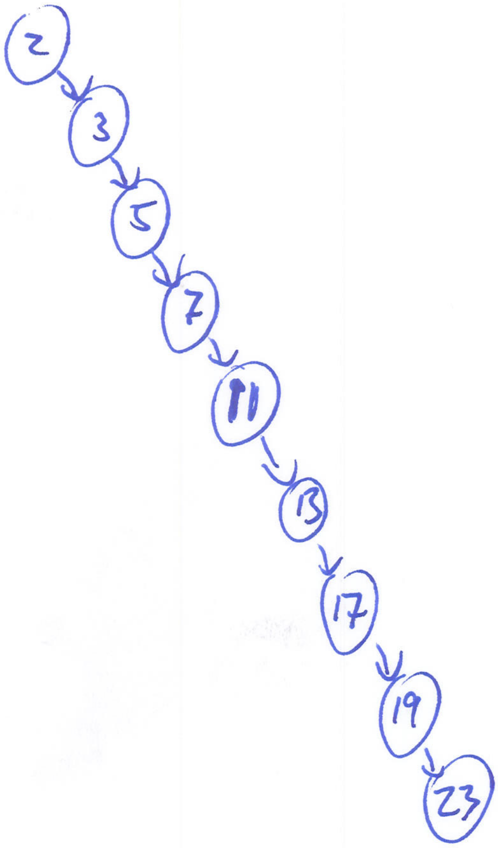
18.12.07

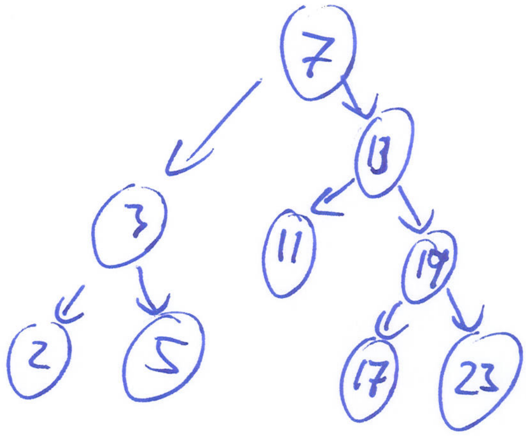
4.5 AVL-Bäume

Wichtig für binäre Suchbäume: Höhe!
(Einfügen + Entfernen in $O(h)$)

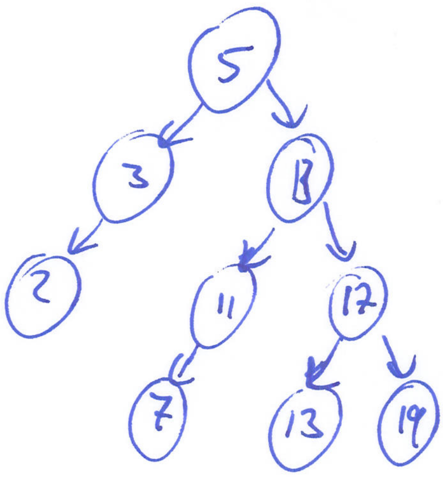
Schlecht:



Gut:



Auch gut:



Idee 1: Gestalte Baum „balanciert", d.h. linken und rechten Teilbaum gleich groß.

Problem 1: Nicht immer möglich!

Idee 2: Betrachte „annäherad balanciert"

Problem 2: Gewicht ist nicht lokal bei lokalen Änderungen, eher global...

Idee 3: wichtig ist eigentlich gar nicht das Gewicht, sondern die Tiefe!

Also:

Definition 4.7 (AVL-Baum) nach Adel'son-Vel'skij und Landis 1962

(1) Ein binärer Suchbaum ist <u>höhenbalanciert</u>, wenn sich für jeden inneren Knoten v die Höhe der beiden Kinder von v um höchstens 1 unterscheidet.

(2) Ein höhenbalancierter Suchbaum heißt auch AVL-Baum.

Problem 3: Reicht „höhenbalanciert", um logarithmische Höhe zu garantieren?

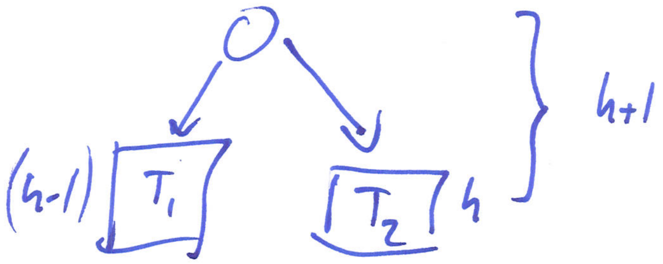
Problem 4: Wie sorgt man dafür, dass die Eigenschaft „höhenbalanciert" bei ~~INSERT~~ ~~und~~ EINFÜGEN und ENTFERNEN erhalten bleibt?

