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 2009

24. April 2009

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:	18	16	16	50
Erreichte Punkte:				

Viel Erfolg!

Aufgabe 1.A: $\Omega/O/\Theta$-Notation **(18 Punkte)**

a) (6 Punkte) Gegeben ist die folgende Funktion:

$$f(n) = \begin{cases} n \log n^2, & \text{falls } n \text{ prim} \\ n^2 + \log n, & \text{sonst} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	$\Theta(.)$	$O(.)$	$\Omega(.)$	keines
n^2				
$n\sqrt{n}$				
$n + n \log n$				

Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

b) (6 Punkte) Bestimmen Sie die Laufzeiten der unten angegebenen Algorithmen in Abhängigkeit von n in Θ-Notation. Verwenden Sie hierfür einen möglichst einfachen Term.

A

```
a = n^2;
solange a > 0 {
  für b = 1, ..., n {
    c = c + b;
  }
  a = a - n;
}
```

B

```
i = 1;
a = 1;
wiederhole {
  i = i + 1;
  a = a + i;
} bis i ≥ n;
für j = 0, ..., a {
  x = x + j;
}
```

c) (6 Punkte) Kreuzen Sie jeweils an, ob die Aussage zutrifft oder nicht.

$f(n) = \Theta(g(n))$	$\Rightarrow$	$g(n) = O(f(n))$	☐ ja	☐ nein
$f(n) = O(g(n))$	$\Rightarrow$	$g(n) = O(2f(n))$	☐ ja	☐ nein
$(f(n) = \Theta(n^2)) \wedge (g(n) = \Theta(n^2))$	$\Rightarrow$	$f(n) - g(n) = \Theta(1)$	☐ ja	☐ nein

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

a) (6 Punkte) Auf die unten angeführten Folgen A und B wird jeweils der `Partition()` Algorithmus des *Quicksort* Verfahrens angewendet um sie aufsteigend zu sortieren. Kreuzen Sie jeweils das Ergebnis an, das nach der Ausführung von `Partition()` entstanden ist (Implementierung wie im Skriptum, Pivotelement rechts).

Folge A:		Folge B:	
$\langle 1,6,9,8,3,4,7,5\rangle$		$\langle 9,8,7,6,5,4,3,1\rangle$	
Ergebnis A:		Ergebnis B:	
$\langle 1,4,3,5,9,6,7,8\rangle$	☐	$\langle 1,3,4,5,6,7,8,9\rangle$	☐
$\langle 1,6,5,9,3,4,7,8\rangle$	☐	$\langle 3,4,5,1,7,8,9,6\rangle$	☐
$\langle 1,3,4,5,9,6,7,8\rangle$	☐	$\langle 1,9,8,7,6,5,4,3\rangle$	☐
$\langle 1,4,3,5,8,6,7,9\rangle$	☐	$\langle 1,3,7,6,5,4,8,9\rangle$	☐
keine dieser Folgen	☐	keine dieser Folgen	☐

b) (10 Punkte) Gegeben sei folgender Algorithmus `Sortiere()`:

Algorithmus: `Sortiere`(**var** A, l, r)
Eingabe: Feld $A[1,...,n]$; Indexgrenzen l und r
Ausgabe: Feld A innerhalb der Grenzen l und r aufsteigend sortiert

```
minpos = l;
für i = l + 1, ..., r {
  falls A[i].key < A[minpos].key dann {
    minpos = i;
  }
}
falls minpos > l dann {
  Vertausche A[minpos] mit A[l];
}
Sortiere(A, l + 1, r);
```

Der Algorithmus wird für ein Feld A mit n Elementen durch den Aufruf von `Sortiere`$(A, 1, n)$ gestartet. *Hinweis:* Das Feld beginnt mit Index 1.

- Auf welchem aus der Vorlesung bekannten Sortierverfahren beruht der oben angeführte Algorithmus? Ist diese Implementierung korrekt? Falls nein, dann beschreiben Sie kurz das Problem.

- Stimmen die folgenden Aussagen über den Algorithmus `Sortiere()`?

Der Algorithmus beruht auf dem *Teile und Erobere*-Prinzip:	☐ ja ☐ nein
Das Sortierverfahren ist stabil:	☐ ja ☐ nein
Die max. Anzahl der Schlüsselbewegungen ist in $O(n)$:	☐ ja ☐ nein
Die max. Anzahl der Schlüsselvergleiche ist in $O(n \log n)$:	☐ ja ☐ nein

Aufgabe 3.A: Theorie **(16 Punkte)**

Bei der Beantwortung der Fragen gehen Sie von den in der Vorlesung bzw. im Skriptum vorgestellten Implementierungen der Sortieralgorithmen aus.

a) (6 Punkte) Kreuzen Sie in der folgenden Tabelle die stabilen Sortierverfahren an und geben Sie die Best- und Worst-Case Laufzeit, in Abhängigkeit von der Anzahl der zu sortierenden Elemente n, in Θ-Notation an:

	ist stabil	Best-Case Laufzeit	Worst-Case Laufzeit
Insertion-Sort	☐ ja ☐ nein		
Heap-Sort	☐ ja ☐ nein		
Fachverteilung	☐ ja ☐ nein		

b) (3 Punkte) Merge-Sort basiert bekanntlich auf dem Prinzip *Teile und Erobere*. Aus welchen drei Phasen besteht jedes *Teile und Erobere*-Verfahren? Beschreiben Sie jede der Phasen *kurz* am Beispiel von Merge-Sort.

c) (4 Punkte) Kreuzen Sie in der folgenden Tabelle an, ob die angeführten Voraussetzungen oder Aussagen für jeweils Fachverteilung und Bucketsort zutreffend sind. Im Folgenden entspricht n der Anzahl der zu sortierenden Schlüssel.

