Institut für Informatik I
Prof. Dr. Heiko Röglin
Tobias Brunsch

universitätbonn

Algorithmen und Berechnungskomplexität I
Wintersemester 2010/11
Abgabe: 27.10.2010 in der Vorlesung

Übungsblatt 2

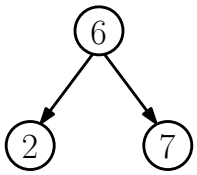
Aufgabe 2.1

4+2 Punkte

Gegeben sei der in der Abbildung dargestellte AVL-Baum.

(a) Fügen Sie die Schlüssel $9, 10, 8, 1, 4, 5, 3$ in der angegebenen Reihenfolge in den Baum ein. Zeichnen Sie den Baum nach jeder Änderung (einfügen, rotieren).

(b) Löschen Sie anschließend die Schlüssel $6, 5, 1$ in der angegebenen Reihenfolge. Zeichnen Sie den Baum nach jeder Änderung (ggf. vertauschen + löschen, rotieren).



Hinweis: Achten Sie darauf, das Rebalancieren mit dem richtigen Knoten zu beginnen. Dieser muss nicht immer die Wurzel des Baumes sein.

Aufgabe 2.2

4+2 Punkte

(a) Seien $n, h \in \mathbb{N}_0$ nichtnegative ganze Zahlen. Beweisen Sie induktiv folgende Aussage: Es gibt einen AVL-Baum mit n Knoten und Höhe h genau dann, wenn gilt

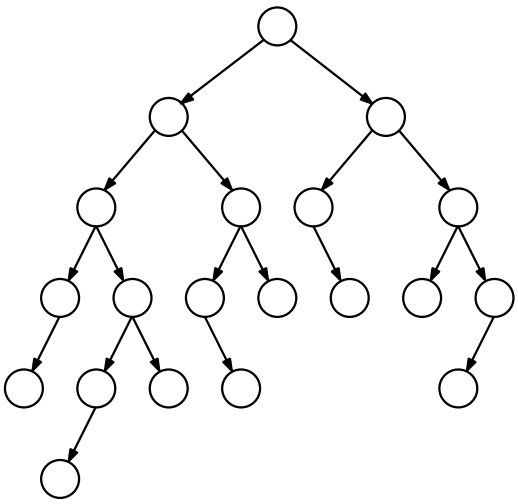
$$F_{h+3} - 1 \leq n \leq 2^{h+1} - 1\,.$$

Dabei ist F_n die n-te Fibonacci-Zahl und es gilt $F_0 = 0$, $F_1 = 1$ sowie $F_n = F_{n-1} + F_{n-2}$ für $n \geq 2$.

Hinweis: Nutzen Sie aus, dass ein binärer Baum genau dann ein AVL-Baum ist, wenn der linke und der rechte Teilbaum der Wurzel AVL-Bäume sind, deren Höhen sich betragsmäßig höchstens um 1 unterscheiden. Sie können außerdem folgende Aussagen als gegeben annehmen:

- Für $h \geq 2$ gilt $F_{h+2} - 1 \leq 2^{h-1}$.
- Ein binärer Baum ist genau dann vollständig, wenn der Balancewert aller seiner Knoten gleich 0 ist.

(b) Gegeben sei der folgende AVL-Baum. Es sollen die Schlüssel $1, \ldots, 20$ mit AVL-Einfügeoperationen in einen zu Beginn leeren Baum eingefügt werden. Geben Sie eine Reihenfolge dieser Schlüssel an, bei der der unten stehende Baum erzeugt wird.



Aufgabe 2.3

3+3 Punkte

Wir betrachten ein aufsteigend sortiertes Feld A mit n Einträgen.

(a) Es sollen die Einträge von A als Schlüssel in einen zu Beginn leeren binären Suchbaum eingefügt werden. Das Einfügen erfolgt über die Standardmethode für binäre Suchbäume (ohne jegliche Balancierungen) und benötigt $O(h)$ Operationen, wenn h die aktuelle Höhe des Baumes ist. Geben Sie einen Algorithmus an, der alle Einträge von A einfügt und $O(n \log n)$ Operationen benötigt.

Hinweis: Da Sie die Einfügeoperation nicht beeinflussen können, ist nur wichtig, in welcher Reihenfolge die einzelnen Schlüssel eingefügt werden.

(b) Geben Sie einen Algorithmus mit Laufzeit $O(n)$ an, der einen AVL-Baum erstellt, der genau die Einträge von A als Schlüssel enthält. Sie können dabei beliebige Operationen verwenden und die Baumstruktur selbst verwalten. Zwischendurch muss der Baum die AVL-Eigenschaft nicht besitzen.

Aufgabe 2.4

6 Punkte

In einem Algorithmus soll ein Integer-Feld der Länge n für eine sehr große natürliche Zahl n verwendet werden. Dieses sei fortlaufend in einem riesigen Speicher angelegt. Von dem Algorithmus sei bekannt, dass er nur positive ganze Zahlen in das Feld schreibt. Daher assoziieren wir einen nicht gesetzten Eintrag mit der Zahl 0. Weiterhin sei bekannt, dass die Speicherzugriffe sehr langsam sind und dass der Algorithmus nur einen Bruchteil des Feldes nutzen wird. Die Indizes, auf die der Algorithmus zugreifen wird, sind jedoch nicht a priori bekannt und können beliebig über das gesamte Feld verteilt sein. Wir wollen uns die aufwendige Initialisierung des Feldes, in dem zu Beginn beliebiger Unsinn („Bitmüll“) steht, mit Nullen ersparen. Gesucht ist eine Datenstruktur zur Verwaltung des Feldes, die folgende Methoden bereitstellt.

- `Write(v, i)` schreibt den Wert v an Position i des Feldes und markiert Eintrag i als gesetzt.
- `Delete(i)` markiert Eintrag i als ungesetzt.
- `Read(i)` liest Eintrag i aus. Ist dieser nicht als gesetzt markiert (also Bitmüll), wird 0 zurückgeliefert, ansonsten Eintrag i.

Beschreiben Sie den Aufbau und die Funktionsweise einer Datenstruktur, die obige Methoden bereitstellt und bei der jede Methode nur $O(1)$ Zeit benötigt. Quelltext ist nicht nötig. Eine schematische Darstellung mit Erklärungen genügt. Sie können dabei annehmen,

- dass das Lesen und Schreiben einer Speicheradresse in Zeit $O(1)$ möglich ist,
- dass in jeder Einheit des Speichers eine beliebig große Zahl gespeichert werden kann,
- dass die Anzahl m der Aufrufe der `Write`-methode a priori bekannt ist und
- dass genügend freier Speicher vorhanden ist.

Hinweis: : Benutzen Sie eine Extradatenstruktur, in der steht, welche Einträge des Feldes als gesetzt markiert sind. Dort werden sozusagen *Zeugen* der Echtheit verwaltet. Benutzen Sie außerdem eine Extrainformation für jeden Feldeintrag, die ggf. auf den Zeugen verweist.