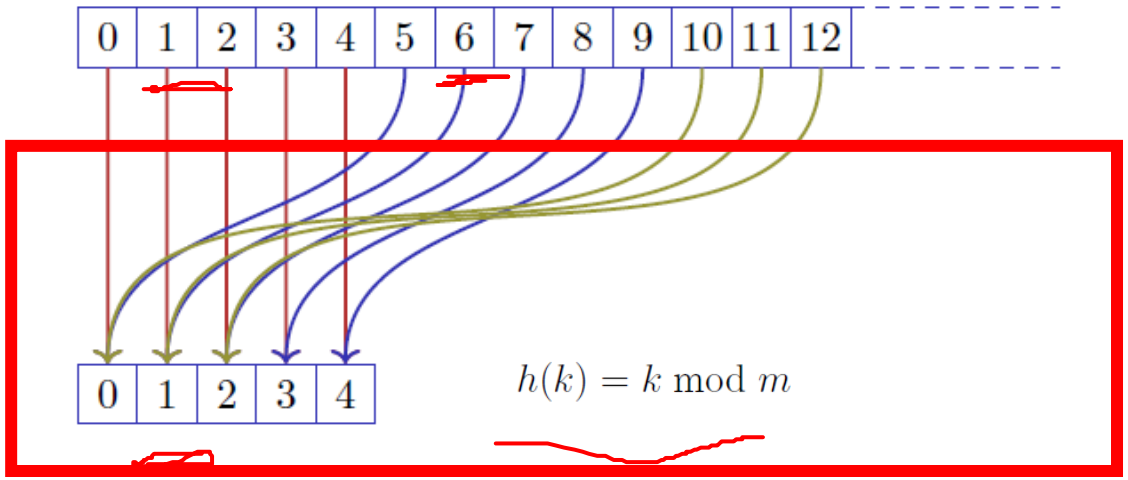
Kollisionen

Kommen vor wenn mehrere Schlüssel zu einem Tabelleneintrag zugeordnet werden.

Lösungen:

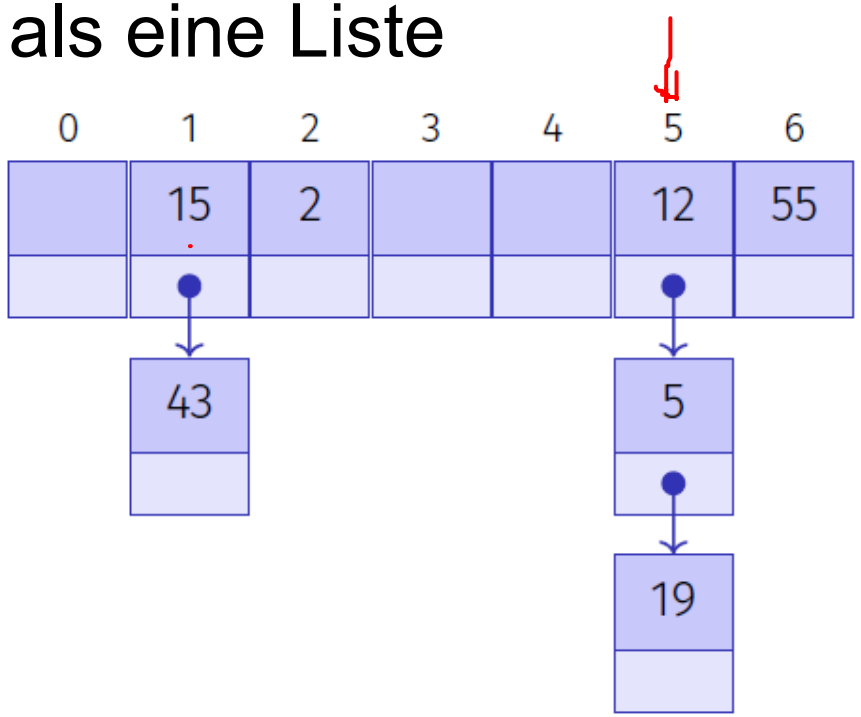
a) Offenes Hashing

b) Geschlossenes Hashing



Offenes Hashing:

Mehrere Einträge pro Tabelleneintrag möglich → miteinenander verkettet als eine Liste



$$m = 7, h(k) = k \bmod m.$$

Schlüssel 12 , 55 , 5 , 15 , 2 , 19 , 43

Offenes Hashing:

