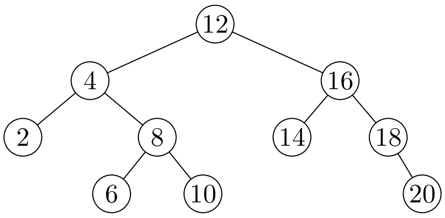
**RWTH Aachen**
**Lehrgebiet Theoretische Informatik**
Kneis—Mölle—Richter—Rossmanith

**SoSe 2007**
Blatt 4
30.4.2007

## Übungen zur Vorlesung Datenstrukturen und Algorithmen

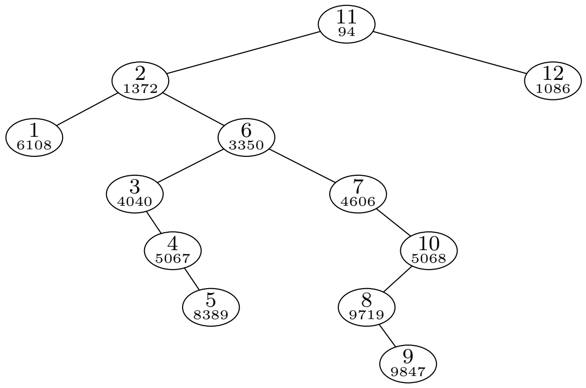
### T7

Gegeben ist der folgende AVL-Baum:




Was passiert, wenn wir erst 21 und dann 17 einfügen, danach erst 12 und dann 2 löschen? Schließlich wird die 12 wieder eingefügt. Wie sieht der Baum am Ende aus?

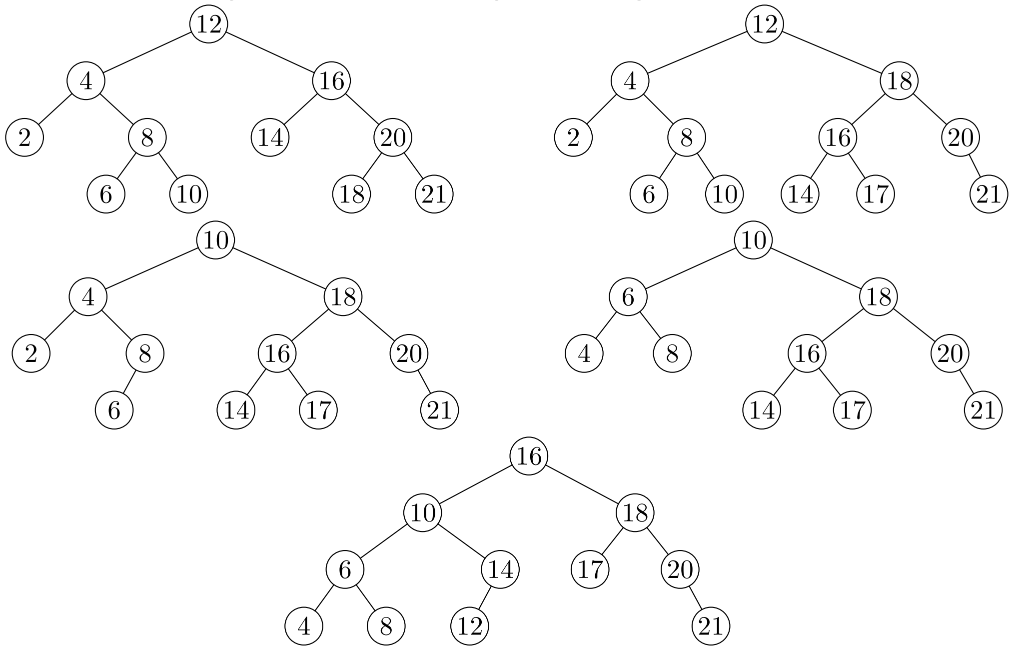
### T8



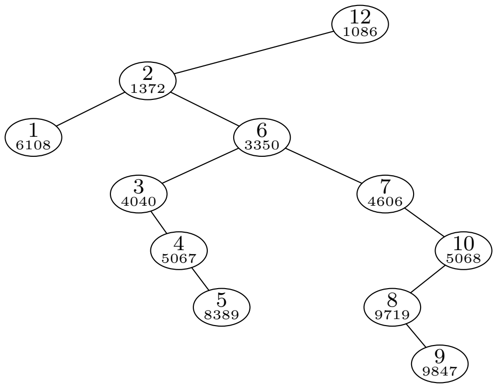

Was passiert, wenn die 11 aus obigen Treap gelöscht und anschließend mit Priorität 3000 wieder eingefügt wird?

