Dr. Andreas Abel
Dr. Ulrich Schöpp
Dr. Steffen Jost

Ludwig-Maximilians-Universität München
Institut für Informatik
17. Januar 2011

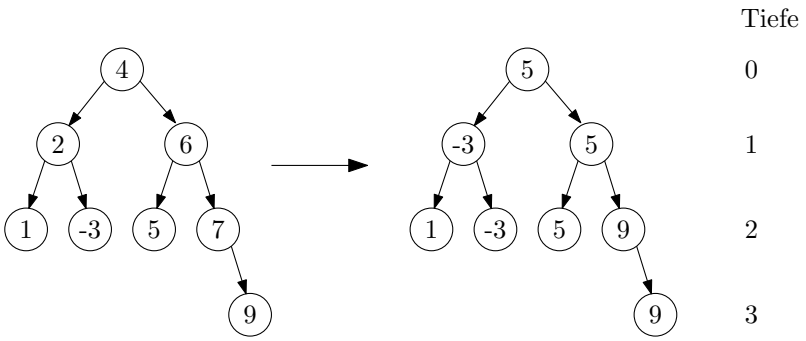
12. Musterlösung zur Vorlesung

Einführung in die Informatik: Programmierung und Software-Entwicklung

Aufgabe 12-1 ***Max-Min Berechnung*** **(4 Punkte)** (Geben Sie alle `.java`-Dateien Ihrer Lösung ab)
In dieser Aufgabe sollen Sie eine Max-Min-Berechnung auf Bäumen implementieren.

Gegeben ist ein Binärbaum mit einem ganzzahligen Wert in jedem Knoten. Diese Werte sollen so verändert werden, dass am Ende folgende Bedingungen gelten. Jeder Blattknoten behält seinen Wert. Für alle anderen Knoten gilt: Hat der Knoten gerade Tiefe, so ist sein Wert das Maximum der Werte seiner Kindknoten. Hat der Knoten ungerade Tiefe, so ist sein Wert das Minimum der Werte seiner Kindknoten. Beachten Sie, dass alle Werte, die in diesen Bedingungen erwähnt werden, die Werte im Endergebnisbaum sind.

Zum Beispiel wird der linksstehende Baum folgendermaßen verändert:



Erweitern Sie die Klasse `BinTree` für Binärbäume aus der Angabe zum vorherigen Übungsblatt 11 um eine Methode `void computeMinMax()`, welche den Baum nach obiger Vorschrift verändert. Es bietet sich dafür an, die gegebenen Klassen noch um weitere eigene Methoden zu erweitern.

LÖSUNG: Wir lösen die Aufgabe wieder durch Rekursion. Wie in Aufgabe 11-1 findet die Hauptarbeit in der Methode innerhalb der Klasse `Node` statt. Die Methode in der Klasse `BinTree` prüft lediglich ob der der Baum leer ist:

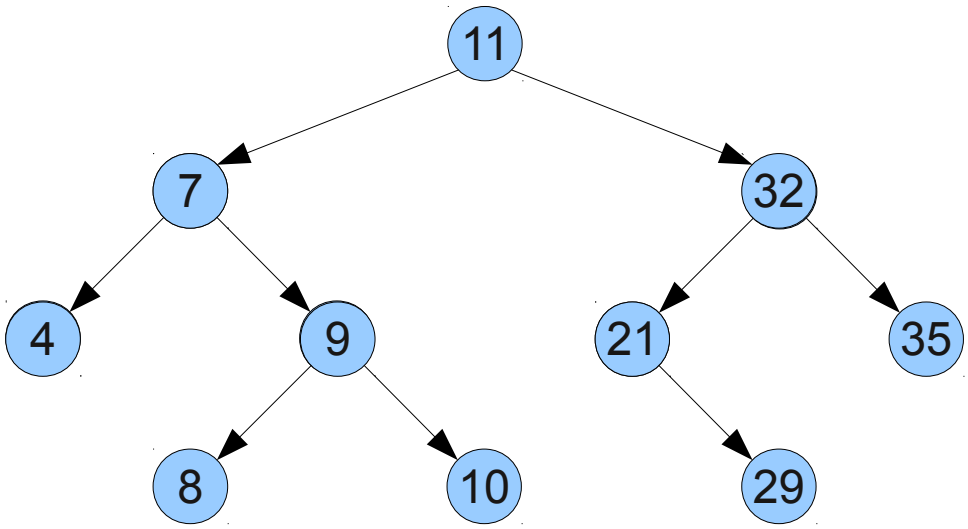
```
public void computeMinMax() {
    if (root != null) {
        root.computeMinMax(true);
    }
}
```

Die Methode `computeMinMax` innerhalb der Klasse `Node` erhält ein boolesches Argument, welches mitzählt ob wir uns in gerader oder ungerader Tiefe befinden. Für jeden rekursiven Aufruf auf einem der beiden Unterbäume drehen wir den Wert des boolschen Argumentes durch Negation einfach um.

