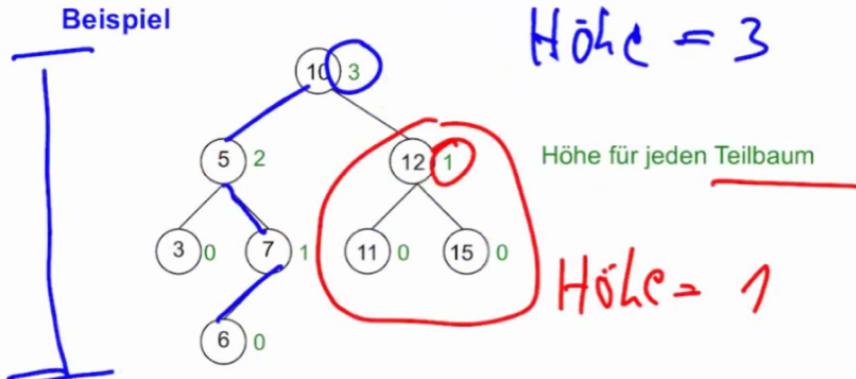


S3-3

## Höhe eines Baums

Höhe = Maximale Anzahl von Kanten von seiner Wurzel zu einem Blatt.



### Beachte

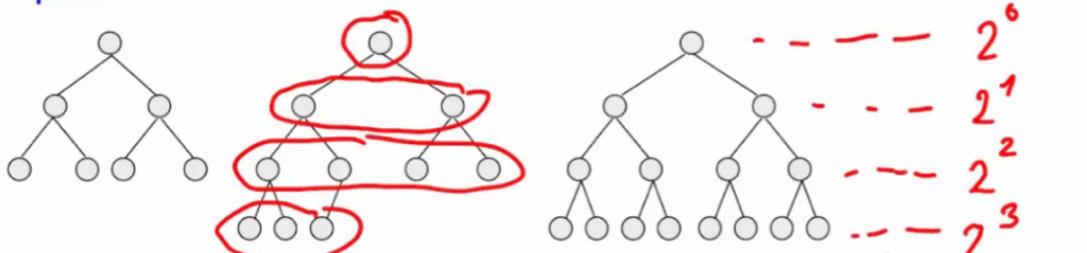
- Ein Baum, der nur aus einem Knoten besteht, besitzt die Höhe 0.
- Aus technischen Gründen wird die Höhe eines leeren Baums (d.h. Anzahl Knoten = 0) als -1 definiert.

S3-4

## Vollständiger Binärbaum

Ein vollständiger Binärbaum ist ein Binärbaum, bei dem jede Ebene (bis auf die letzte) vollständig gefüllt und die letzte Ebene von links nach rechts gefüllt ist.

### Beispiele



### Eigenschaften

- Ein vollständiger Binärbaum mit  $n$  Knoten hat die Höhe  $h = \lfloor \log_2 n \rfloor$ .
- Vollständige Binärbäume sind (bei einer gegebenen Knotenzahl) Binärbäume mit einer minimalen Höhe.

$$n = 2^{h+1} - 1$$

S3-19

## Löschen in binären Suchbäumen (4) Rückgabe

```

private V oldValue; // Rückgabeparameter

public V remove(K key) {
    root = removeR(key, root);
    return oldValue;
}

private Node<K,V> removeR(K key, Node<K,V> p) {
    if (p == null) { oldValue = null; }
    else if (key.compareTo(p.key) < 0)
        p.left = removeR(key, p.left);
    else if (key.compareTo(p.key) > 0)
        p.right = removeR(key, p.right);
    else if (p.left == null || p.right == null) {
        // p muss gelöscht werden
        // und hat ein oder kein Kind:
        oldValue = p.value;
        p = (p.left != null) ? p.left : p.right;
    } else {
        // p muss gelöscht werden und hat zwei Kinder:
        MinEntry<K,V> min = new MinEntry<K,V>();
        p.right = getRemMinR(p.right, min);
        oldValue = p.value;
        p.key = min.key;
        p.value = min.value;
    }
    return p;
}

```

```

private Node<K,V> getRemMinR(Node<K,V> p, MinEntry<K,V> min) {
    assert p != null;
    if (p.left == null) {
        min.key = p.key;
        min.value = p.value;
        p = p.right;
    } else
        p.left = getRemMinR(p.left, min);
    return p;
}

private static class MinEntry<K, V> {
    private K key;
    private V value;
}

