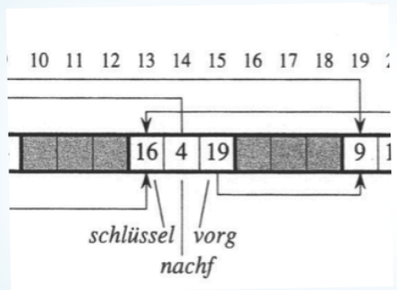
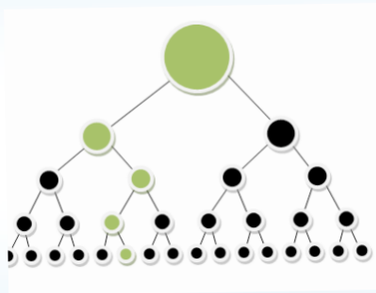





Kapitel 4.6: AVL-Bäume

Algorithmen und Datenstrukturen
WS 2025/26

Prof. Dr. Sándor Fekete

4.1 Grundoperationen

Langsam:

- ***O(n):*** ***lineare Zeit***

  Alle Objekte anschauen

Sehr schnell:

- ***O(1):*** ***konstante Zeit***

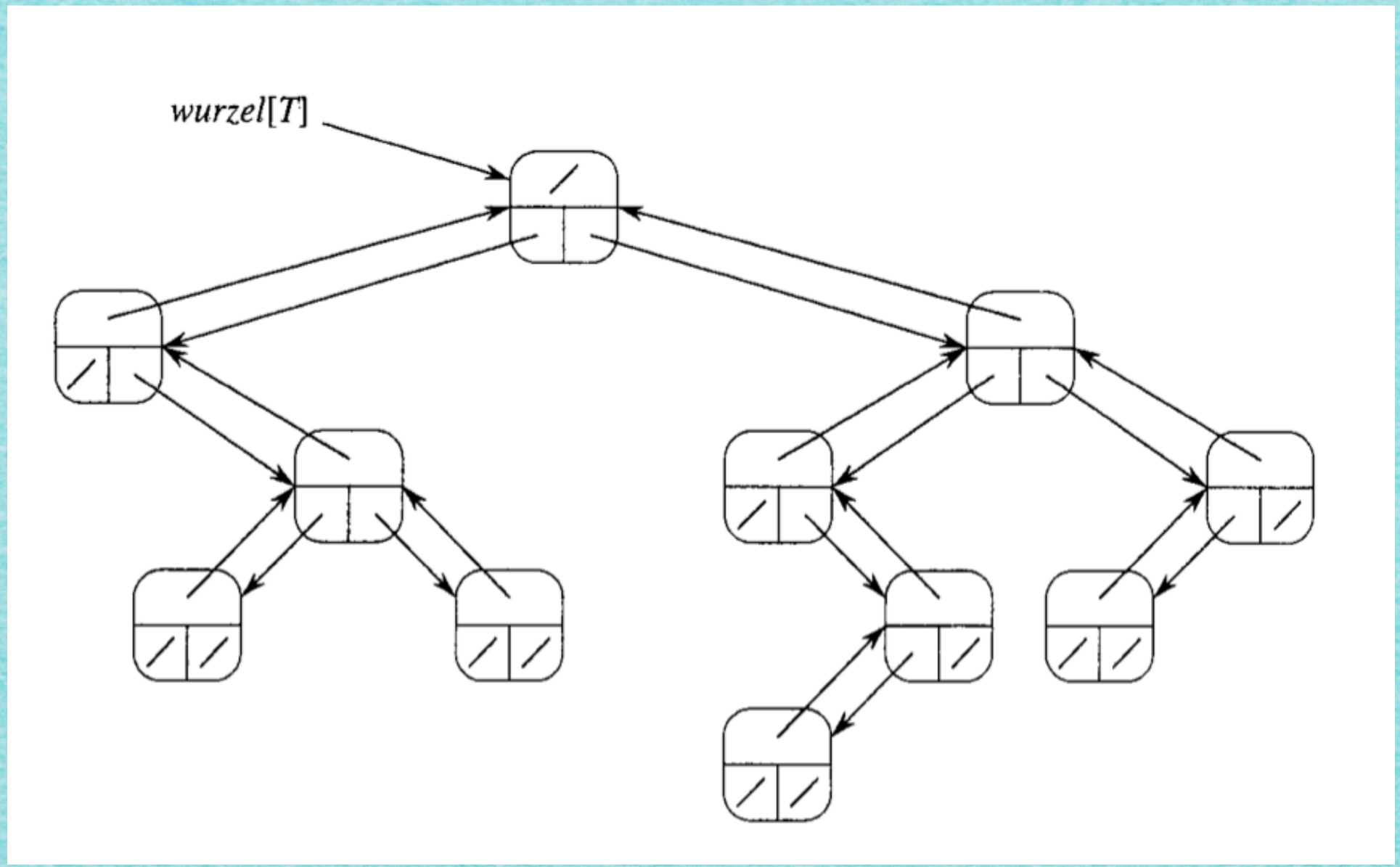
  Immer gleich schnell, egal wie groß S ist.