Vorteile:
+ Schlüssel löschen sehr einfach

Nachteile:
- Laufzeit O(n) bei degenerierten Hashtabellen mit lange Kollisionsketten
- Häufige Speicher(de)allokationen

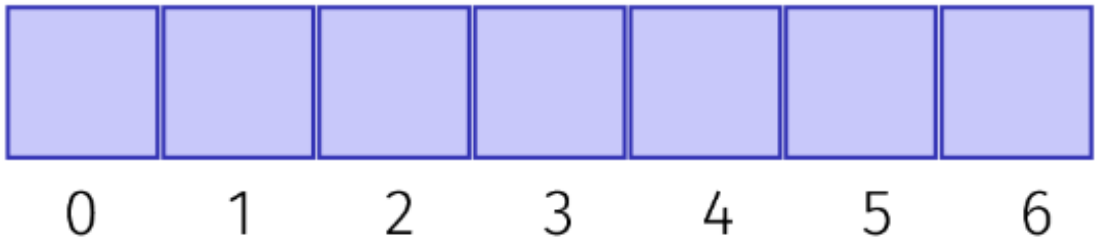
Geschlossenes Hashing: Lineares Sondieren

Ein Eintrag pro Tabelleneintrag. Bei einer Kollision gehen wir einfach zur nächsten Stelle: $s(k, j) = h(k) + j$

Beispiel:

$$25, 4, 17, 45$$

$$h(k) = k \bmod 7$$



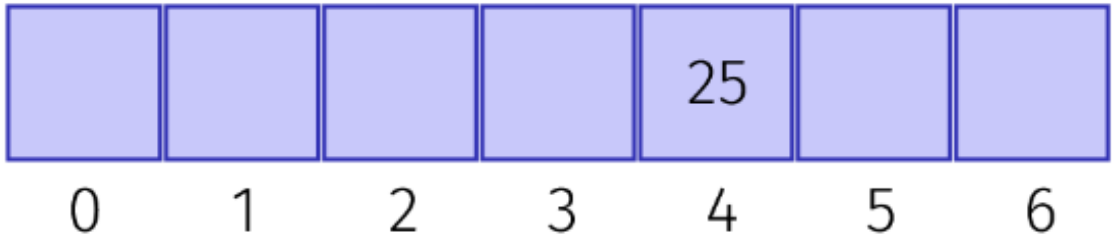
Geschlossenes Hashing: Lineares Sondieren

Ein Eintrag pro Tabelleneintrag. Bei einer Kollision gehen wir einfach zur nächsten Stelle: $s(k, j) = h(k) + j$

Beispiel:

$$25, 4, 17, 45$$

$$h(k) = k \bmod 7$$



Geschlossenes Hashing: Lineares Sondieren

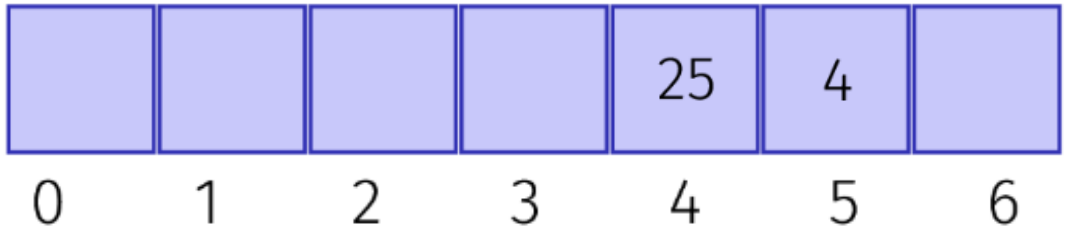
Ein Eintrag pro Tabelleneintrag. Bei einer Kollision gehen wir einfach zur nächsten Stelle:

$$s(k, j) = h(k) + j$$

Beispiel:

$$25, 4, 17, 45$$

$$h(k) = k \bmod 7$$



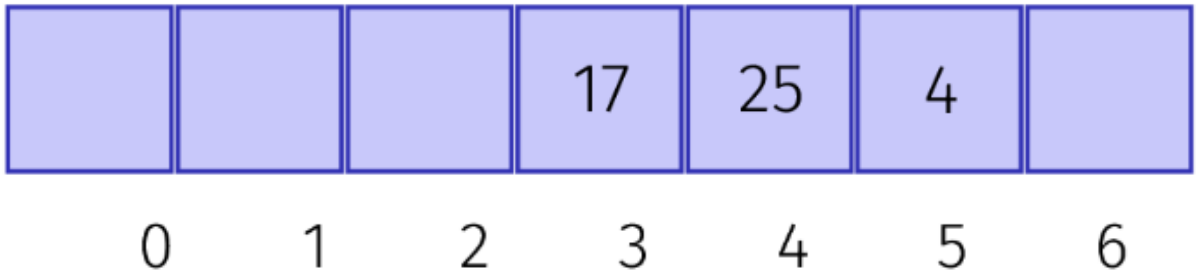
Geschlossenes Hashing: Lineares Sondieren

Ein Eintrag pro Tabelleneintrag. Bei einer Kollision gehen wir einfach zur nächsten Stelle: $s(k, j) = h(k) + j$

Beispiel:

$$25, 4, 17, 45$$

$$h(k) = k \bmod 7$$



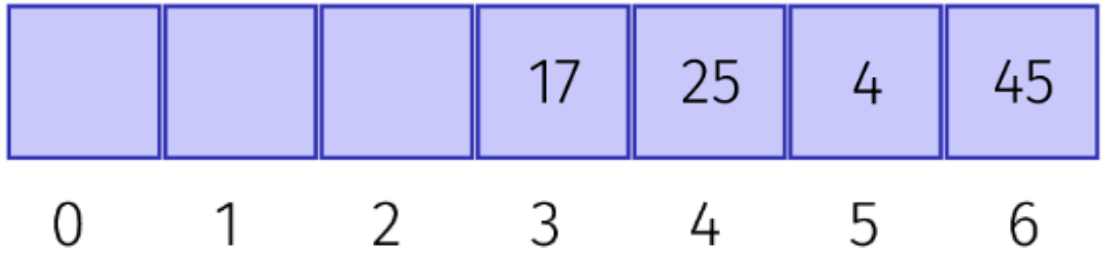
Geschlossenes Hashing: Lineares Sondieren

Ein Eintrag pro Tabelleneintrag. Bei einer Kollision gehen wir einfach zur nächsten Stelle: $s(k, j) = h(k) + j$

Beispiel:

$$25, 4, 17, 45$$

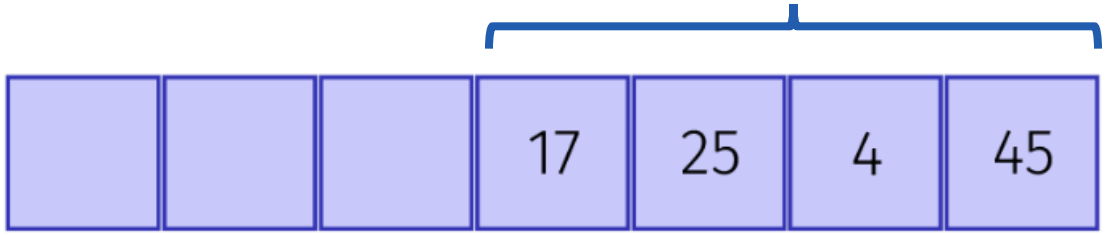
$$h(k) = k \bmod 7$$



Geschlossenes Hashing: Lineares Sondieren

Vorteile:
+ Wir verwenden ausschliesslich die Hashtabelle
+ Laufzeit O(1) falls keine Häufung antrifft
- Primäre Häufung

primäre Häufung: lange, durchgehend belegte Bereiche bei ähnlichen Schlüssel



Geschlossenes Hashing: Double Hashing:

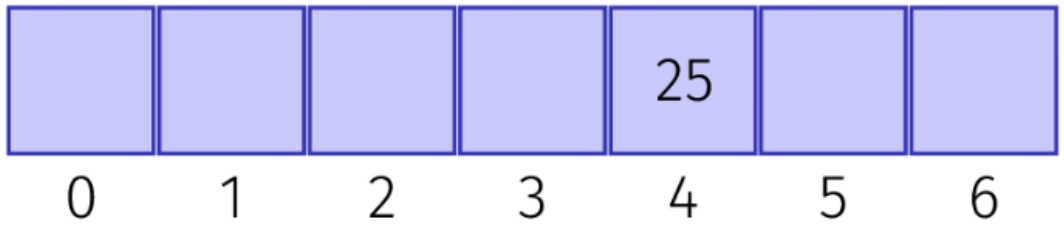
Ein Eintrag pro Tabelleneintrag. Bei einer Kollision addiert man den Betrag einer zweiten Hashfunktion `h'()` multipliziert mit dem Zähler j

$$s(k, j) = h(k) + j \cdot h'(k).$$

Beispiel:

$$25, 4, 17, 45$$

$$h(k) = k \bmod 7 + j \cdot (1 + (k \bmod 5)).$$



Geschlossenes Hashing: Double Hashing:

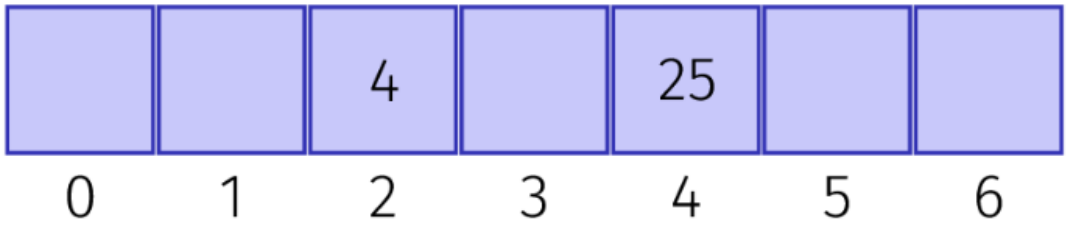
Ein Eintrag pro Tabelleneintrag. Bei einer Kollision addiert man den Betrag einer zweiten Hashfunktion `h'()` multipliziert mit dem Zähler j

$$s(k, j) = h(k) + j \cdot h'(k).$$

Beispiel:

$$25, 4, 17, 45$$

$$h(k) = k \bmod 7 + j \cdot (1 + (k \bmod 5)).$$



Geschlossenes Hashing: Double Hashing:

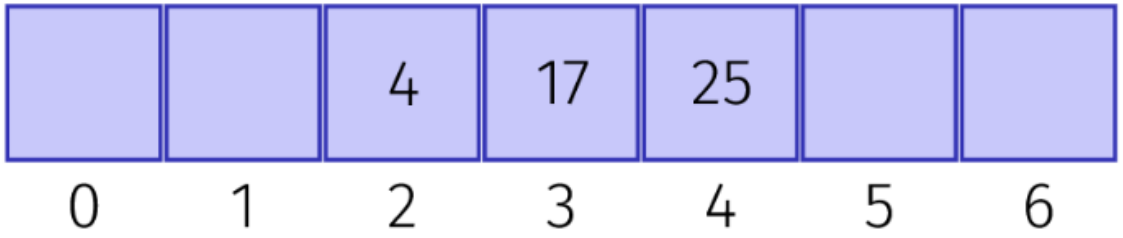
Ein Eintrag pro Tabelleneintrag. Bei einer Kollision addiert man den Betrag einer zweiten Hashfunktion `h'()` multipliziert mit dem Zähler j

$$s(k, j) = h(k) + j \cdot h'(k).$$

Beispiel:

$$25, 4, 17, 45$$

$$h(k) = k \bmod 7 + j \cdot (1 + (k \bmod 5)).$$



Geschlossenes Hashing: Double Hashing:

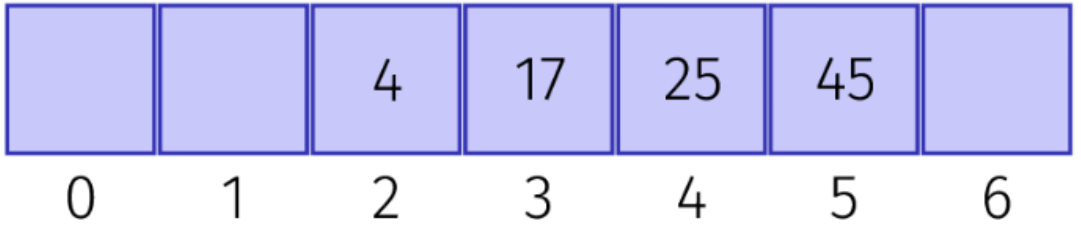
Ein Eintrag pro Tabelleneintrag. Bei einer Kollision addiert man den Betrag einer zweiten Hashfunktion `h'()` multipliziert mit dem Zähler j