```

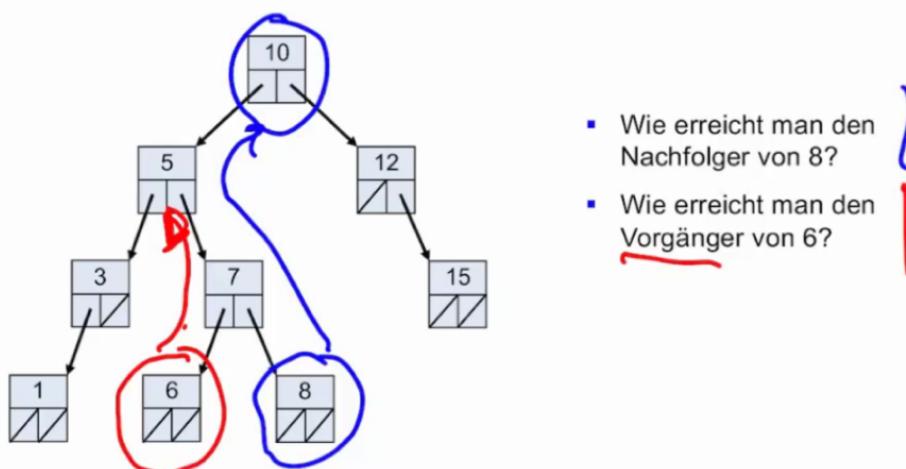
- getRemMinR löscht im Baum p den Knoten mit kleinstem Schlüssel und liefert Schlüssel und Daten des gelöschten Knotens über min zurück
- MinEntry ist ein Hilfsdatentyp für den Rückgabe-Parameter min von getRemMinR

S3-22

## Iterative Traversierung von Binärbäumen

### Problem

- In binären Suchbäumen gibt es für einen Knoten im allgemeinen keinen effizienten Zugriff auf seinen InOrder-Vorgänger bzw. -Nachfolger.
- Daher ist mit der bisher besprochenen Datenstruktur für Suchbäume keine effiziente Vorwärts- bzw. Rückwärtstraversierung mit Iteratoren machbar.



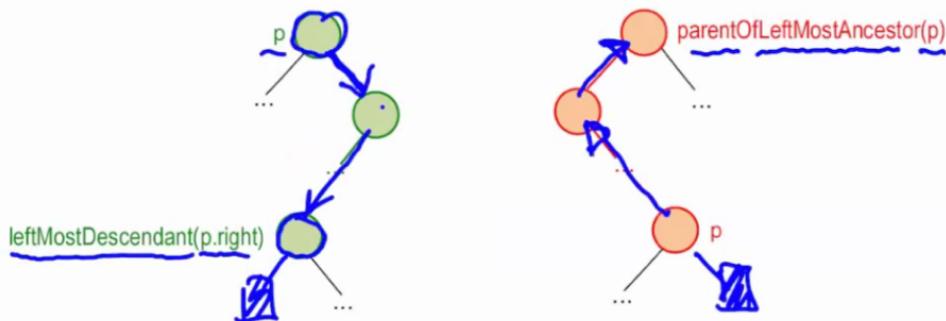
→ Schau wie inOrder aussieht, dann merkst du warum 5 Vorgänger von 6 ist und 10 Nachfolger von 10

## Iterative Traversierung von Bäumen

- Hier: nur Traversierung zum InOrder-Nachfolger.  
Traversierung zum InOrder-Vorgänger geht analog.
- Sei  $p$  der aktuelle Knoten. Gesucht ist der Nachfolger von  $p$ .  
Wir unterscheiden zwei Fälle:

```

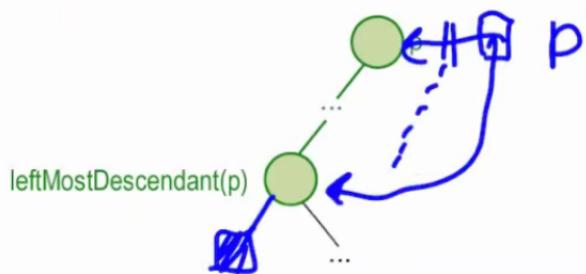
if (p.right != null) |
    p = leftMostDescendant(p.right);
else
    p = parentOfLeftMostAncestor(p);
  