Schnell:

- ***O(log n):*** ***logarithmische Zeit***

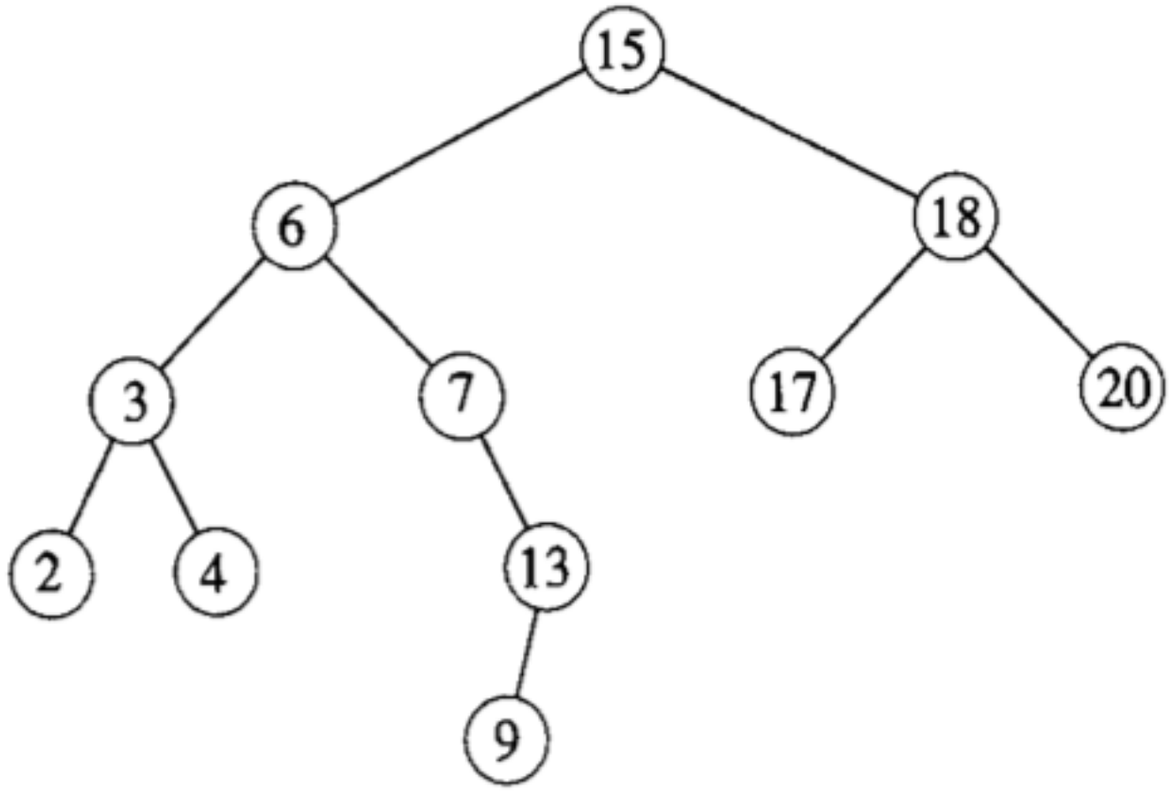
  Wiederholtes Halbieren

Binärer Suchbaum



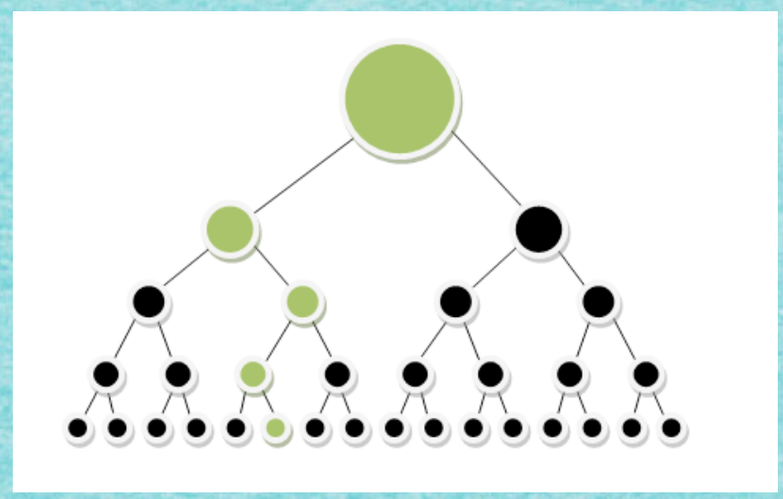
Außerdem wichtig: Struktur der Schlüsselwerte!

Suche im Suchbaum



ITERATIVE-TREE-SEARCH(x, k)

```
1  while x ≠ NIL und k ≠ schlüssel[x]
2        do if k < schlüssel[x]
3              then x ← links[x]
4              else  x ← rechts[x]
5  return x