Ansonsten gilt es einfach, alle Sonderfälle der Reihe nach abzuarbeiten. Lediglich im letzten Fall, bei dem beide Unterbäume nicht leer sind, müssen wir je nach dem Wert des Argumentes `gerade` das Maximum oder Minimum auswählen.

```
public void computeMinMax(boolean gerade) {
    if (left == null && right == null) {
        return;
    } else if (left == null && right != null) {
        right.computeMinMax(!gerade);
        value = right.value;
    } else if (right == null && left != null) {
        left.computeMinMax(!gerade);
        value = left.value;
    } else {
        left.computeMinMax(!gerade);
        right.computeMinMax(!gerade);
        if (gerade) {
            value = Math.max(left.value, right.value);
        } else {
            value = Math.min(left.value, right.value);
        }
    }
}
```

Aufgabe 12-2 *Baumrotation* (4 Punkte) (Abgabeformat: `.txt` oder `.pdf`)
Gegeben ist der folgende Suchbaum:



a) Fügen Sie in den gegeben Suchbaum ein Knoten mit dem Wert 27 ein, und löschen Sie anschließend den Wert 7 aus dem Suchbaum heraus.

   Geben Sie jeweils die vollständig gezeichnete Darstellung des Baumes nach dem Einfügen des Wertes 27 und nach der darauffolgenden Löschung des Wertes 7 an.

   Nutzen Sie dazu die einfachen Algorithmen, welche in der Vorlesung am 12.01.2011 vorgestellt wurden. Dies bedeutet, dass Sie als Ergebnis einen Suchbaum erhalten werden, welcher eventuell nicht ausgeglichen ist.

   Achten Sie zur Ihrer eigenen Kontrolle darauf, dass ihre jeweiligen Ergebnisse die Suchbaumeigenschaft nicht verletzen!

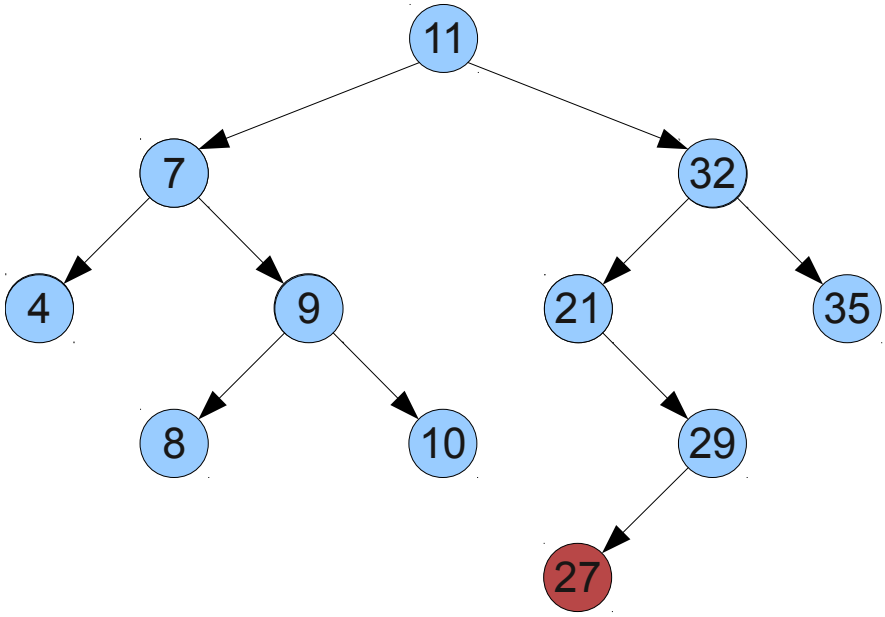
b) In der Vorlesung am 19.01.11 werden Algorithmen zum Einfügen und Löschen in AVL-Bäume vorgestellt. Der oben angegebene Baum ist ein solcher AVL-Baum. (*Hinweis:* Die Folien dazu sind bereits im Foliensatz zur Vorlesung am 12.01.2011 enthalten, welche auf der Homepage der Vorlesung verfügbar ist.)

