Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

1. Übungstest SS 2011

15. April 2011

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: Vorname:

Matrikelnummer: Studienkennzahl:

Anzahl abgegebener Zusatzblätter:

Legen Sie bitte Ihren Studentenausweis vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie auf Wunsch von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden.

Die Verwendung von Taschenrechnern, Mobiltelefonen, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

Die Arbeitszeit beträgt 55 Minuten.

	A1:	A2:	A3:	Summe:
Erreichbare Punkte:	16	16	18	50
Erreichte Punkte:				

Viel Erfolg!

Aufgabe 1.A: $\Omega/O/\Theta$-Notation **(16 Punkte)**

a) (12 Punkte)

- Bestimmen Sie jeweils die Laufzeit der unten angegebenen Codestücke A und B in Abhängigkeit von n in Θ-Notation.
- Welchen Wert haben die Variablen a und b jeweils nach dem Ausführen der Codestücke A und B in Abhängigkeit von n in Θ-Notation?

A

```
a = n;
b = 1;
c = 1;
für i = 1, ..., n {
  a = a + 1;
  b = b + 1;
}
c = a · b
solange c > 1 {
  für i = 1, ..., ⌊log n⌋ {
    b = b + 1;
  }
  c = c − a;
}
```

B

```
a = 1;
b = 1;
für i = 1, ..., n {
  solange a < n {
    a = a + 1;
    falls a < log n dann {
      b = 2 · b;
    }
  }
}
a = b²;
für k = 1, ..., a {
  b = 2 · k + b;
}
```

b) (4 Punkte)

Gegeben ist die folgende Funktion:

$$f(n) = \begin{cases} n^2 \cdot \log(n^n) + n^2 \cdot \sqrt[3]{n}, & \text{falls } n \text{ ungerade und } n > 100 \\ 4 \cdot n^3 + n^3 \cdot \log(n^2) + n^2, & \text{falls } n \text{ gerade} \\ n^4 + n^3 + n, & \text{sonst} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:
(Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.)

$f(n)$ ist	$\Theta(.)$	$O(.)$	$\Omega(.)$	keines
$n^3 \cdot \log n$				
$n^3 \cdot \sqrt[3]{n}$				
$n^2 \cdot \sqrt[3]{n}$				
n^3				

Aufgabe 2.A: Sortierverfahren und Datenstrukturen **(16 Punkte)**

a) (16 Punkte) Schreiben Sie in detailliertem Pseudocode eine Funktion *bucketSort(*L, min, max*)*, die eine einfach verkettete Liste mit dem Verweis L auf das erste Listenelement aufsteigend nach dem Prinzip *Bucket-Sort* sortieren soll. min stellt den kleinsten und max den größten Inhalt der Liste dar. Folgende Punkte sind zu beachten:

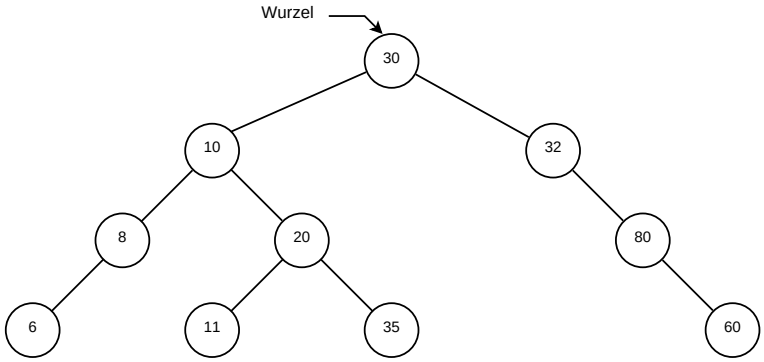
- Die Liste enthält n Elemente, wobei gilt: $n \geq 1$.
- L soll am Ende Ihres Algorithmus auf das erste Element der sortierten Liste verweisen.
- Der Inhalt der jeweiligen Listenelemente ist aus der Menge der natürlichen Zahlen $\mathbb{N}$.
- Die Anzahl der verwendeten Buckets m soll höchstens
$$m = max - min + 1$$
betragen.