```

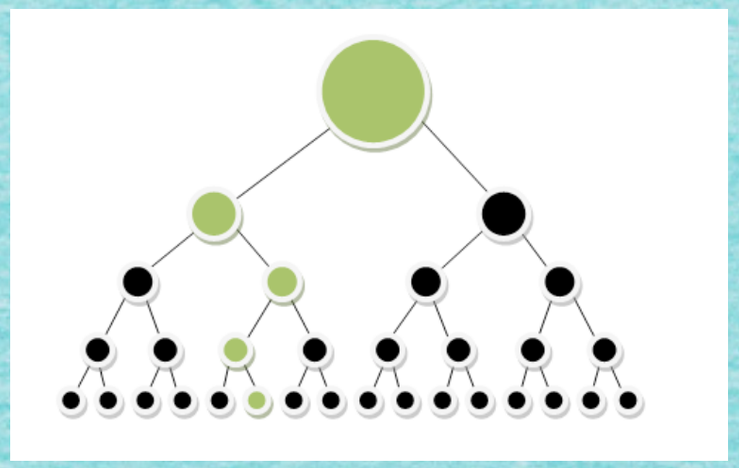


Satz 4.4

Suchen, Minimum, Maximum, Nachfolger, Vorgänger können in einem binären Suchbaum der Höhe h in Zeit O(h) durchlaufen werden.

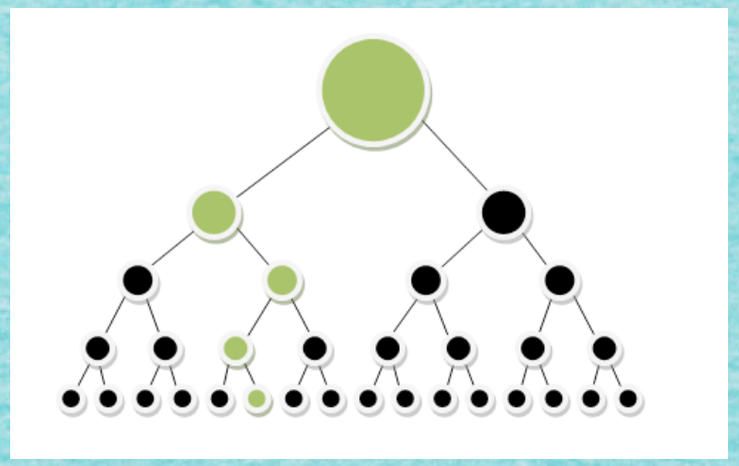
Beweis:

Klar, der Baum wird nur einmal vertikal durchlaufen!



Satz 4.5
Einfügen benötigt O(h) für einen binären Suchbaum der Höhe h.

Beweis:
Klar, der Baum wird nur vertikal abwärts durchlaufen!