   Fügen Sie erneut den Knoten 27 in den ursprünglich oben angegebenen Baum ein, und löschen Sie erneut den Wert 7 heraus; verwenden Sie dabei aber nun die Algorithmen für AVL-Bäume. Geben Sie als Lösung erneut beide grafische Darstellungen des Baumes ab, einmal nach Einfügen des Wertes und dann nach dem anschließenden Löschen.

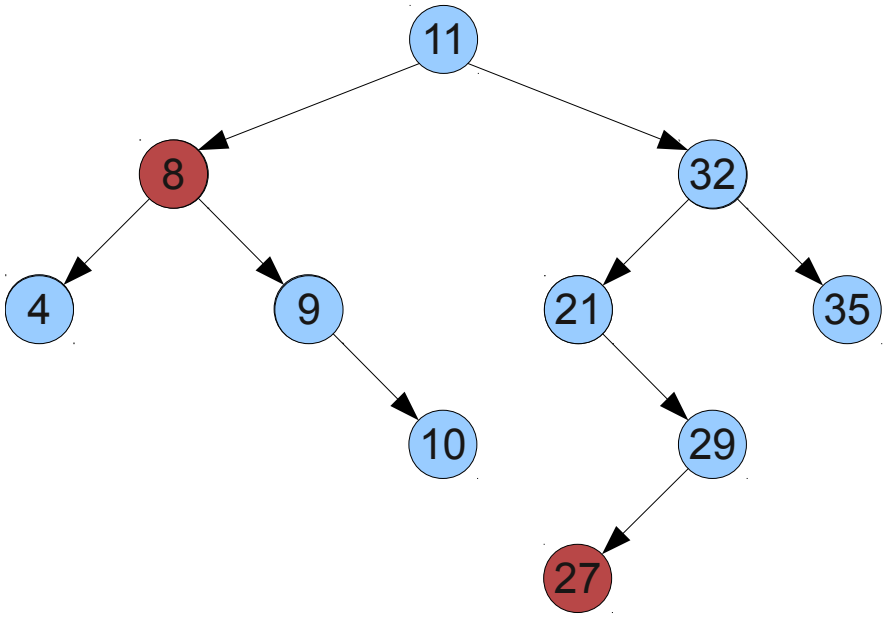
   Achten Sie zur Ihrer eigenen Kontrolle darauf, dass ihre jeweiligen Ergebnisse die AVL-Baumeigenschaft erfüllen!

LÖSUNG:

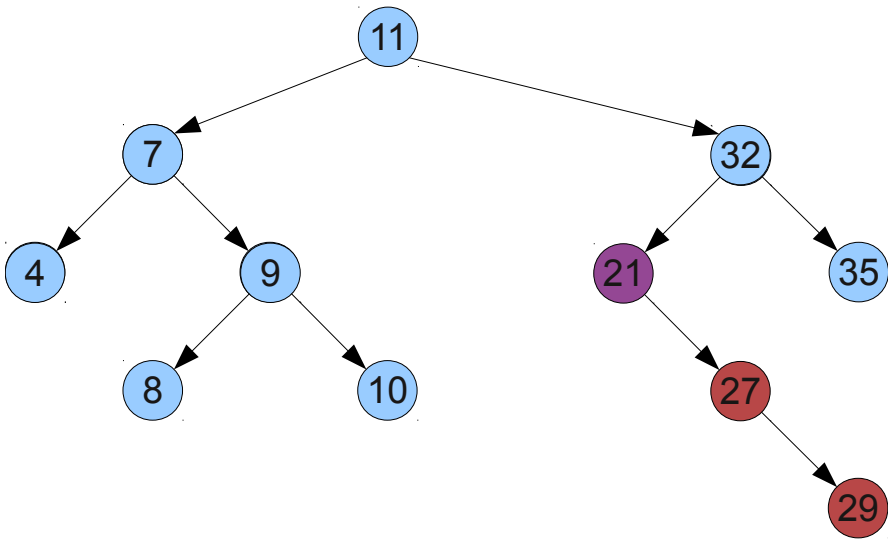
a) Zum Einfügen wandern wir den Baum herab. Dabei wählen wir den rechten Teilbaum, wenn der Wert kleiner als 27 ist und ansonsten den linken Teilbaum. Am Ende fügen wir dann ein frisches Blatt mit der Numer 27 ein.

