Informatik II

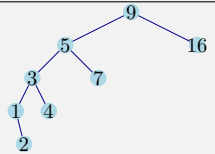
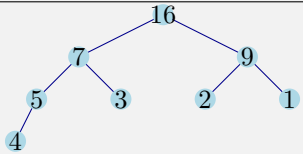
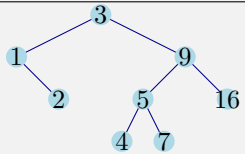
Übung 6

FS 2020

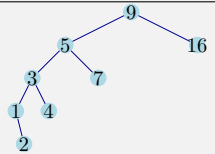
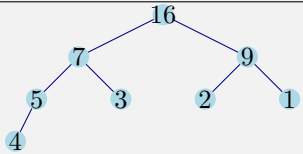
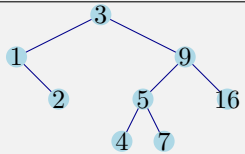
Heutiges Programm

1. Rekapitulation binäre Bäume
2. Wiederholung Vorlesung
   - AVL Bedingung
   - AVL Einfügen
3. In-Class-Exercises

Vergleich binärer Bäume

	Suchbäume	**Heaps** Min- / Max- Heap	**Balancierte Bäume** AVL, red-black tree
in Java:		PriorityQueue	TreeSet
			
Einfügen	$\mathcal{O}(h(T))$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
Suchen	$\mathcal{O}(h(T))$	$\mathcal{O}(n)$ (!!)	$\mathcal{O}(\log n)$
Löschen	$\mathcal{O}(h(T))$	Suchen + $\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$

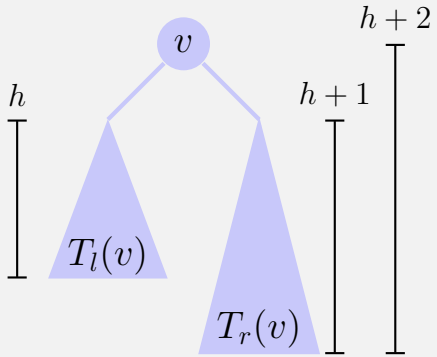
Vergleich binärer Bäume

	Suchbäume	**Heaps** Min- / Max- Heap	**Balancierte Bäume** AVL, red-black tree
in Java:		PriorityQueue	TreeSet
			
Einfügen	$\mathcal{O}(h(T))$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
Suchen	$\mathcal{O}(h(T))$	$\mathcal{O}(n)$ (!!)	$\mathcal{O}(\log n)$
Löschen	$\mathcal{O}(h(T))$	Suchen + $\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$

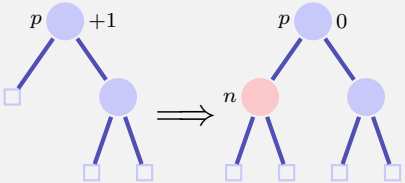
Bemerke: $\mathcal{O}(\log n) \leq \mathcal{O}(h(T)) \leq \mathcal{O}(n)$

AVL Bedingung

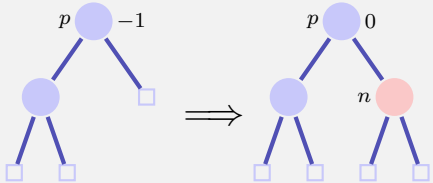
AVL Bedingung: für jeden Knoten v eines Baumes gilt $\mathrm{bal}(v) \in \{-1, 0, 1\}$



Balance am Einfügeort



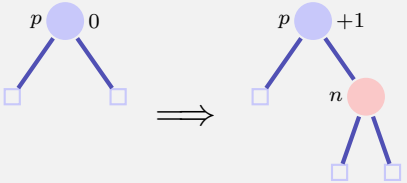
Fall 1: $\mathrm{bal}(p) = +1$



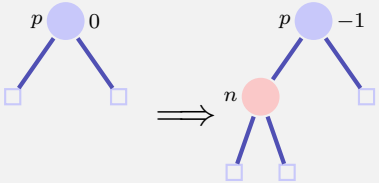
Fall 2: $\mathrm{bal}(p) = -1$

Fertig in beiden Fällen, denn der Teilbaum ist nicht gewachsen.

Balance am Einfügeort



Fall 3.1: $\mathrm{bal}(p) = 0$ rechts



Fall 3.2: $\mathrm{bal}(p) = 0$, links

In beiden Fällen noch nicht fertig. Aufruf von `upin(p)`.

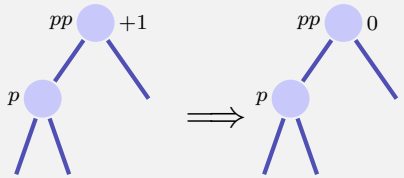
upin(p) - Invariante

Beim Aufruf von `upin(p)` gilt, dass

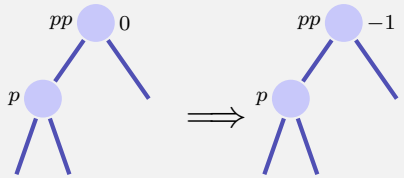
- der Teilbaum ab p gewachsen ist und
- $\mathrm{bal}(p) \in \{-1, +1\}$

upin(p)

Annahme: p ist linker Sohn von pp[1]



Fall 1: $\mathrm{bal}(pp) = +1$, fertig.