Geben Sie für Ihren Algorithmus den Zeitaufwand für den Worst-Case in Θ-Notation in Abhängigkeit der Anzahl n der zu sortierenden Elemente und der Anzahl m der benötigten Buckets an.

Handelt es sich bei Ihrer Implementierung um ein stabiles Sortierverfahren? Begründen Sie Ihre Antwort.

Aufgabe 3.A: Such- und Sortierverfahren **(18 Punkte)**

a) (10 Punkte) Gegeben sei folgender binärer Baum T_1:



- Handelt es sich bei T_1 um einen AVL-Baum? Falls nicht geben Sie *alle* Gründe, an warum es sich um keinen AVL-Baum handelt und kennzeichnen diese im Baum T_1.
- Geben Sie die Post-, Pre- und Inorder Traversierungsreihenfolgen von T_1 an.
- Sollte T_1 kein AVL-Baum sein, zeichnen Sie einen AVL-Baum T_2, der die gleichen Knoten und die gleiche Preorder Traversierungsreihenfolge wie T_1 hat.

b) (8 Punkte)

Gegeben sind folgende Zahlen:

$$\langle 12, 12, 3, 2, 103, 103, 3, 2 \rangle$$

Diese Zahlen sollen mit Hilfe des Verfahrens *Insertion Sort* aufsteigend sortiert werden.

- Geben Sie die Zahlenfolge nach jeder Iteration der äußeren Schleife an. Unterstreichen Sie dabei jeweils die Elemente, die ihre Position verändert haben und kreisen Sie alle Schlüssel ein, die bereits Ihre endgültige Position erreicht haben.
- Wieviele Schlüsselvergleiche und Schlüsselbewegungen sind bei einer bereits aufsteigend sortierten Folge in Abhängigkeit der Anzahl n der zu sortierenden Schlüssel notwendig, wenn diese erneut mit Insertion Sort aufsteigend sortiert wird?

Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

1. Übungstest SS 2011

15. April 2011

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: Vorname:

Matrikelnummer: Studienkennzahl:

Anzahl abgegebener Zusatzblätter:

Legen Sie bitte Ihren Studentenausweis vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie auf Wunsch von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden.

Die Verwendung von Taschenrechnern, Mobiltelefonen, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

Die Arbeitszeit beträgt 55 Minuten.

	B1:	B2:	B3:	Summe:
Erreichbare Punkte:	18	16	16	50
Erreichte Punkte:				

Viel Glück!

Aufgabe 1.B: Such- und Sortierverfahren **(18 Punkte)**

a) (8 Punkte)

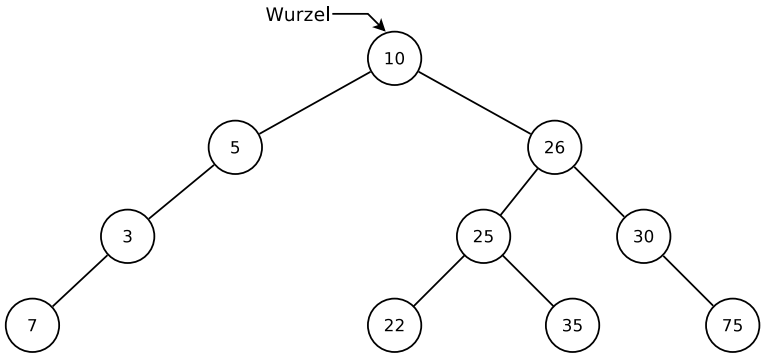
Gegeben sind folgende Zahlen:

$$\langle 15, 15, 2, 3, 17, 17, 2, 3 \rangle$$

Diese Zahlen sollen mit Hilfe des Verfahrens *Insertion Sort* aufsteigend sortiert werden.