Idee 3: Wenn ein Baum höhenbalanciert ist,
Beobachtung sind es alle seine Teilbäume, d.h.
Höhenbalanciertheit ist eine rekursive Eigenschaft!

Idee ~~3~~ 4: Wieviele Knoten braucht man für einen AVL-Baum der Höhe h?

(Also: Untersuche nicht Höhe in Abhängigkeit von Knotenzahl,
SONDERN
Knotenzahl in Abhängigkeit von Höhe!)

Wenn ~~das~~ n mindestens exponentiell in h wächst, wächst h höchstens logarithmisch in n.



Mit $n(1) = 1$ und $n(2) = 2$ also

$$n(h+1) = 1 + n(h) + n(h-1)$$

~~oder~~ Erste Werte:

h	$n(h)$
1	1
2	2
3	4
4	7
5	12
6	20
7	33

Sehr ähnlich

$$f(1) = 1$$
$$f(2) = 1$$
$$f(h+1) = f(h) + f(h-1)$$

liefert 1, 1, 2, 3, 5, 8, 13, 21, 34

→ Fibonacci-Zahlen! Wachsen exponentiell...

Jetzt aber sauber:

Satz 4.7

Die Höhe eines AVL-Baumes mit n Knoten ist $O(\log n)$.

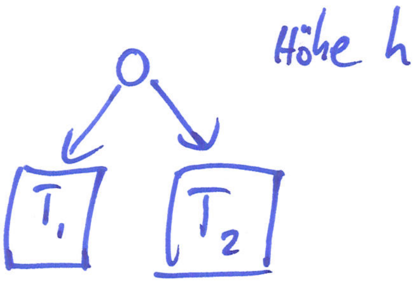
Beweis:

Statt einer oberen Schranke für die Höhe eines AVL-Baumes zeigen wir zunächst eine untere Schranke für die Zahl der Knoten eines AVL-Baumes!

Sei $n(h)$ die kleinste Zahl von Knoten eines AVL-Baumes der Höhe h.

Wir beobachten $n(1) = 1$ (ein Knoten erforderlich)
$n(2) = 2$ (zwei Knoten erforderlich wegen Höhe!)

Jetzt noch einmal:



Einer der Teilbäume hat Höhe $h-1$, der andere mindestens Höhe $h-2$.

Daraus ergibt sich

$$n(h) = 1 + n(h-1) + n(h-2).$$

Jetzt zeigen wir per Induktion:

$$\boxed{\text{Behauptung:} \quad n(h) \geq 2^{\frac{h}{2}-1}}$$

Beweis:

Induktionsanfang: Die Behauptung gilt für

$h=1$: $n(1) = 1 \geq 2^{-\frac{1}{2}} = 2^{\frac{h}{2}-1}$

$h=2$: $n(2) = 2 \geq 2^{0} = 2^{\frac{h}{2}-1}$

Induktionsschritt: Gelte die Behauptung für alle $h \in \{1, \dots, k\}$.

Wir zeigen: Die Behauptung gilt für $k+1$!

Denn: $$n(k+1) = 1 + n(k) + n(k-1)$$

Da $n(h)$ monoton wächst (für größere Höhe braucht man mindestens genausoviele Knoten!), gilt

$$n(k) \geq n(k-1)$$

Also

$$
\begin{aligned}
n(k+1) &\geq 1 + 2 \cdot n(k-1) \\
&\geq 1 + 2 \cdot 2^{\frac{k-1}{2}-1} \\
&= 1 + 2^{\left(\frac{k-1}{2}+1-1\right)} \\
&= 1 + 2^{\left(\frac{k+1}{2}-1\right)} \\
&> 2^{\frac{k+1}{2}-1} .
\end{aligned}
$$

Damit gilt die Behauptung auch für $k+1$.

Induktionsschluss: Die Behauptung gilt für alle $h \in \mathbb{N}$.

Also $\quad n(h) \geq 2^{\frac{h}{2}-1}$

und damit $\quad \log(n(h)) \geq \frac{h}{2} - 1$

oder $\quad h \leq 2 \log n + 2$.

Damit hat ein AVL-Baum ~~der Höhe~~ mit n Knoten höchstens Höhe $(2 \log n + 2)$, also $O(\log n)$.

Jetzt zurück zu Problem 4:

Wie erhält man bei Löschen und Entfernen die Höhenbalanciertheit?