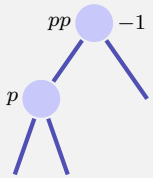
Fall 2: $\mathrm{bal}(pp) = 0$, `upin(pp)`

In beiden Fällen gilt nach der Operation die AVL-Bedingung für den Teilbaum ab pp

[1] Ist p rechter Sohn: symmetrische Fälle unter Vertauschung von $+1$ und -1

upin(p)

Annahme: p ist linker Sohn von pp



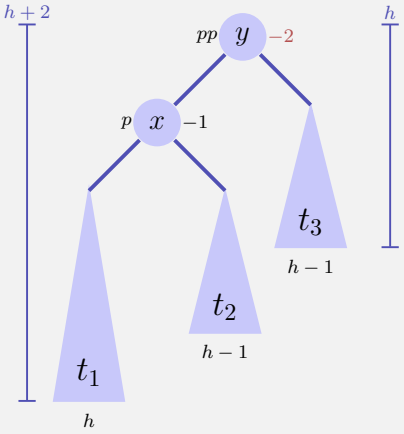
Fall 3: $\mathrm{bal}(pp) = -1$,

Dieser Fall ist problematisch: das Hinzufügen von n im Teilbaum ab pp hat die AVL-Bedingung verletzt. Rebalancieren!

Zwei Fälle $\mathrm{bal}(p) = -1$, $\mathrm{bal}(p) = +1$

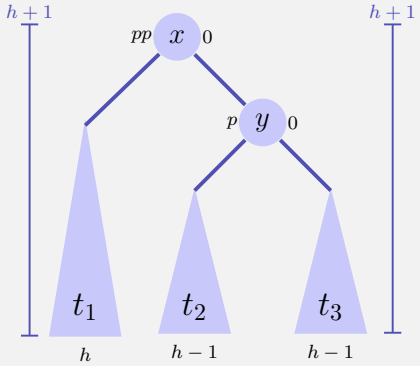
Rotationen

Fall 1.1 $\mathrm{bal}(p) = -1$. [2]



$\Longrightarrow$

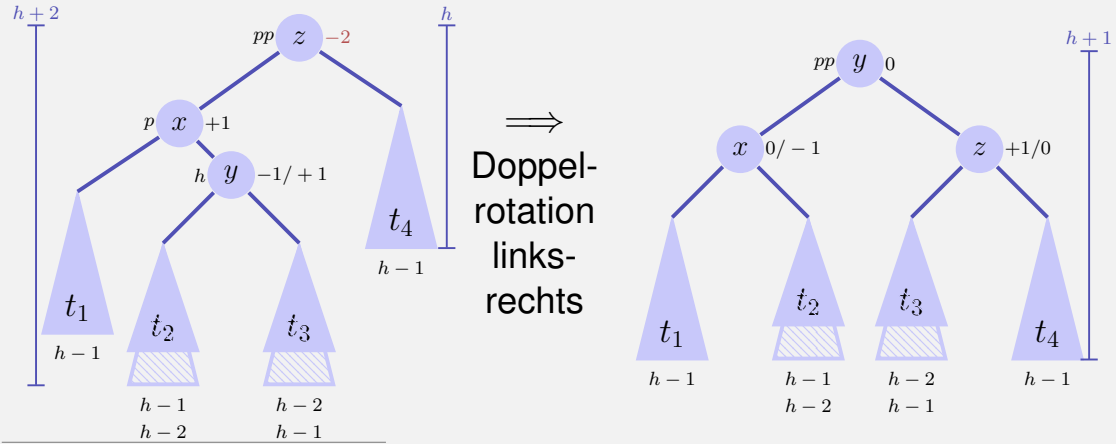
Rotation nach rechts



[2] p rechter Sohn $\Rightarrow \mathrm{bal}(pp) = \mathrm{bal}(p) = +1$, Linksrotation

Rotationen

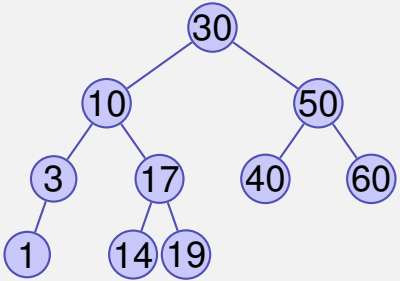
Fall 1.2 $\mathrm{bal}(p) = +1$. [3]



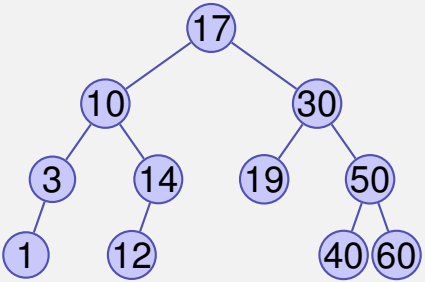
[3] p rechter Sohn $\Rightarrow \mathrm{bal}(pp) = +1, \mathrm{bal}(p) = -1$, Doppelrotation rechts links

Quiz

Fügen Sie in folgendem AVL Baum den Schlüssel 12 ein und rebalancieren Sie (wie in der Vorlesung gezeigt). Wie sieht der AVL Baum nach der in der Vorlesung gezeigten Operation aus?



Lösung



3. In-Class-Exercises

1. Rekursion auf Bäumen

Aufgabe:
Implementieren Sie eine rekursive Funktion zur Berechnung der Höhe und des Gewichts (eines Knotens) eines Suchbaumes

[Code Expert, Code Examples 6]

2. Baum Augmentieren

Aufgabe:
Augmentieren Sie die Knoten `n` eines Suchbaumes mit ihrer Höhe `n.height`. Stellen Sie sicher, dass die Höhe konsistent bleibt, auch wenn Knoten eingefügt werden.

[Code Expert, Code Examples 6]

Fragen oder Anregungen?