Institut für Computergraphik und Algorithmen

Abteilung für Algorithmen und Datenstrukturen

TECHNISCHE UNIVERSITÄT WIEN

Prof. Petra Mutzel
Günther Raidl
Ivana Ljubić
René Weiskircher

Sommersemester 2001

Klausur zur Vorlesung
Algorithmen und Datenstrukturen 1
22. Juni 2001

a.) Machen Sie bitte die folgenden Angaben in deutlicher Blockschrift:

Name: _____ Vorname: _____

Matrikelnummer: _____ Studienkennzahl: _____

b.) Klausur soll gewertet werden für (nur ein Kreuz!):

- ○ VO-Prüfung **und** LU-Test Algorithmen und Datenstrukturen 1 (*default*)
- ○ VO-Prüfung Algorithmen und Datenstrukturen 1
- ○ LU-Test Algorithmen und Datenstrukturen 1
- ○ VO-Prüfung Rekursive Prozeduren und flexible Datenstrukturen

c.) Legen Sie während der Klausur Ihren Studentenausweis vor sich auf das Pult.

d.) Schreiben Sie die Lösungen direkt auf das jeweilige Aufgabenblatt. Wenn Ihnen das Papier ausgeht, bitten Sie die Aufsicht um Nachschub. Es ist nicht erlaubt eigenes Papier zu verwenden!

e.) Denken Sie daran, dass keinerlei Hilfsmittel erlaubt sind – weder Taschenrechner, irgendwelche Unterlagen, Handys,...

VOR DER ABGABE AUSZUFÜLLEN:

f.) Geben Sie bitte die Anzahl der zusätzlich abgegebenen Blätter an: _____

g.) Kreuzen Sie bitte die von Ihnen bearbeiteten Aufgaben in der ersten Zeile der Tabelle an:

Aufgabe	A 1	A 2	A 3	A 4	A 5		Note
bearbeitet						—	_____
maximale Punktzahl	10	10	10	15	15	60	_____
erreichte Punktzahl							

Viel Erfolg!

Aufgabe 1: O-Notation **(10 Punkte)**

Sei

$$f(n) = \begin{cases} \frac{\log 10n}{10n} + 5n^3, & 2^n < 10^8 \\ \frac{1}{5n^3} + \frac{1}{25n^4} + \frac{1}{125n^5}, & 2^n \geq 10^8 \end{cases}$$

Füllen Sie die folgende Tabelle aus:

$f(n)$ ist	$O(.)$	$\Omega(.)$	$\Theta(.)$
n^3			
$\frac{1}{n^3}$			
$\frac{1}{n^5}$			
$\frac{\log n}{n}$			
$\log n$			

Aufgabe 2: Sortierverfahren **(10 Punkte)**

Gegeben sind zwei Algorithmen:

```
(0) Algorithmus Sort1(A[1..n])
(1) /* Input: Folge A von n Zahlen. */
(2) /* Output: Sortierte Folge A. */
(3)     Für j = 1, 2, ..., n − 1 {
(4)         mpos = j;
(5)         Für i = j + 1, ..., n   {
(6)             Falls (A[i].key < A[mpos].key) mpos = i; };
(7)         t = A[mpos];
(8)         A[mpos] = A[j];
(9)         A[j] = t; }
```

```
(0) Algorithmus Sort2(A[1..n])
(1) /* Input: Folge A von n Zahlen. */
(2) /* Output: Sortierte Folge A. */
(3)     Für j = 1, 2, ..., n − 1{
(4)         mpos = j;
(5)         Für i = j + 1, ..., n   {
(6)             Falls (A[i].key < A[mpos].key)  mpos = i; };
(7)         Falls (mpos! = i)   {
(8)             t = A[mpos];
(9)             Für k = mpos, ..., j + 1   {
(10)                A[k] = A[k − 1]; }
(11)            A[j] = t; } }
```

a.) Was für ein Sortierverfahren repräsentiert jeder Algorithmus?

b.) Sind diese Algorithmen stabil? Begründen Sie Ihre Antwort.

c.) Berechnen Sie (in Θ-Notation) wieviele Schlüsselvergleiche im besten und im schlimmsten Fall jeder Algorithmus benötigt.