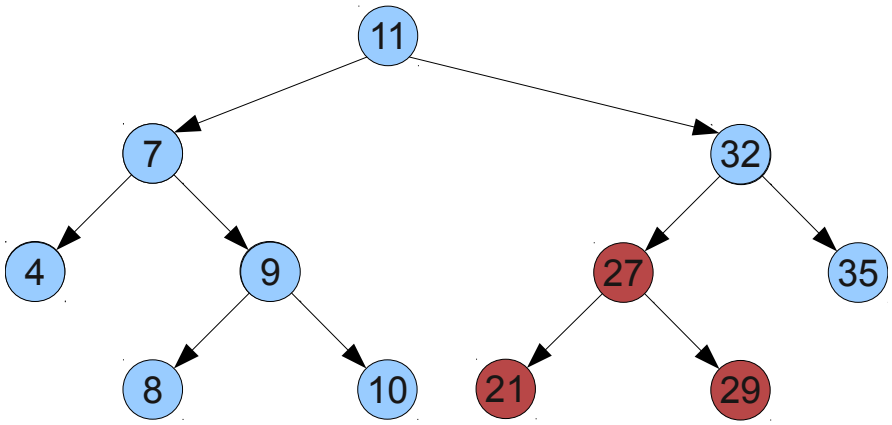


Zum Löschen der 7, müssen wir diese erst finden. Da die 7 zwei Nachfolger hat, müssen wir jetzt das kleinste Element im rechten Teilbaum der 7 finden. Dies ist die 8. Als kleinstes Element im rechten Teilbaum kann sie keinen linken Nachfolger haben, und so setzen wir den ehemaligen linken Teilbaum der 7 als neuen linken Nachfolger der 8. Der rechte Teilbaum wird der ehemalige rechte Teilbaum der 7, nach herauslöschen der 8 aus diesem Teilbaum.

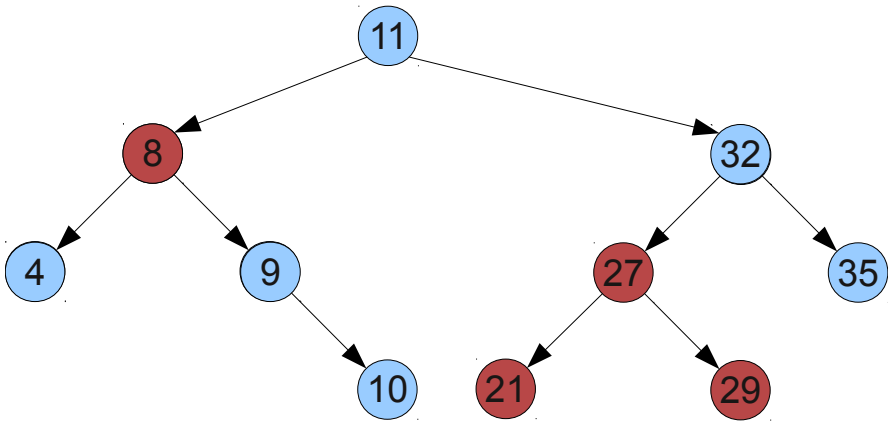


b) Das Einfügen der 27 in einem AVL-Baum beginnt erst einmal genauso wie in der vorherigen Teilaufgabe. Danach prüfen wir aufsteigend vom frisch eingefügten Element, ob die AVL-Eigenschaft verletzt ist. Dies ist im Knoten 21 der Fall. Gemäß Folie 39 der Vorlesung liegt Fall 4 vor, wir müssen also eine Rechts-Links-Doppelrotation durchführen. In zwei Schritten (nach der Rechts-Rotation und am Ende) sieht das so aus:





Das Löschen der 7 verläuft analog zum vorherigen Teil, da durch das Herauslöschen die AVL-Eigenschaft des Baumes nicht verletzt wird. Damit erhalten wir am Ende dieses Bild des Baumes.



Abgabe: Sie können ihre Lösungen bis Montag, den 24.01.2011, 14 Uhr über UniWorX abgeben. Java Dateien, welche nicht kompilieren, werden nicht beachtet!