

**Norman May**

B6, 29, Raum C0.05  
68131 Mannheim  
Telefon: (0621) 181-2517  
Email: norman@pi3.informatik.uni-mannheim.de

**Matthias Brantner**

B6, 29, Raum C0.05  
68131 Mannheim  
Telefon: (0621) 181-2517  
Email: msb@pi3.informatik.uni-mannheim.de

Algorithmen und Datenstrukturen  
Wintersemester 2004/05

9. Übungsblatt  
22. Dezember 2004

**Aufgabe 1****15 Punkte**

Implementieren Sie Rot-Schwarz-Bäume ohne Verwendung des Parent-Zeigers. Wenn Ihnen diese Implementierung zu schwer ist, geben Sie eine Implementierung mit Parent-Zeiger an (- 33%).

Befolgen Sie bei der Implementierung folgende Schnittstelle — Ihr Programm wird gegen diese Schnittstelle getestet. Weitere Hilfsfunktionen können hilfreich sein.

```
public class RBTreeNode {  
  
    /* liefere den Wert, der in dem Knoten gespeichert ist */  
    public Comparable getValue() { }  
  
    /* true, wenn roter Knoten, sonst false */  
    public boolean isRed() { }  
  
    /* liefere linken Kindknoten */  
    public RBTreeNode getLeft() { }  
  
    /* liefere rechten Kindknoten */  
    public RBTreeNode getRight() { }  
  
    /* Textrepräsentation des Knotenwerts und des Rot-Schwarzerts (e.g. 5 (r)) */  
    public String toString() { }  
  
}  
  
public class RedBlackTree {  
  
    /* liefere den NIL-Wert */  
    public static RBTreeNode getNil() { }  
  
    /* liefere den Wurzelknoten des Baums */  
    public RBTreeNode getRoot() { }  
  
}
```

```

/* erzeuge einen leeren red-black tree */
public RedBlackTree() { }

/* loesche Knoten z aus dem Baum */
public RBTreeNode delete(RBTreeNode z) { }

/* fuege value in den Baum ein */
public RBTreeNode insert(Comparable value) { }

/* suche nach einem Knotem mit Wert value */
public RBTreeNode lookup(Comparable value) { }
}

```

---

**Aufgabe 2**
**4 Punkte**


---

Zeichnen Sie den Rekursionsbaum von *MERGE-SORT* für ein Feld von 16 Elementen. Erklären Sie, warum *Memoization* zur Steigerung der Effizienz eines guten Teile-und-Herrsche-Algorithmus wie *MERGE-SORT* unwirksam ist.

---

**Aufgabe 3**
**8 Punkte**


---

Die Fibonacci-Zahlen sind definiert durch folgende Rekurrenz:

$$\begin{aligned}
 F_0 &= 0 \\
 F_1 &= 1 \\
 F_i &= F_{i-1} + F_{i-2}
 \end{aligned}$$

**Aufgabe 3 a)**
**2 Punkte**

Geben Sie am Beispiel der Berechnung von  $F_{10}$  an, wieviele Teilberechnungen in diesem Beispiel insgesamt redundant ausgeführt werden?

**Aufgabe 3 b)**
**3 Punkte**

Implementieren Sie die Fibonacci-Zahlen und verwenden Sie *Memoization*, um wiederholte Berechnungen von  $F_i$  zu vermeiden.

**Aufgabe 3 c)**
**3 Punkte**

Implementieren Sie die Berechnung der Fibonacci-Zahlen mit dynamischem Programmieren.

---

**Aufgabe 4**
**4 Punkte**


---

Sei  $\langle A_1, A_2, A_3, A_4, A_5, A_6 \rangle$  eine Matrizen­sequenz und  $\langle 5, 10, 3, 12, 5, 50, 6 \rangle$  die dazugehörige Dimensionensequenz. Bestimmen Sie ein optimal geklammertes Produkt der gegebenen Matrizen­sequenz.

---

Aufgabe 5

3 Punkte

---

Sei das in der Vorlesung vorgestellte “Aktivitätsauswahlproblem” gegeben. Ist dieses Problem ein Matroid? Beweisen Sie Ihre Aussage.

---

Aufgabe 6

5 Punkte

---

Die Schlüssel 23, 25, 15, 7, 5, 8, 9, 10, 11, 12, 13 sollen in einen B-Baum mit  $t = 2$  eingefügt werden. Geben Sie den B-Baum an, der durch das Einfügen der Schlüssel in einen leeren B-Baum entsteht.

---

Aufgabe 7

9 Punkte

---

Zeigen oder widerlegen Sie folgende Aussagen.

Aufgabe 7 a)

3 Punkte

$$f(n) = \Theta(f(n/2))$$

Aufgabe 7 b)

3 Punkte

$$f(n) = O(g(n)) \Rightarrow g(n) = \Omega(f(n))$$

Aufgabe 7 c)

3 Punkte

$$o(f(n)) + \omega(f(n)) = \Theta(f(n))$$