Binäre Suchbäume – Löschen

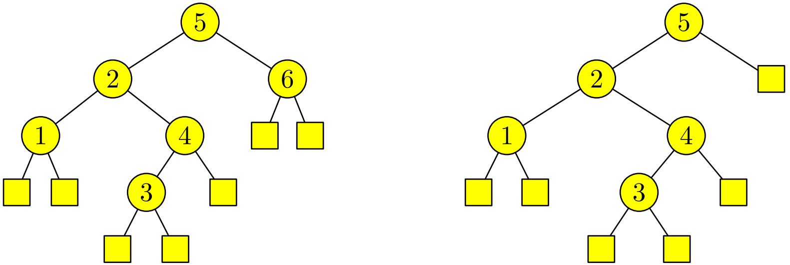
Beim Löschen unterscheiden wir drei Fälle:

- Blatt (einfach)
- Der Knoten hat kein linkes Kind (einfach)
- Der Knoten hat ein linkes Kind (schwierig)

Damit sind alle Fälle abgedeckt!
Warum kein Fall: Kein rechtes Kind?

Binäre Suchbäume – Löschen

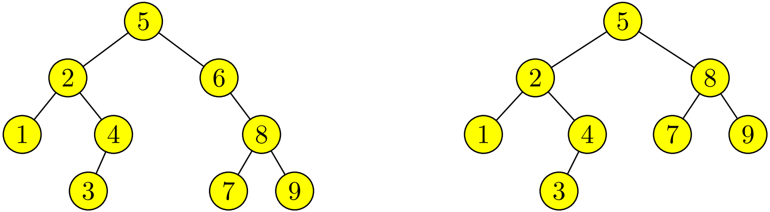
Löschen eines Blatts:



Ein Blatt kann durch Zeigerverbiegen gelöscht werden.

Binäre Suchbäume – Löschen

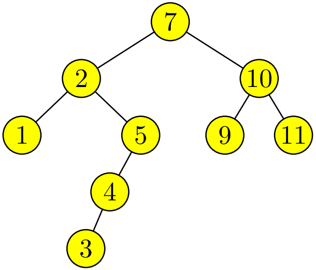
Löschen eines Knotens ohne linkes Kind:



Wir können kopieren oder Zeiger verbiegen.

Binäre Suchbäume – Löschen

Löschen eines Knotens mit linkem Kind:



1. Finde den größten Knoten im linken Unterbaum
2. Kopiere seinen Inhalt
3. Lösche ihn

Binäre Suchbäume – Löschen

In der Klasse insert():

Java

```
public void delete(K k) {
  if(root == null) return;
  if(root.left == null && root.right == null && root.key == k)
    root = null;
  else {
    Searchtreenode⟨K, D⟩ n = root.findsubtree(k);
    if(n ≠ null) n.delete();
  }
}
```

Binäre Suchbäume – Löschen

Java

```
void delete() {
  if(left == null && right == null) {
    if(parent.left == this) parent.left = null;
    else parent.right = null; }
  else if(left == null) {
    if(parent.left == this) parent.left = right;
    else parent.right = right;
    right.parent = parent; }
  else {
    Searchtreenode⟨K, D⟩ max = left;
    while(max.right ≠ null) max = max.right;
    copy(max); max.delete();
  }
}
```

Binäre Suchbäume – Löschen

In **null** jetzt korrekt:

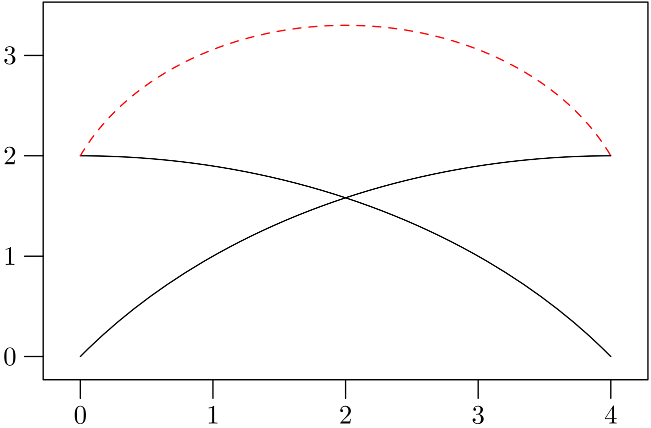
Java

```
public void delete(K k) {
  if(root == null) return;
  if(root.key.equals(k))
    if(root.left == null && root.right == null) {
      root = null; return;
    }
    else if(root.left == null) {
      root = root.right; root.parent = null; return;
    }
  Searchtreenode⟨K, D⟩ n = root.findsubtree(k);
  if(n ≠ null) n.delete();
}
```