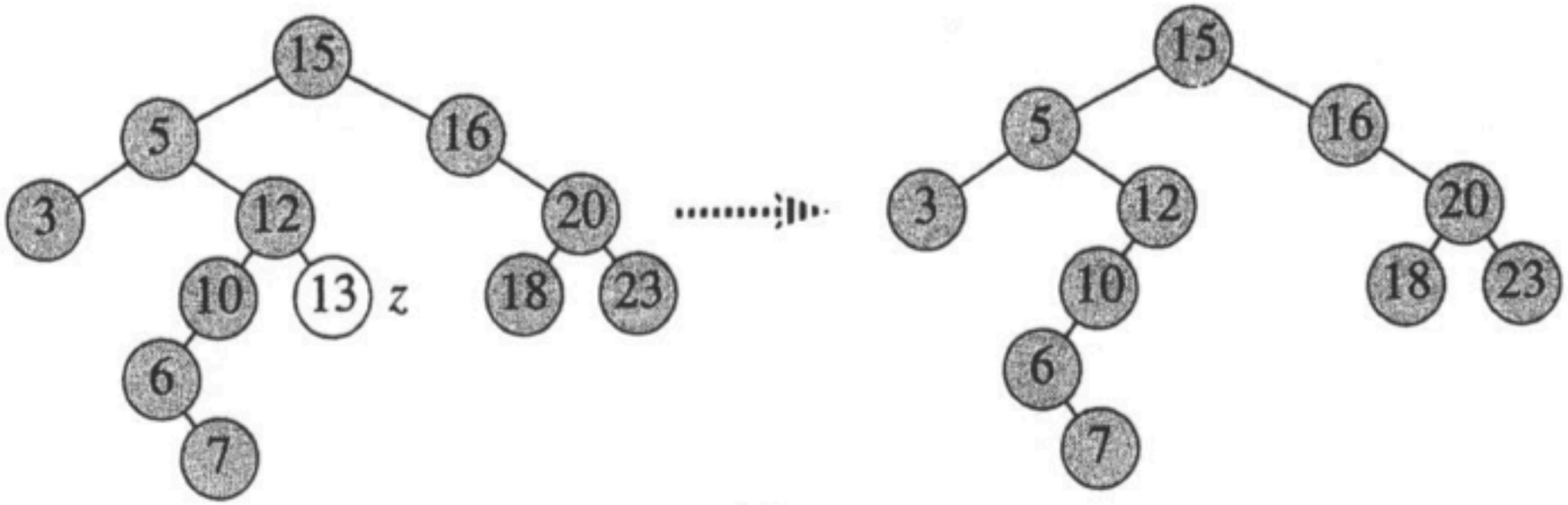


4.1 Grundoperationen

<u>DELETE(S,x):</u> **“Entferne x aus S”**

Lösche das unter der Adresse x stehende Element aus der Menge S.

Löschen im Suchbaum



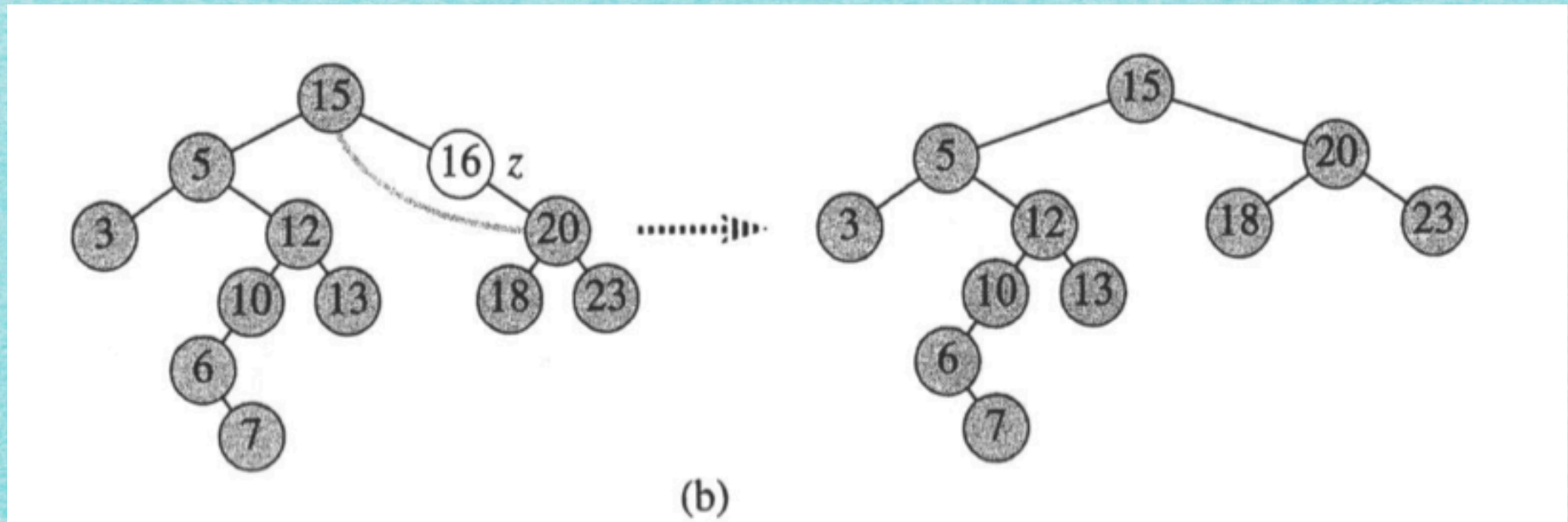
(a)

