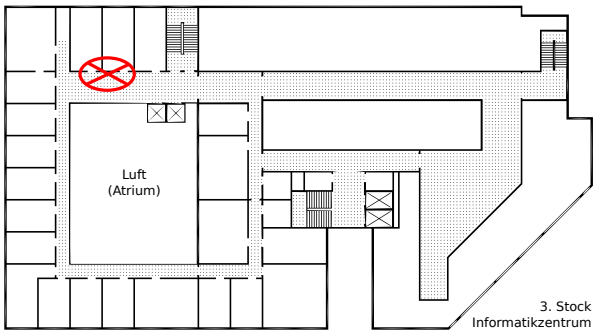
Abteilung Algorithmik **Winter 2013/14**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Stephan Friedrichs

Algorithmen und Datenstrukturen
Übung 5 vom 08. 01. 2014

Abgabe der Lösungen bis zum Mittwoch, den 22. 01. 2014 um 13:00 im Hausaufgabenrückgabeschrank.

Bitte die Blätter zusammenheften und vorne deutlich mit eigenem Namen, Matrikel- und Gruppennummer, sowie Studiengang versehen!



Aufgabe 1 (Klausurvorbereitung): Gib deinen Namen (Format: Nachname, Vorname), Matrikelnummer, Studiengang und angestrebten Abschluss *leserlich* an. Diese Angaben brauchen wir für die Weiterleitung der Klausurergebnisse, also gebt euch bitte Mühe ;-) **(2 Punkte)**

Aufgabe 2 (Mergesort): Sortiere die Sequenz $(18, 11, 12, 3, 14, 11, 4, 5)$ mit Hilfe von Mergesort. Gib die Zwischenschritte in geeigneter Form an. **(13 Punkte)**

Aufgabe 3 (Mastertheorem): Bestimme mit Hilfe des Mastertheorems aus der Vorlesung das asymptotische Wachstum der folgenden Rekursionen. Gib jeweils die Werte aller im Mastertheorem auftretenden Parameter an.

a) $U(n) = 13 \cdot U\left(\frac{n}{4}\right) + 27n^3 + 27 \cdot U\left(\frac{n}{6}\right) + 13n$

b) $V(n) = 3125 \cdot V\left(\frac{n}{5}\right) + 17n^3$

c) $T(n) = 63 \cdot T\left(\frac{n}{12}\right) + 36 \cdot T\left(\frac{n}{8}\right) + 51n^2 + 3$

(5+5+5 Punkte)

Aufgabe 4 (Sortieren mit Suchbäumen): Wir betrachten nun einen Sortieralgorithmus, der binäre Suchbäume verwendet. Algorithmus 1, TREESORT, fügt dazu ein (unsortiertes) Feld elementweise in einen binären Suchbaum T ein. Anschließend wird T in *inorder* traversiert, das heißt, die Elemente in aufsteigender Reihenfolge durchlaufen und ausgegeben.

```
1: function INORDER(T)
2:     if T = TREE() then                                  ▷ Leerer (Teil-)Baum
3:         return ( )                                      ▷ Leeres Feld zurückgeben
4:     else if T = TREE(L, a, R) then      ▷ a ist Wurzel, L linkes und R rechts Kind
5:         return INORDER(L), a, INORDER(R)
6:     end if
7: end function

8: function TREESORT(T : Tree, a_1, ..., a_n)
9:     for i ← 1, ..., n do                                ▷ Elemente in Baum einfügen
10:        T ← INSERT(T, a_i)
11:    end for
12:    return INORDER(T)
13: end function
```

Algorithmus 1: Sortieren eines Feldes mit Hilfe von Bäumen

Wir wollen dabei verschiedene Typen von Suchbäumen T betrachten und ihre asymptotischen Laufzeiten vergleichen. Benutzt werden darf dabei, dass die Laufzeit von INORDER() $O(n)$ ist.

a) Berechne TREESORT$(T, 8, 10, 7, 5, 3, 1)$ für einen anfangs leeren, (unbalancierten) binären Suchbaum T. Abgabe: Der Baum T nach Einfügen aller Elemente, sowie das zurückgegebene Feld.

b) Welche asymptotische Laufzeit (in O-Notation) hat TREESORT, wenn T ein unbalancierter, binärer Suchbaum ist? Begründe deine Antwort!

c) Berechne TREESORT$(T, 8, 10, 7, 5, 3, 1)$ und verwende dazu für T einen zu Beginn leeren AVL-Baum. Abgabe: Der Baum T nach jeder Einfügeoperation sowie vor und nach jeder Rotation, sowie das am Ende zurückgegebene Feld.

d) Welche asymptotische Laufzeit (in O-Notation) hat TREESORT, wenn T ein AVL-Baum ist? Begründe deine Antwort!

(3+4+4+4 Punkte)

Aufgabe 5 (AVL-Bäume): Wir möchten aus einem bereits sortierten Feld ohne Duplikate effizient einen AVL-Baum erzeugen. Betrachte dazu Algorithmus 2: BUILDAVLTREE nimmt ein sortiertes Feld von n Elementen und erzeugt daraus einen AVL-Baum.

a) Zeichne den Baum T, den BUILDAVLTREE$(1, 2, 3, 4, 5, 6, 7)$ erzeugt. Überprüfe, ob T ein AVL-Baum ist.

b) Sei $n = 2^k - 1$ mit $0 \leq k \in \mathbb{N}$. Zeige: Für $a_1 < \cdots < a_n$ liefert Algorithmus 2, also BUILDAVLTREE$(a_1, \ldots, a_n)$, immer einen AVL-Baum T minimaler Höhe. Dazu ist zu zeigen, dass

   (i) T ein binärer Suchbaum ist,

   (ii) T die AVL-Eigenschaft erfüllt,

   (iii) T minimale Höhe hat.

```
1: function BUILDAVLTREE(a_1, ..., a_n)
2:     if n = 0 then
3:         return TREE()                                  ▷ Leeren Baum zurückgeben
4:     else
5:         m ← ⌈n/2⌉                                      ▷ Index des mittleren Elements
6:         L ← BUILDAVLTREE(a_1, ..., a_{m-1})
7:         R ← BUILDAVLTREE(a_{m+1}, ..., a_n)
8:         return TREE(L, a_m, R)          ▷ a_m ist Wurzel, L linkes und R rechts Kind
9:     end if
10: end function
```

Algorithmus 2: Erzeugung eines AVL-Baumes aus sortiertem Feld

(Hinweis: Vollständige Induktion über k mit Induktionsbeginn bei $k = 0$. Außerdem ist $\lceil \frac{2^k-1}{2} \rceil = 2^{k-1}$.)

c) Bestimme die Laufzeit von Algorithmus 2 mit Hilfe des Mastertheorems. TREE() hat konstante Laufzeit. Wie ist die Laufzeit, wenn alle Elemente der Reihe nach mit INSERT in einen AVL-Baum eingefügt werden?

(4+7+4 Punkte)