```



`leftMostDescendant`: einmal nach rechts, dann nach links bis es nicht geht

## Implementierung von leftMostDescendant

```
private Node<K,V> leftMostDescendant(Node<K,V> p) {  
    assert p != null;  
    while (p.left != null)  
        p = p.left;  
    return p;  
}
```



## Implementierung von parentOfLeftMostAncestor

```
private Node<K,V> parentOfLeftMostAncestor(Node<K,V> p) {
    assert p != null;
    while (p.parent != null && p.parent.right == p)
        p = p.parent;
    return p.parent;      // kann auch null sein
}
```



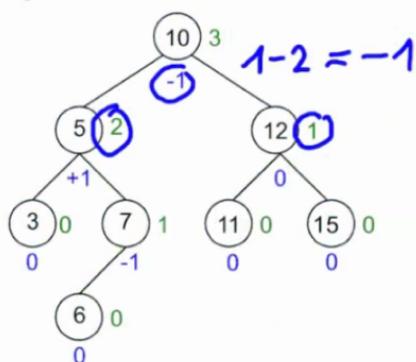
While wird nur ausgeführt wenn p rechtes Kind vom Eltern ist.

S3-35

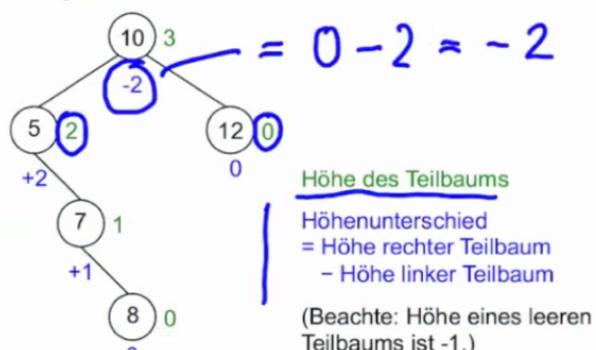
## Definition von AVL-Bäumen

- Ein binärer Suchbaum heißt **AVL-Baum** oder (höhen)balenzierter Baum, falls sich für jeden Knoten die Höhen der beiden Teilbäume um höchstens 1 unterscheiden.
- Die Abkürzung AVL geht zurück auf Adelson-Velskij und Landis.

### Beispiel für AVL-Baum:



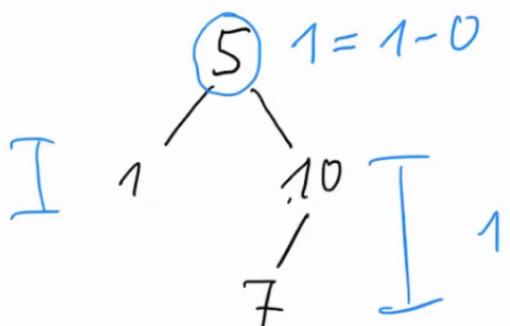
### Beispiel für Nicht-AVL-Baum



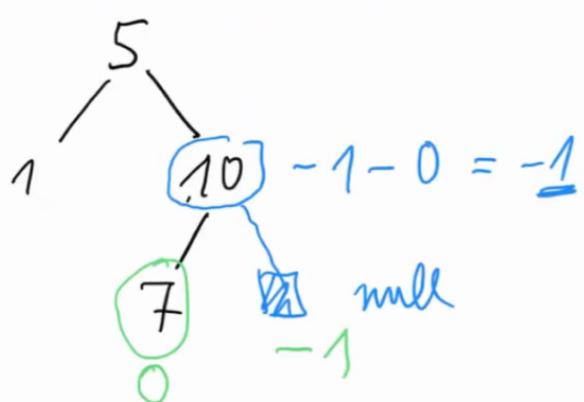
### Eigenschaft

- Ein AVL-Baum hat eine maximale Höhe von etwa  $1.5 \log_2 n$ . [Ottmann u. Widmayer].
- Damit lassen sich alle Dictionary-Operationen in  $O(\log n)$  realisieren.

wie kann die höhe von einem teilbaum -1 sein?



Höhen -  
unterschied



Höhen -  
unterschied

Höhe