$$s(k, j) = h(k) + j \cdot h'(k).$$

Beispiel:

$$25, 4, 17, 45$$

$$h(k) = k \bmod 7 + j \cdot (1 + (k \bmod 5)).$$



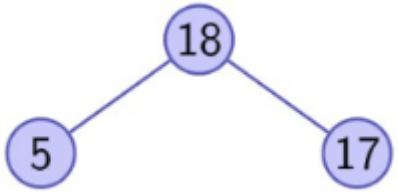
Geschlossenes Hashing: Double Hashing

Vorteile:
+ Wir verwenden ausschliesslich die Hashtabelle
+ Häufung wird vermieden, wenn beide `h()` und `h'()` unabhängig (z.B. benutzen verschiedene Primzahlen)

02 – Quiz

Quiz Frage 1

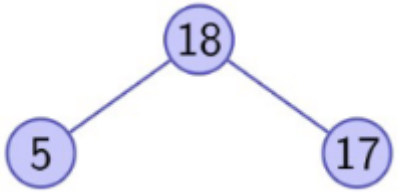
Welche Struktur ist das?



a) BST
b) AVL
c) Heap
d) Keine davon

Quiz Frage 1

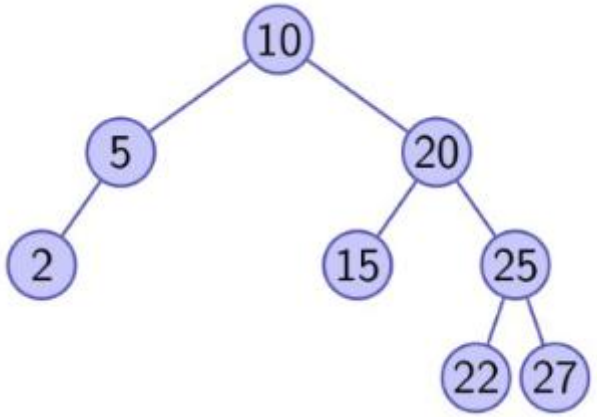
Welche Struktur ist das?



a) BST
b) AVL
c) Heap
d) Keine davon

Quiz Frage 2

Welche Struktur ist das **nicht**?



a) BST
b) AVL
c) Heap
d) Keine Davon

Quiz Frage 3

Wie viele Blätter (hier: Knoten ohne Kinder) *k* hat ein AVL-Baum der Höhe *h* = 7 **höchstens**?

a) 12
b) 32
c) 64
d) 72

Quiz Frage 3

Wie viele Blätter (hier: Knoten ohne Kinder) *k* hat ein AVL-Baum der Höhe *h* = 7 **höchstens**?

a) 12
b) 32
c) 64
d) 72

Quiz Frage 4

Wie viele Blätter (hier: Knoten ohne Kinder) *k* hat ein AVL-Baum der Höhe *h* = 7 mindestens?

a) 5
b) 10
c) 12
d) 13

Quiz Frage 4

Wie viele Blätter (hier: Knoten ohne Kinder) *k* hat ein AVL-Baum der Höhe *h* = 7 mindestens?

a) 5
b) 10
c) 12
d) 13

Quiz Frage 5

Wie viele **Knoten** *k* hat ein AVL-Baum der Höhe *h* = 7 **mindestens**?

a) 13
b) 33
c) 48
d) 64

Quiz Frage 5

Wie viele **Knoten** *k* hat ein AVL-Baum der Höhe *h* = 7 **mindestens**?

a) 13
b) 33
c) 48
d) 64

Quiz Frage 6

Welche Art von Zahlen eignen sich besonders gut für ein Modulo von der Hash-Funktion?

a) Irrationale Zahlen
b) Quadratische Zahlen
c) Kubische Zahlen
d) Primzahlen

Quiz Frage 6

Welche Art von Zahlen eignen sich besonders gut für ein Modulo von der Hash-Funktion?

a) Irrationale Zahlen
b) Quadratische Zahlen
c) Kubische Zahlen
d) Primzahlen

Quiz Frage 7

Diese Hashtabelle wurde mit einer Hash-Funktion
h(k) = k mod 10 mit linearen Sondierung eingefügt.