Binäre Suchbäume – Beispiel

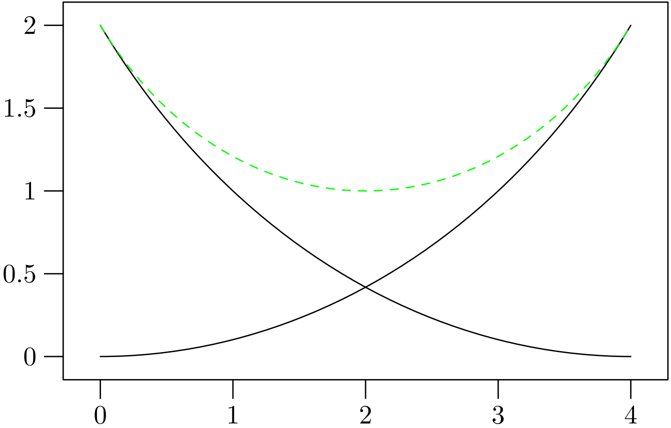
Die Schlüssel von 1 bis 40 werden zufällig eingefügt oder gelöscht.

Binäre Suchbäume – Analyse



Summe schlechte Näherung des Maximums.

Binäre Suchbäume – Analyse



Summe gute Näherung des Maximums.

Die Kurven sind steiler.

Binäre Suchbäume – Analyse

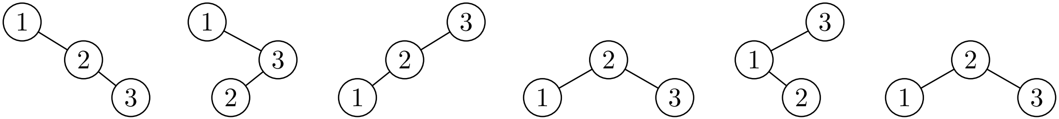
Wir fügen die Knoten $1, \ldots, n$ in zufälliger Reihenfolge in einen leeren Suchbaum ein.

Sei T_n die Höhe dieses Suchbaums.

Wir interessieren uns für $E(T_n)$.

Wir betrachten erst einmal T_0, T_1, T_2 und T_3:

- $E(T_0) = 0$
- $E(T_1) = 1$
- $E(T_2) = 2$
- $E(T_3) = 8/3$



Allgemeiner Fall:

Die Wurzel des Baums enthält $W \in \{1, \ldots, n\}$.

$$\Pr[W = k] = 1/n \text{ für } k \in \{1, \ldots, n\}$$

Wie sieht der Rest des Baums aus, falls $W = k$?

In den linken Teilbaum wurden $\{1, \ldots, k-1\}$ in zufälliger Reihenfolge eingefügt.

Seine Höhe ist T'_{k-1}.

Die Höhe des rechten Teilbaums ist T''_{n-k}.
Die Gesamthöhe ist $T_n = \max\{T'_{k-1}, T''_{n-k}\} + 1$.

Die Gesamthöhe ist $T_n = \max\{T'_{k-1}, T''_{n-k}\} + 1$.

$$E(T_n) = \frac{1}{n} \sum_{k=1}^{n} E(\max\{T'_{k-1}, T''_{n-k}\} + 1)$$

Wir können das Maximum durch die Summe abschätzen:

$$E(T_n) \leq \frac{1}{n} \sum_{k=1}^{n} \Big(E(T_{k-1}) + E(T_{n-k}) + 1 \Big)$$

Leider zu grob! Führt zu einer schlechten Abschätzung.