Schließlich werden für den Fall (c) die Informationen für y umgespeichert und der Speicherplatz für y freigegeben.

Man sieht wieder:

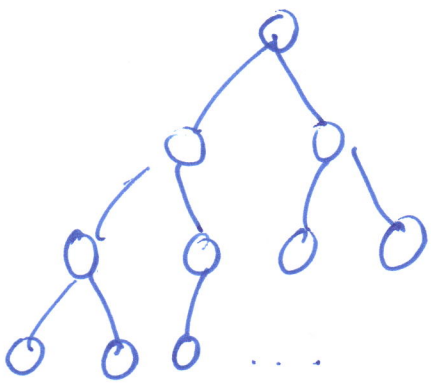
Satz 4.6

Löschen kann für einen binären Suchbaum der Höhe h in Zeit $O(h)$ vorgenommen werden.

Bew.: Klar.

Problematik: Wie kann man verhindern, dass die Höhe des Suchbaumes zu groß wird?!

Idee wäre,



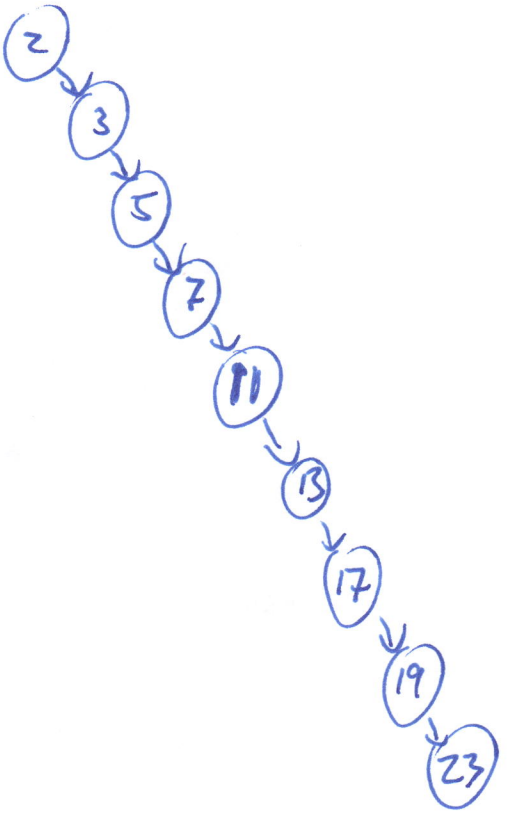
d.h. ein „balancierterer" Form als im degenerierten Fall!

4.5 AVL-Bäume

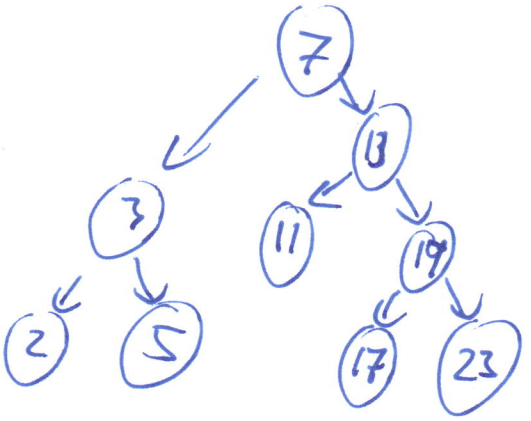
Wichtig für binäre Suchbäume: Höhe!

(Einfügen + Entfernen in $O(h)$)

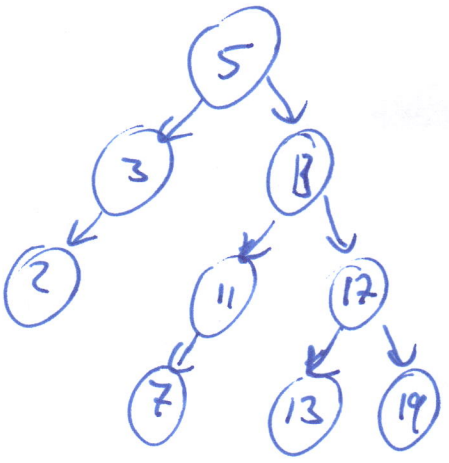
Schlecht:



Gut:



Auch gut:



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

Problem 1: Nicht immer möglich!

Idee 2: Betrachte „annähernd 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 höhenbalanciert, 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 EINFÜGEN und ENTFERNEN erhalten bleibt?

Idee 3: Beobachtung

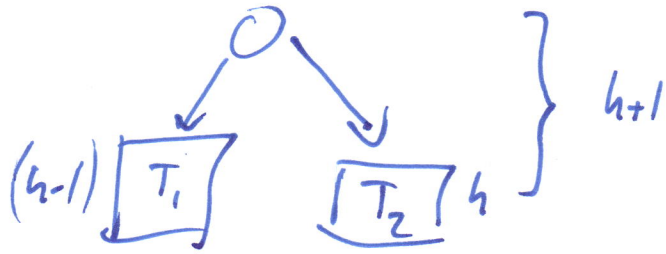
Wenn ein Baum höhenbalanciert ist, sind es alle seine Teilbäume, d.h. Höhenbalanciertheit ist eine rekursive Eigenschaft!

Idee 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 n mindestens exponentiell in h wächst, wächst h höchstens logarithmisch in n.



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

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

Erste Werte:

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

Sehr ähnlich

$$f(0) = 1$$
$$f(2) = 1$$
$$f(h+1) = f(h) + f(h-1)$$

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

$\Rightarrow$ 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!)