Algorithmen und Programmierung III

WS 2002/2003 — Übung 9 — HSchweppe

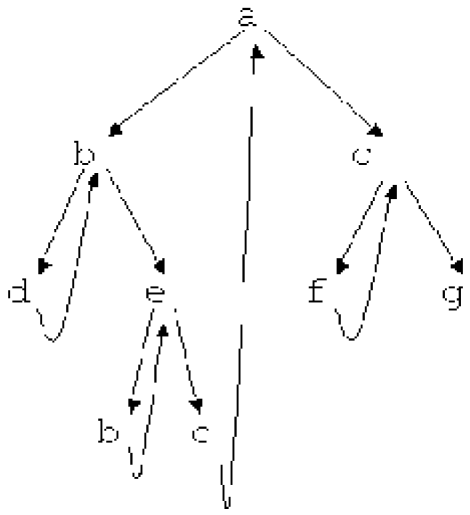
Ausgabe: 12.12. 2002
Abgabe: 7. 1. 2003 15.00

Aufgabe 9.1 (6 P)
Gegeben ist ein binärer, verkettet implementierter Baum:

```
class BinTree1 implements BinTree{
  Object val;
  BinTree1 left, right; ....}
```

mit den üblichen Operationen . Gesucht ist eine Methode `BinTree1 clone()`, die einen Baum "tief" kopiert (*deep copy*). eine tiefe Kopie erzeugt einen neuen Baum, der eine Kopie des ursprünglichen ist. (Frage am Rande: wie würden Sie im Gegensatz dazu eine "flache" Kopie definieren?)

Aufgabe 9.2 (10 P)
Die iterative Traversierung eines vollen Binärbaums (jeder Knoten hat zwei Kinder oder ist Blatt) kann durch geeignete Zeiger beschleunigt werden. Das folgende Beispiel zeigt einen sogenannten gefädelten Operatorbaum (*threaded tree*). Der "Faden" entspricht hier der Inorder-Traversierung des Baums. Gesucht ist eine Klasse `class ThreadedTree` , die mindestens Methode `public Iterator inOrderIterator()` und `void expand(ThreadedTree left, ThreadedTree right)` besitzt. Diese macht einen Blattknoten zu einem inneren Knoten mit zwei Kindern mit korrekter Fderlung.



Aufgabe 9.3 (9 P)

a) (2 P) Zeichnen Sie binäre Suchbäume der Höhen 2, 3, 4, 5 und 6 für die Schlüsselmenge {5,7,8,11,17,19,35}

b) (1)Welche minimale bzw. maximale Anzahl von Knoten enthält ein binärer Suchbaum der Höhe h?

c) (2 P) Angenommen wir haben in einem binären Suchbaum Zahlen zwischen 1 und 1000 gespeichert und wollen nach der Zahl 363 suchen. Überprüfen Sie für jede der Folgen 1 - 4, ob die Elemente Werte von Knoten sein können, die bei der Suche traversiert wurden.

1) 1000, 200, 899, 248, 898, 273, 361, 363
2) 943,217,335,712,289,397,348,363
3) 1001, 199, 935, 240, 936, 273,363
4) 2, 398, 388, 211, 276, 385, 379, 278, 363

d) (2 P) Sei T ein binärer Suchbaum, x ein Blattknoten, y sein Vaterknoten. Der Schlüssel eines Knotens k wird mit *val(k)* bezeichnet. Zeigen Sie: *val(y)* ist entweder der kleinste Schlüssel im Baum größer als *val(x)* oder der größte Schlüssel in dem Baum, der kleiner als *val(x)* ist.

e) (2 P) S. Tupid glaubt eine wichtige Eigenschaft binärer Suchbäume entdeckt zu haben. Angenommen, die Suche nach dem Schlüssel k endet erfolgreich in einem Blatt. Folgende 3 Knotenmengen lassen sich dann definieren: A als Menge der Knoten, die links vom Suchpfad liegen, B als Menge der Knoten auf dem Suchpfad zu k und C als Menge der Knoten rechts vom Suchpfad. Die Behauptung von Tupid lautet: für alle $a \in A, b \in B, c \in C$ gilt: $a \leq b \leq c$. Finden Sie ein möglichst kleines Gegenbeispiel für diese Behauptung.

Aufgabe 9.4 (7 P)

a) (7 P) Ein zufällig erzeugter binärer Suchbaum mit n Knoten hat eine Höhe von ca. $C*\log n$. Sie sollen diese Aussage experimentell bestätigen. Erzeugen Sie große binäre Suchbäume mit n Elementen und bestimmen Sie deren Höhe. Um die Suchbäume aufzubauen , erzeugen Sie Zufallszahlen zwischen `0` und `MAXINT` und bestimmen Sie die Konstante C näherungsweise.

Dazu müssen Sie eine Klasse für binäre Suchbäume mit geeigneten Methoden implementieren Verwenden Sie `Comparable` als Typ für die Knotenwerte. Alternativ zu Java kann die Untersuchung mit einer Haskell-Implementierung durchgeführt werden.

b) (3 Bonuspunkte) Vergleichen Sie die Suche im binären Suchbaum mit der Suche durch Einschachtelung ("binäre Suche") in einem Vektor.

32 Punkte (+ 3 Bonuspunkte)

Schöne Weihnachten und guten Rutsch.