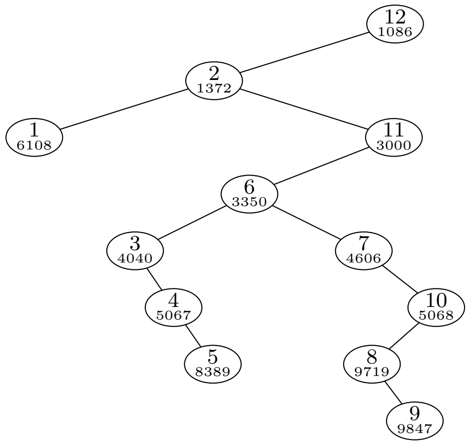
### Lösungsvorschlag T7:

Wir sollten uns streng an die in der Vorlesung erörterten Operationen halten. Dann ergeben sich für die obigen fünf Anwendungen die folgenden AVL-Bäume:



**Lösungsvorschlag T8:**
Wir sollten uns streng an die in der Vorlesung erörterten Operationen halten. Dann ergeben sich die folgenden Treaps:




**T9**
Beantworten Sie diese Fragen:

a) Führt jede Eingabereihenfolge der Schlüssel $1, \ldots, n$ in einen leeren Suchbaum zu verschiedenen Suchbäumen?

b) Vorgegeben sei ein beliebiger binärer Baum, der die AVL-Eigenschaft erfüllt. Kann man durch die Einfüge- und Löschoperationen für AVL-Bäume einen AVL-Baum mit genau dieser Struktur erzeugen?

c) Können Sie eine Operation *Split* für Treaps angeben, die einen Treap für einen gegebenen Schlüssel $s$ in zwei Treaps $T_1, T_2$ unterteilt, so daß alle Schlüssel in $T_1$ kleiner und alle Schlüssel in $T_2$ größer als $s$ sind?

**Lösungsvorschlag:**

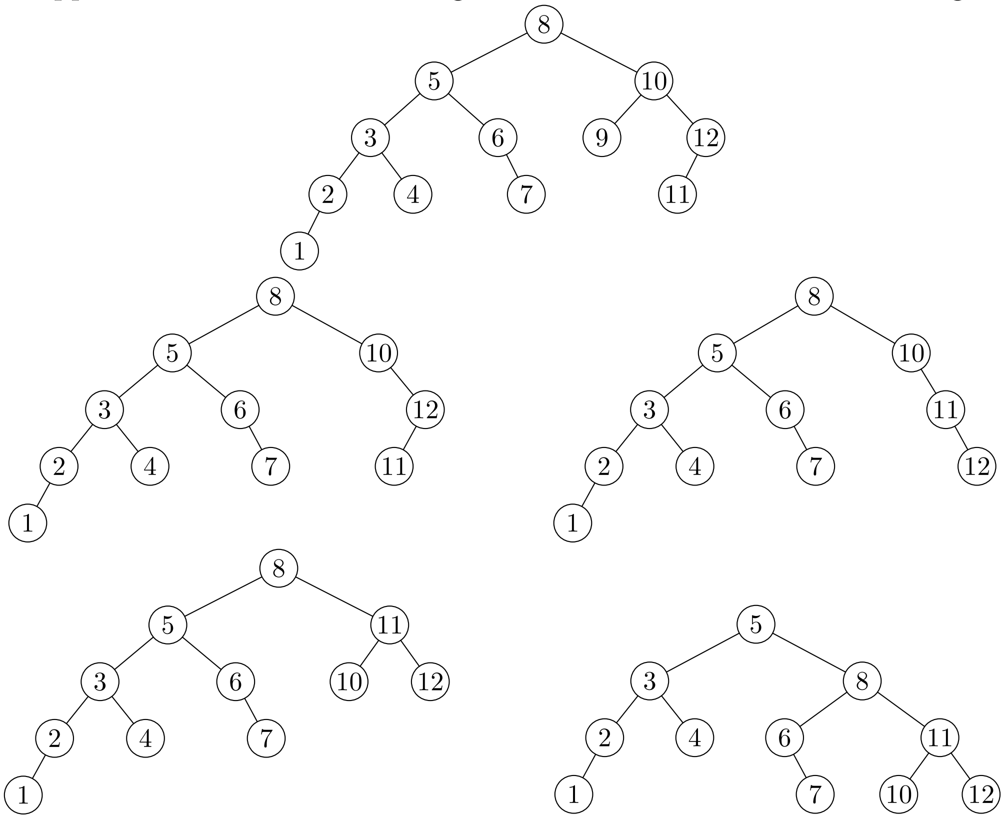
a) Während dies für $n = 1, 2$ noch gilt, ist die Aussage ab $n = 3$ falsch. So führen beispielsweise die Eingaben $2, 1, 3$ und $2, 3, 1$ zu denselben Suchbäumen. Dieses Gegenbeispiel läßt sich offenbar leicht verallgemeinern.

