Algorithmen und Datenstrukturen
Übungszettel 5

Martin Avanzini <martin.avanzini@uibk.ac.at>
Thomas Bauereiß <thomas.bauereiss@uibk.ac.at>
Herbert Jordan <herbert@dps.uibk.ac.at>
René Thiemann <rene.thiemann@uibk.ac.at>

12. April, zur Besprechung am 3. Mai

Aufgabe 1) **Amortisierte Kosten** Betrachten Sie das folgende Programm, das einen binären Zähler beliebiger Länge realisiert.

```
1  public class Counter {
2     byte[] bits;
3
4     public Counter(int n) {
5        bits = new byte[n];
6     }
7
8     public void tick() {
9        int j = bits.length - 1;
10       while (j >= 0) {
11           if (bits[j] == 1) {
12               bits[j] = 0;
13               j--;
14           } else {
15               break;
16           }
17       }
18       if (j >= 0) {
19           bits[j] = 1;
20       }
21    }
22 }
```

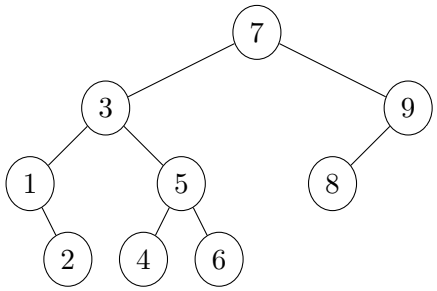
Geben Sie eine geeignete Guthaben-Funktion G an und zeigen Sie, dass die Methode `tick()` amortisierte Kosten von $\mathcal{O}(1)$ hat. Nehmen Sie dabei an, dass die Ausführung der Zeilen-Blöcke 9, 11-16 und 18-20 jeweils 1 Zeiteinheit kostet.

Aufgabe 2) **Einfügen und Löschen in AVL Bäume**

a) Führen Sie die Werte 1,7,5,9,6,8,3,4 und 2 in der gegebenen Reihenfolge in einen anfangs leeren AVL Baum ein. Stellen Sie den Baum nach jedem Schritt dar, geben Sie die Höhendifferenz

in jeden Knoten an und illustrieren Sie notwendige Rotationen.

b) Gegeben - der folgende AVL Baum:



Löschen Sie den Wert 8 aus dem Baum und zeigen Sie die notwendigen Balancierungsschritte.

c) Kann der `hdiff` Wert eines Knotens aus den `hdiff` Werten der Kind-Knoten ermittelt werden? Was ist der `hdiff` eines inneren Knotens dessen Kind-Knoten jeweils den Wert 0 aufweisen?

d) Auf Folie 164 der Vorlesungsunterlagen fehlt der `hdiff` Wert für die Knoten n und p. Beschreiben Sie, wie diese Werte lediglich aus den vorhergehenden `hdiff` Werten der Knoten n, p und r ermittelt werden können.

Aufgabe 3) AVL Implementierung Implementieren Sie die folgenden Operationen auf AVL Bäumen:

- Suche eines Elements (`contains`)
- Einfügen eines Elements (`insert`) inklusive der notwendigen Rotationen (`rotateLeft` und `rotateRight`) und Balancierungsschritte (`balGrow`).

Achten Sie bei Ihrer Implementierung auf die korrekte Erhaltung der AVL Bedingungen. Ein Grundgerüst für Ihre Implementierung findet sich in den beiliegenden Java Sources (AVLTree.java). Zusätzlich finden Sie einen `TestCase` welcher es Ihnen erlaubt, Ihre Implementierung zu überprüfen.
Hinweis: Falls Sie das `Comparator`-Interface im Grundgerüst nicht verwenden wollen, gibt es auch eine Version, in der die Einträge Zahlen sind (AVLTree2.java).