Das kann also zu einem degenierten Baum führen: einer Liste der Höhe $h = n$!

Abhilfe?! → Später! (Erfordert Umbau des Baumes!)

10.01.2012

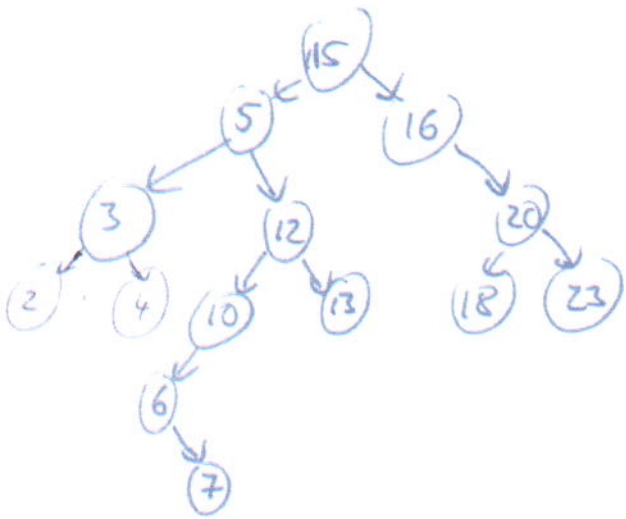
Vorher:

Löschen

Gegeben: Suchbaum T, Knoten z

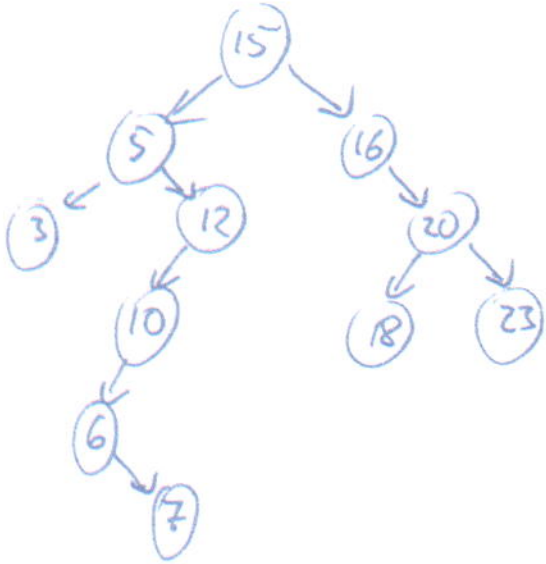
Aufgabe: z aus Baum löschen, dabei Suchbaumeigenschaft erhalten!

Vorbetrachtung:

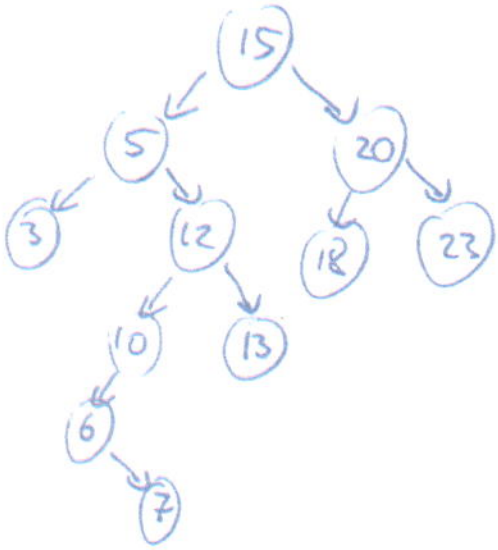


Lösche: (a) 13
(b) 16
(c) 5

(a) z hat keinen Nachfolger: löschen!



(b) z hat einen Nachfolger: ausschneiden,
d.h. Nachfolger „aufrutschen" lassen!



(c) z hat zwei Nachfolger:

Wähle einen anderen Knoten aus, der an die Stelle von z rückt!

Wichtig: (i) Suchbaumeigenschaft (Def. 4.3 (8)) muss erhalten bleiben!

(ii) ~~Grad~~ Zahl der ~~Nachfolger~~ Kinder muss noch passen

Beobachtung: Der Nachfolger von z ist ein geeigneter Kandidat!

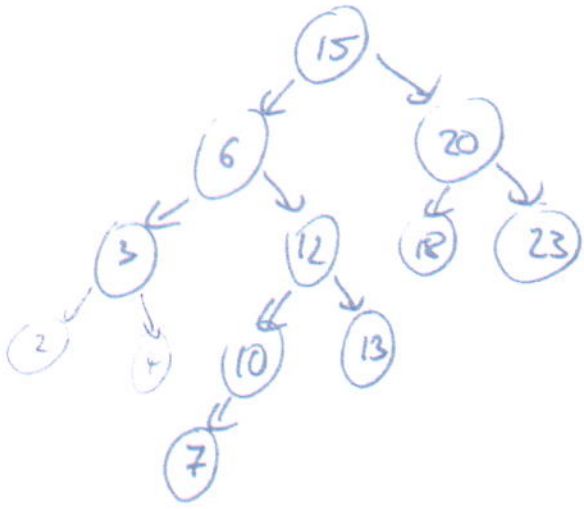
Denn: (i) Er ist größer als jeder andere Knoten im linken Teilbaum von z UND

Er ist kleiner als jeder andere Knoten im rechten Teilbaum von z

das ist kritisch! →

(ii) Er hat bislang nur ~~einen~~ ein Kind, kann also einfach ausgeschnitten werden.

Also hier:



Zusammengefasst:

Baum-Löschen (T, z)

```
1   IF (l[z] = NIL ODER r[z] = NIL) THEN
2       y := z
3   ELSE
4       y := Baum-Nachfolger (z)
5   IF (l[y] ≠ NIL) THEN
6       x := l[y]
7   ELSE
8       x := r[y]
9   IF (x ≠ NIL) THEN
10      p[x] := p[y]
```

```
11    IF ( p[y] = NIL) THEN
12        w[T] := x
13    ELSE IF ( y = l[p[y]]) THEN
14        l[p[y]] := x
15    ELSE
16        r[p[y]] := x
17    IF ( y ≠ z) THEN
18        S[z] := S[y]
19    RETURN y
```

Also: Falls nur ein oder kein Kind von z existiert, schneide z aus (Zeilen 1-2); sonst (Zeile 3) bestimme Nachfolger von z (Zeile 4).
Dieser hat nur ein Kind.
Falls dieses das linke ist, wähle dieses (Zeile 5/6), sonst das rechte (Zeile 7/8).
In der Folge (Zeile 9) ~~setzt~~ schneidet man y aus (Zeilen 10-16); dabei muss man aufpassen, weil x=NIL oder y die Wurzel sein kann.

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

Man sieht wieder:

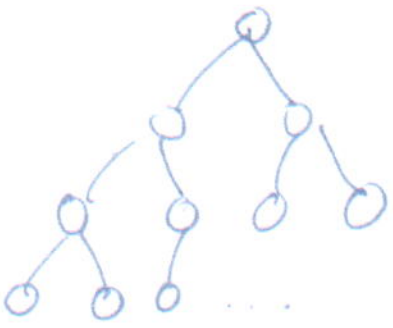
Satz 4.6

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?!

Ideal wäre,



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