Lösche 13!

(a) Keine Kinder:

Einfach entfernen

Löschen im Suchbaum



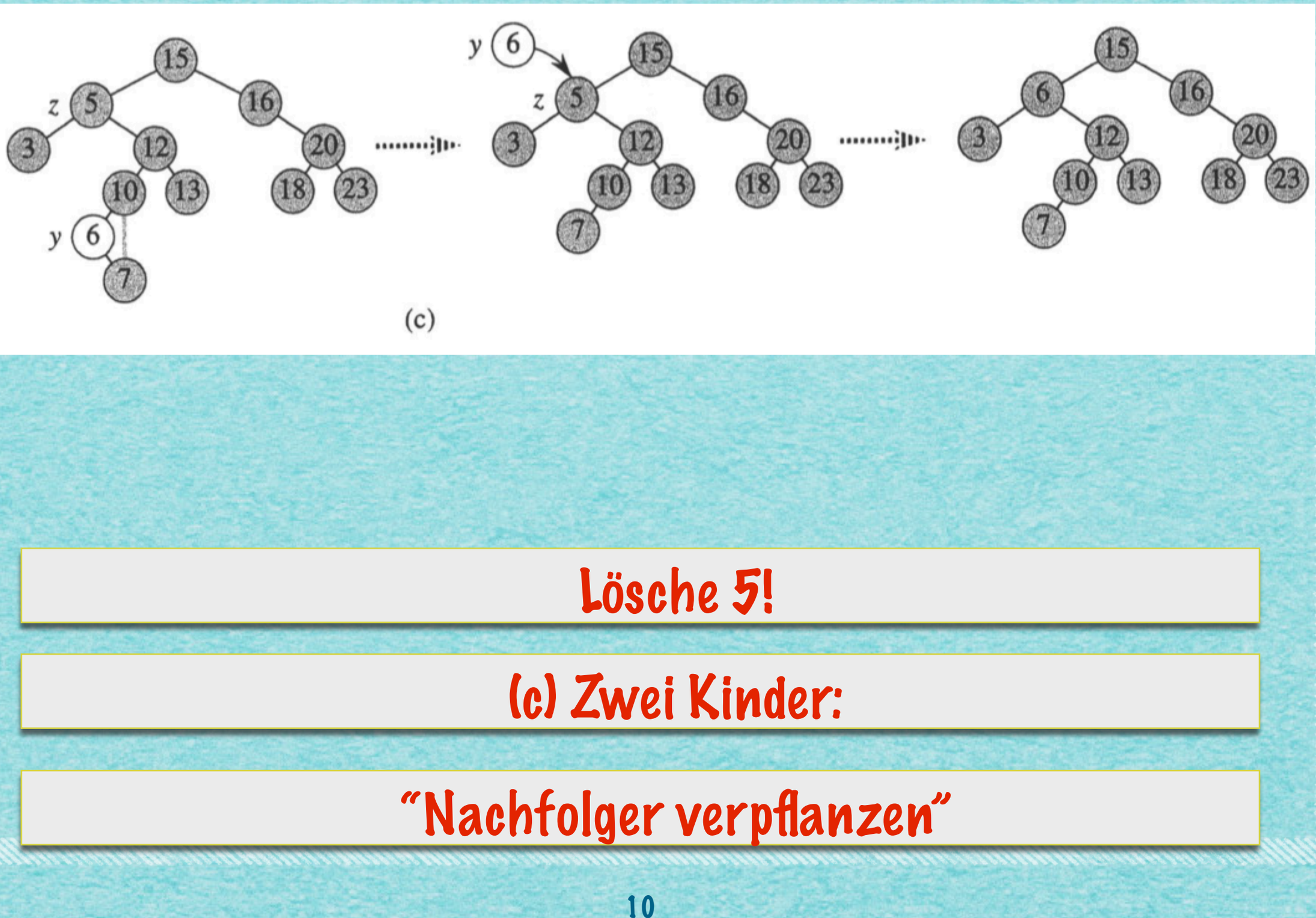
(b)