- Geben Sie die Zahlenfolge nach jeder Iteration der äußeren Schleife an. Unterstreichen Sie dabei jeweils die Elemente, die ihre Position verändert haben und kreisen Sie alle Schlüssel ein, die bereits Ihre endgültige Position erreicht haben.
- Wieviele Schlüsselvergleiche und Schlüsselbewegungen sind bei einer bereits aufsteigend sortierten Folge in Abhängigkeit der Anzahl n der zu sortierenden Schlüssel notwendig, wenn diese erneut mit Insertion Sort aufsteigend sortiert wird?

b) (10 Punkte) Gegeben sei folgender binärer Baum T_1:



- Handelt es sich bei T_1 um einen AVL-Baum? Falls nicht geben Sie *alle* Gründe, an warum es sich um keinen AVL-Baum handelt und kennzeichnen diese im Baum T_1.
- Geben Sie die Post-, Pre- und Inorder Traversierungsreihenfolgen von T_1 an.
- Sollte T_1 kein AVL-Baum sein, zeichnen Sie einen AVL-Baum T_2, der die gleichen Knoten und die gleiche Preorder Traversierungsreihenfolge wie T_1 hat.

Aufgabe 2.B: $\Omega/O/\Theta$-Notation **(16 Punkte)**

a) (4 Punkte)

Gegeben ist die folgende Funktion:

$$f(n) = \begin{cases} n^3 \cdot \log(n^n) + n^3 \cdot \sqrt[3]{n}, & \text{falls } n \text{ gerade und } n > 100 \\ 4 \cdot n^3 + n^4 \cdot \log(n^2) + n^2, & \text{falls } n \text{ ungerade} \\ n^5 + n^3 + n, & \text{sonst} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:
(Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.)

$f(n)$ ist	$\Theta(.)$	$O(.)$	$\Omega(.)$	keines
$n^3 \cdot \sqrt[3]{n}$				
n^3				
$n^2 \cdot \sqrt[3]{n}$				
$n^3 \cdot \log n$				

b) (12 Punkte)

- Bestimmen Sie jeweils die Laufzeit der unten angegebenen Codestücke A und B in Abhängigkeit von n in Θ-Notation.
- Welchen Wert haben die Variablen a und b jeweils nach dem Ausführen der Codestücke A und B in Abhängigkeit von n in Θ-Notation?

A

```
a = 1;
b = 1;
für i = 1, ..., n {
  solange b < n {
    b = b + 1;
    falls b < log n dann {
      a = 2 · a;
    }
  }
}
b = a^2;
für k = 1, ..., b {
  a = 2 · k + a;
}
```

B

```
a = n;
b = 1;
c = 1;
für i = 1, ..., n {
  a = a + 1;
  b = b + 1;
}
c = a · b
solange c > 1 {
  für i = 1, ..., ⌊log n⌋ {
    a = a + 1;
  }
  c = c − b;
}
```

Aufgabe 3.B: Sortierverfahren und Datenstrukturen **(16 Punkte)**

a) (16 Punkte) Schreiben Sie in detailliertem Pseudocode eine Funktion *bucketSort(*L, min, max*)*, die eine einfach verkettete Liste mit dem Verweis L auf das erste Listenelement aufsteigend nach dem Prinzip *Bucket-Sort* sortieren soll. min stellt den kleinsten und max den größten Inhalt der Liste dar. Folgende Punkte sind zu beachten:

- Die Liste enthält n Elemente, wobei gilt: $n \geq 1$.
- L soll am Ende Ihres Algorithmus auf das erste Element der sortierten Liste verweisen.
- Der Inhalt der jeweiligen Listenelemente ist aus der Menge der natürlichen Zahlen $\mathbb{N}$.
- Die Anzahl der verwendeten Buckets m soll höchstens
$$m = max - min + 1$$
betragen.

Geben Sie für Ihren Algorithmus den Zeitaufwand für den Worst-Case in Θ-Notation in Abhängigkeit der Anzahl n der zu sortierenden Elemente und der Anzahl m der benötigten Buckets an.

Handelt es sich bei Ihrer Implementierung um ein stabiles Sortierverfahren? Begründen Sie Ihre Antwort.