b) Ja, dies ist immer möglich. Es genügt hierzu, die Elemente des vorgegebenen Baums schichtenweise und von der Wurzel aus in einen anfangs leeren AVL-Baum einzufügen. Zu jedem Zeitpunkt ist der so entstehende Baum ein Suchbaum mit der AVL-Eigenschaft.

c) Ja, hoffentlich. Ist der Schlüssel $s$ vorhanden, wird er zunächst gelöscht. Danach wird $s$ mit der Priorität $-\infty$ neu eingefügt und dabei natürlich zur Wurzel durchrotiert. Nun ergeben der linke und rechte Unterbaum die gewünschten Treaps. Die Laufzeit für das Löschen und Einfügen von $s$ ist linear in der Höhe des ursprünglichen Treaps.

**H8 (10 Punkte)**
Geben Sie einen AVL-Baum und einen Schlüssel $k$ an, so daß das Löschen von $k$ genau drei Rotationen verursacht.
Wie haben Sie diesen Baum konstruiert?

**Lösungsvorschlag:**
Die Lösungsidee ist folgende. Wir haben schon in der Aufgabe T7 gesehen, wie man eine Doppelrotation herbeiführen kann. Ausgehend von der dafür erforderlichen Baumstruktur können wir den Baum nach oben so erweitern, daß eine weitere Rotation erforderlich ist, weil die Doppelrotation einen darüberliegenden Knoten aus der Balance bringt.



**H9 (10 Punkte)**
Erweitern Sie Treaps um die Operation *Merge*, die zwei Treaps $T_1, T_2$ zu einem neuen Treap zusammenfügt. Dabei wird vorausgesetzt, daß alle Schlüssel in $T_1$ kleiner sind als alle Schlüssel in $T_2$. Die Laufzeit der Operation muß auf $O(h_1+h_2)$ beschränkt sein, wobei $h_i$ die Höhe von $T_i$ bezeichne.

**Lösungsvorschlag:**
Zuerst wird ein Schlüssel $s$, der größer als alle Schlüssel in $T_1$ und kleiner als alle Schlüssel in $T_2$ ist, in $T_1$ eingefügt, indem das normale Einfügen in Treaps verwendet wird und $-\infty$ als Priorität gewählt wird. Dadurch ist dann $s$ die neue Wurzel von $T_1$ und hat kein rechtes Kind.

Jetzt machen wir die Wurzel von $T_2$ zum rechten Kind von $s$. Offensichtlich erfüllt dieser neue Baum die Suchbaum- und die Heapeigenschaften, ist also ein Treap. Abschließend wird $s$ aus dem Treap gelöscht.

Das Einfügen von $s$ benötigt Zeit $O(h_1)$. Danach kann der kombinierte Treap in konstanter Zeit konstruiert werden. Das Löschen von $s$ erfordert dann Zeit $O(h_1 + h_2)$.