Effiziente funktionale Datenstrukturen
Übungen

Alexander Nutz

3. Dezember 2007

1. Betrachten sie folgenden Programmcode und zeichnen sie die Bäume, die in den Variablen t1 bis t4 referenziert werden, ausgehend von dem im Vortrag vorgestellten Code für einen Binärsuchbaum.

```
import Binsearchtree
t1 = insert 5 E
t2 = insert 7 (insert 13 t1)
t3 = insert 23 t2
t4 = insert 23 t1
```

   Kontrollieren sie ihre Zeichnungen, indem sie das Programm laden und an der Kommandozeile die Variablen abfragen. Was stellen sie fest bezüglich der Persistenz?

   (Sie müssen die Datei binsearchtree.hs von der Website in das entsprechende Verzeichnis kopieren.)

2. Die *balance*-Funktion führt im vorgestellten Algorithmus einige unnötige Überprüfungen durch. Zum Beispiel ist es unnötig, wenn die *ins*-Funktion einen Knoten am linken Kind-Knoten einfügt, auf eine Rot-Rot-Kombination beim rechten Kind zu testen.

   (a) Teilen sie *balance* in zwei Funktionen *lbalance* und *rbalance* auf, die jeweils auf Verletzung der Invarianten am rechten oder linken Kind-Knoten testen. Ersetzen sie die *balance*- Aufrufe in *ins* mit der entsprechenden neuen Funktion.

   (b) Außerdem ist eine der verbleibenden Überprüfungen der Enkel-Knoten unnötig. Schreiben sie *ins* so um, dass es nie die Farbe von Knoten kontrolliert, die sich nicht auf dem Suchpfad befinden.

3. Erweitern sie die vorgestellte Queue zu einer double-ended Queue, einer sogenannten Deque, die Lese-und Schreiboperationen an beiden Enden der Queue erlaubt.

   - Die Listen fs und rs müssen symmetrisch behandelt werden.
   - Keine von beiden darf bei mehr als zwei Elementen im Deque leer werden.
   - Wenn eine der Listen leer wird, wird die andere halbiert und eine der Hälften invertiert.

   (a) Implementieren sie eine solche Deque.

   (b) Zeigen sie, dass alle Operationen in amortisiert $O(1)$ laufen. Nutzen sie die Potentialfunktion $\Phi(fs, rs) = abs(|fs| - |rs|)$, wobei $abs(x)$ den Absolutbetrag einer Zahl x darstellt.