Algorithmen und Programmieren III, WS 1999/2000 — 6. Übungsblatt

Abgabe bis Dienstag, 14. Dezember 1999

---

31. (0 Punkte) *Rot-Schwarz-Bäume* sind binäre Bäume mit folgenden Eigenschaften:[1]

    (a) Jeder innere Knoten hat zwei Kinder.
    (b) Jeder Knoten ist entweder als *rot* oder als *schwarz* gekennzeichnet.
    (c) Die Wurzel und alle Blätter sind schwarz.
    (d) Die Kinder eines roten Knotens sind schwarz.
    (e) Ein schwarzer Knoten kann höchstens ein rotes Kind haben.
    (f) Alle Wege von der Wurzel zu den Blättern enthalten gleich viele schwarzen Knoten.

    Zeichnen Sie Rot-Schwarz-Bäume mit 5, 7 und 12 Blättern. Zeigen Sie, dass man aus jedem Rot-Schwarz-Baum einen 2-3-Baum machen kann, und umgekehrt.

32. (5 Punkte) Erweitern Sie die Algorithmen zum Suchen, Einfügen und Löschen in binären Suchbäumen so, dass man jederzeit auch das i-größte Element finden kann, und dass alle Operationen Zeit proportional zur Höhe des Baumes benötigen.

33. (0 Punkte) Gegeben sind n Güter (Sand, Gold, Salz, usw.). Von jedem Gut ist eine gewisse Menge x_i vorhanden; weiters ist der Wert w_i (pro Gewichtseinheit) bekannt. Welche Güter soll man in welcher Menge auswählen, wenn man insgesamt nur ein Gesamtgewicht von b Einheiten nehmen kann und den größten Wert erzielen möchte? (Welche Güter soll man zum Beispiel bei einem Feuer in Sicherheit bringen?)
    Wie würden Sie dieses Problem (unter Zuhilfenahme binärer Suchbäume) lösen, wenn man für verschiedene Werte von b jeweils schnell eine Antwort bekommen möchte, und wenn sich die Menge der Güter möglicherweise ändert?

34. (5 Punkte) Schreiben Sie ein Programm, das einen (nicht zu großen) binären Baum zweidimensional in übersichtlicher und schön lesbarer Form ausdruckt. Zwei Vorschläge:

```
      20   ----
         23  -
            30
   40  -------
      45
50  ----------
      64  -
         65
   70  ----
```

```
          50
      /----↥---------\
     40              70
 /-------↥\       /---↥
20         45    64
↥\              /-↥\
  23           60   65
  ↥-\
     30
```

35. (5 Punkte) Implementieren Sie für binäre Suchbäume eine Methode, die einen *Iterator* mit den Methoden `next()` und `hasNext()` erzeugt. (Sie dürfen annehmen, dass die Knoten Mutterzeiger wie in Aufgabe 26 haben.) Verwenden Sie dabei *nicht* das Verfahren aus der Vorlesung, das eine Kopie der Daten in einem linearen Feld anlegt, oder eine andere Methode, die mehr als konstant viel zusätzlichen Speicher benötigt.

36. (a) (0 Punkte) In einem binären Suchbaum, der ganze Zahlen speichert, sollen die Zahlen x, die im Intervall $\{\, x \mid a \le x \le b \,\}$ liegen, ausgegeben werden. Schreiben Sie ein Programm, das diese Aufgabe in $O(h+k)$ Zeit erledigt, wobei h die Höhe des Baumes und k die Länge der Ausgabe ist.
    (b) (0 Punkte) Wie kann man die in den Knoten des Baumes gespeicherte Informationen so erweitern, dass man die *Summe* der Zahlen x im Intervall $\{\, x \mid a \le x \le b \,\}$ in $O(h)$ Zeit bestimmen kann (und trotzdem noch in $O(h)$ Zeit einfügen und entfernen kann)?

---

[1] Es gibt verschiedene andere Versionen von Rot-Schwarz-Bäumen, bei denen zum Beispiel auch die inneren Knoten Werte enthalten (im Gegensatz zu 2-3-Bäumen).
Eine Implementierung von Rot-Schwarz-Bäumen kann man zum Beispiel in der Klasse `TreeMap` im Paket `java.util` finden, siehe `http://www.inf.fu-berlin.de/~rote/Lere/alp3/TreeMap.java`.