0	1	2	3	4	5	6	7	8	9
		32	52	33	74	96			

In welcher Reihenfolge konnten die Schlüssel eingefügt werden?

a) [32, 33, 52, 96, 74]
b) [32, 52, 33, 74, 96]
c) [32, 52, 74, 96, 33]
d) [96, 32, 52, 33, 74]

Quiz Frage 7

Diese Hashtabelle wurde mit einer Hash-Funktion
h(k) = k mod 10 mit linearen Sondierung eingefügt.

0	1	2	3	4	5	6	7	8	9
		32	52	33	74	96			

In welcher Reihenfolge konnten die Schlüssel eingefügt werden?

a) [32, 33, 52, 96, 74]
b) [52, 32, 33, 74, 96]
c) [32, 52, 74, 96, 33]
d) [96, 32, 52, 33, 74]

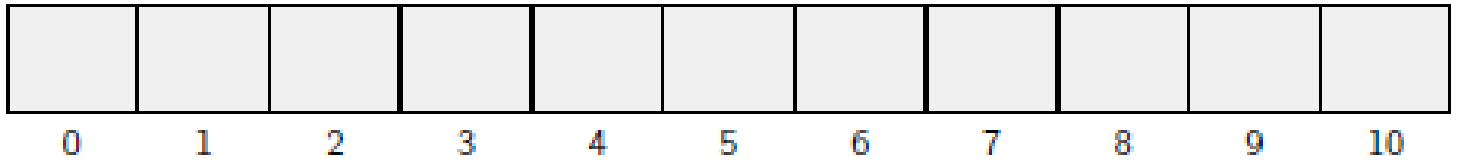
Quiz Frage 8

Füge die Schlüssel in die folgende Hashtabelle mit Double-Hashing

`h(k) = k mod 11`

`h'(k) = 1 + (k mod 9)` zur Sondierung

Schlüssel / *Keys*: 33, 21, 10, 11, 1, 22



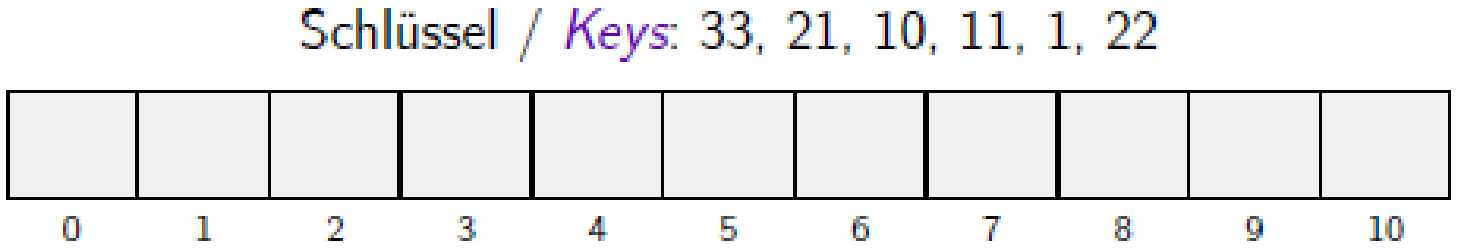
a) [33, 10, -, 11, 22, 1, -, -, -, -, 21]
b) [33, 10, -, 11, -, 1, 22, -, -, -, 21]
c) [33, 10, -, 11, 22, 21, -, -, -, -, -]
d) [-, 10, 33, 11, 22, 1, -, -, -, -, 21]

Quiz Frage 8

Füge die Schlüssel in die folgende Hashtabelle mit Double-Hashing

`h(k) = k mod 11`

`h'(k) = 1 + (k mod 9)` zur Sondierung



a) [33, 10, -, 11, 22, 1, -, -, -, -, 21]

b) [33, 10, -, 11, -, 1, 22, -, -, -, 21]

c) [33, 10, -, 11, 22, 21, -, -, -, -, -]

d) [-, 10, 33, 11, 22, 1, -, -, -, -, 21]