Lösche 16!

(b) Ein Kind:

"Ausschneiden"

Löschen im Suchbaum



(c)

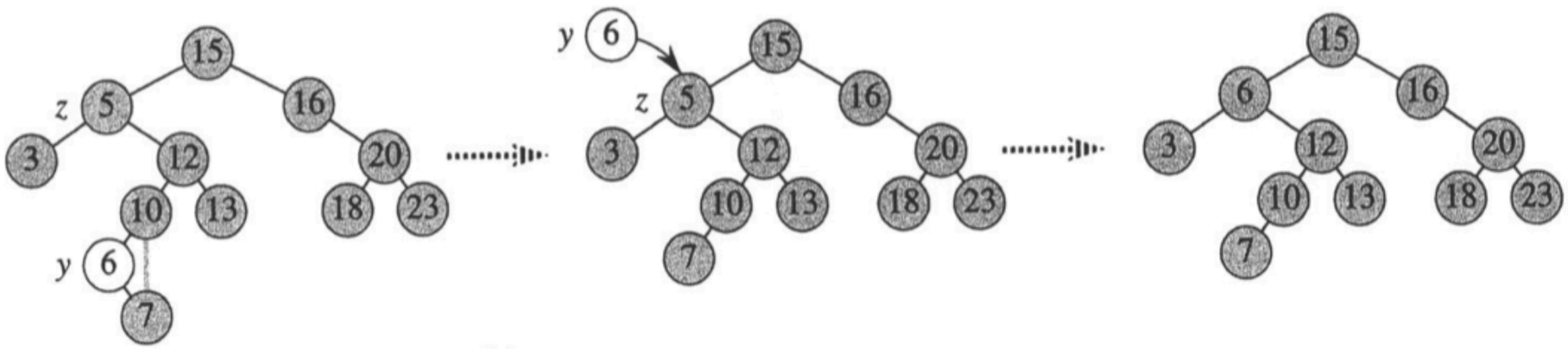
Lösche 5!

(c) Zwei Kinder:

"Nachfolger verpflanzen"

Löschen im Suchbaum

```
TREE-DELETE(T, z)
 1  if links[z] = NIL oder rechts[z] = NIL
 2      then y ← z
 3      else y ← TREE-SUCCESSOR(z)
 4  if links[y] ≠ NIL
 5      then x ← links[y]
 6      else x ← rechts[y]
 7  if x ≠ NIL
 8      then p[x] ← p[y]
 9  if p[y] = NIL
10      then wurzel[T] ← x
11      else if y = links[p[y]]
12              then links[p[y]] ← x
13              else rechts[p[y]] ← x
14  if y ≠ z
15      then schlüssel[z] ← schlüssel[y]
16           kopiere die Satellitendaten von y in z
17  return y
```



(c)

Satz 4.6
Löschen benötigt O(h) für einen binären Suchbaum der Höhe h.

Beweis:
Klar, der Baum wird i.W. nur einmal vertikal durchlaufen!

4.5 Binäre Suchbäume

<u>Schnell:</u>

- ***O(log n): logarithmische Zeit***
- ***O(h): Tiefe des Baumes***

<u>Also: Wie können wir die Tiefe des Baumes auf O(log n) beschränken?</u>

Ab an die Tafel!

s.fekete@tu-bs.de