Aufgabe 3: AVL Bäume **(10 Punkte)**

a.) Konstruieren Sie den balancierten AVL-Baum, der sich aus dem leeren Baum ergibt, wenn man nacheinander die folgenden Elemente einfügt:

$$4, 2, 0, 15, 3, 9, 12, 6, 10, 13.$$

Zeichnen Sie den Baum nach jeder Einfügung.

b.) Löschen Sie dann das Element 4, und geben Sie die erreichte Konfiguration an.

Aufgabe 4: Was macht der folgende Algorithmus? **(15 Punkte)**

```
(0) int WhatAmI (v)
(1) /* Input: Knoten v in einem binären Baum. */
(2) /* Output: ??? */
(3)     int S=0;
(4)     Falls (v.left ≠ NIL) AND (v.right == NIL)
(5)         S=1+WhatAmI(v.left);
(6)     Sonst Falls (v.left == NIL) AND (v.right ≠ NIL)
(7)         S=1+WhatAmI(v.right);
(8)     Sonst Falls (v.left ≠ NIL) AND (v.right ≠ NIL)
(9)         S=WhatAmI(v.left)+ WhatAmI(v.right);
(10)    return S;
```

a.) Welche Bedeutung hat der Rückgabewert von WhatAmI(r), wenn r die Wurzel eines nicht leeren Baumes ist?

b.) Was macht der Algorithmus wenn der Baum leer ist? Wie können Sie diesen Pseudo-Code verbessern?

c.) Zeichnen Sie einen Baum mit 15 Knoten für den der Algorithmus als Rückgabewert 0 liefert.

d.) Zeichnen Sie einen Baum mit 15 Knoten für den der Algorithmus als Rückgabewert die größtmögliche Zahl zurück liefert. Wie groß ist diese Zahl?

Aufgabe 5: Binäre Bäume **(15 Punkte)**

a.) Ein binärer Baum ist gegeben. Die Schlüssel des Baums sind ohne bekannte Regeln gespeichert. Geben Sie einen Pseudo-Code mit bestmöglicher Laufzeit an, der den Knoten mit maximalem Schlüssel zurück liefert. Wenn es mehr als einen gibt, dann soll das erste gefundene im linkest möglichen Teilbaum zurückgegeben werde.

b.) Geben Sie an, in welcher Reihenfolge die Knoten des Beispiel-Baumes von ihrem Algorithmus besucht werden.

c.) Geben Sie die Laufzeit ihres Algorithmus in Θ-Notation an.

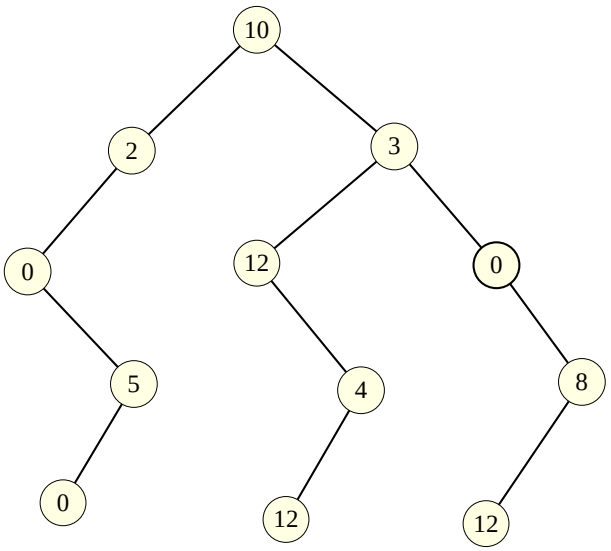

Abbildung 1: Binärer Baum - Beispiel