Quiz Frage 9

Füge die Schlüssel in die folgende Hashtabelle mit Double-Hashing

`h(k) = k mod 7`

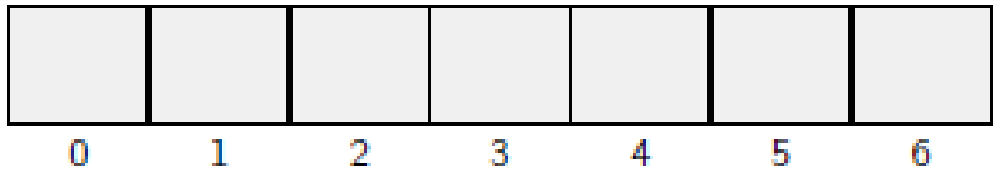
`h'(k) = 1 + (k mod 5)` zur Sondierung

a) [0, 7, 14, 5, 10, 15, 20]
b) [0, 15, 20, 7, 10, 14, 5]
c) [0, 7, 14, 15, 10, 5, 20]
d) [0, 15, 20, 10, 7, 14, 5]

Schlüssel / *Keys*: 0 7 14 5 10 15 20

Werte von h und h': *Values of h and h':*

k	0	7	14	5	10	15	20
$h(k)$	0	0	0	5	3	1	6
$h'(k)$	1	3	5	1	1	1	1



Quiz Frage 9

Füge die Schlüssel in die folgende Hashtabelle mit Double-Hashing

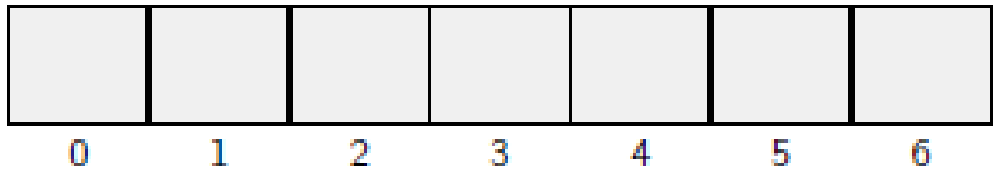
`h(k) = k mod 7`

`h'(k) = 1 + (k mod 5)` zur Sondierung

Schlüssel / *Keys*: 0 7 14 5 10 15 20

Werte von h und h': *Values of h and h':*

k	0	7	14	5	10	15	20
$h(k)$	0	0	0	5	3	1	6
$h'(k)$	1	3	5	1	1	1	1



a) [0, 7, 14, 5, 10, 15, 20]

b) [0, 15, 20, 7, 10, 14, 5]

c) [0, 7, 14, 15, 10, 5, 20]

d) [0, 15, 20, 10, 7, 14, 5]

03 – Programmierübung

Programmierübung: Sliding Window

Sliding Window - Student attempt

```cpp
#include <vector>

struct Result {
  int start_index;
  int sum;
};

Result max_window_sum_naive(const std::vector<int>& v, const unsigned int window_size) {
  Result result = {0, -1};

  if (window_size > v.size()) {
    return result;
  }

  int max_sum = INT_MIN;
  int start_idx = -1;

  for (unsigned int i = 0; i < v.size() - window_size + 1; ++i) {
    const int current_sum = std::accumulate(&v[i], &v[i + window_size], 0);

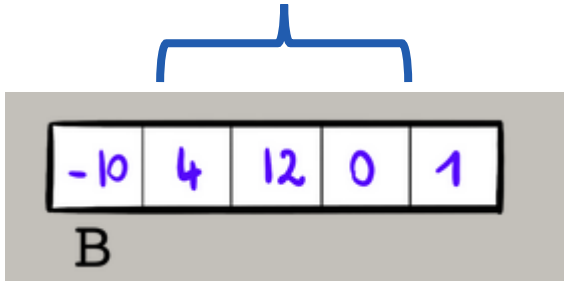
    if (current_sum > max_sum) {
      max_sum = current_sum;
      start_idx = i;
    }
  }

  result.start_index = start_idx;
  result.sum = max_sum;

  return result;
}
```

1. Geh auf CodeExpert → Code Examples → Sliding Window
2. Implementiere `max_window_sum`, das die Summe einer Zahlenfolge von Länge n in einem Array findet, all das in O(n)

Beispiel:
n = 3, Sum = 4 + 12 + 0 = 16



ETH zürich

Michal Oskedra
Teaching Assistent D-ITET
moskedra@ethz.ch