	Bucket-Sort	Fachverteilung
Schlüssel müssen Elemente eines diskreten Wertebereichs sein (z.B. ganzzahlig).	☐ ja ☐ nein	☐ ja ☐ nein
Maximale Länge der Schlüssel muss bekannt und konstant sein.	☐ ja ☐ nein	☐ ja ☐ nein
Laufzeit ist abhängig von der Anzahl der Schlüssel.	☐ ja ☐ nein	☐ ja ☐ nein
Laufzeit ist im Worst-Case $\Theta(n \log n)$.	☐ ja ☐ nein	☐ ja ☐ nein

d) (3 Punkte) Erklären Sie in wenigen Sätzen, warum eine aufsteigend sortierte Folge einen Worst-Case für das Sortieren mit Quick-Sort darstellt. Geben Sie für diesen Fall die Anzahl an Schlüsselvergleichen und -bewegungen in Abhängigkeit der Zahl der Elemente n in Θ-Notation an (es soll aufsteigend sortiert werden).

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 2009

24. April 2009

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:	16	18	16	50
Erreichte Punkte:				

Viel Glück!

Aufgabe 1.B: Theorie **(16 Punkte)**

Bei der Beantwortung der Fragen gehen Sie von den in der Vorlesung bzw. im Skriptum vorgestellten Implementierungen der Sortieralgorithmen aus!

a) (4 Punkte) Kreuzen Sie in der folgenden Tabelle an ob die angeführten Voraussetzungen und Aussagen für jeweils Fachverteilung und Bucketsort zutreffend sind. Im Folgenden entspricht n der Anzahl der zu sortierenden Schlüssel.

	Fachverteilung	Bucket-Sort
Laufzeit ist im Worst-Case $\Theta(n \log n)$.	☐ ja ☐ nein	☐ ja ☐ nein
Maximale Länge der Schlüssel muss bekannt und konstant sein.	☐ ja ☐ nein	☐ ja ☐ nein
Laufzeit ist abhängig von der Anzahl der Schlüssel.	☐ ja ☐ nein	☐ ja ☐ nein
Schlüssel müssen Elemente eines diskreten Wertebereichs sein (z.B. ganzzahlig).	☐ ja ☐ nein	☐ ja ☐ nein

b) (3 Punkte) Merge-Sort basiert bekanntlich auf dem Prinzip *Teile und Erobere*. Aus welchen drei Phasen besteht jedes *Teile und Erobere*-Verfahren? Beschreiben Sie jede der Phasen *kurz* am Beispiel von Merge-Sort.

c) (6 Punkte) Kreuzen Sie in der folgenden Tabelle die stabilen Sortierverfahren an und geben Sie die Best- und Worst-Case Laufzeit, in Abhängigkeit von der Anzahl der zu sortierenden Elemente n, in Θ-Notation an:

	ist stabil	Best-Case Laufzeit	Worst-Case Laufzeit
Insertion-Sort	☐ ja ☐ nein		
Heap-Sort	☐ ja ☐ nein		
Fachverteilung	☐ ja ☐ nein		

d) (3 Punkte) Erklären Sie in wenigen Sätzen, warum eine aufsteigend sortierte Folge einen Worst-Case für das Sortieren mit Quick-Sort darstellt. Geben Sie für diesen Fall die Anzahl an Schlüsselvergleichen und -bewegungen in Abhängigkeit der Zahl der Elemente n in Θ-Notation an (es soll aufsteigend sortiert werden).

Aufgabe 2.B: $\Omega/O/\Theta$-Notation **(18 Punkte)**

a) (6 Punkte) Bestimmen Sie die Laufzeiten der unten angegebenen Algorithmen in Abhängigkeit von n in Θ-Notation. Verwenden Sie hierfür einen möglichst einfachen Term.

A

```
für b = 1, ..., ⌊log n⌋ {
  a = 0;
  solange a < n^2 {
    c = c + b;
    a = a + n;
  }
}
```

B

```
a = 1;
für i = 1, ..., n {
  a = a + i;
}
x = 0;
für j = 0, ..., a {
  x = x + j;
}
```

b) (6 Punkte) Kreuzen Sie jeweils an, ob die Aussage zutrifft oder nicht.

$f(n) = \Omega(g(n))$	$\Rightarrow$	$g(n) = \Theta(f(n))$	☐ ja	☐ nein
$(f(n) = \Theta(n^2)) \wedge (g(n) = \Theta(n^2))$	$\Rightarrow$	$f(n) - g(n) = \Theta(1)$	☐ ja	☐ nein
$f(n) = \Omega(g(n))$	$\Rightarrow$	$g(n) = \Omega(f(n)/2)$	☐ ja	☐ nein

c) (6 Punkte) Gegeben ist die folgende Funktion:

$$f(n) = \begin{cases} n \log n^3, & \text{falls } n \text{ gerade} \\ n^3 + \log n, & \text{sonst} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	$\Theta(.)$	$O(.)$	$\Omega(.)$	keines
n^3				
$n \log n + n$				
$n\sqrt{n}$				

Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

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

a) (10 Punkte) Gegeben sei folgender Algorithmus `Sortiere()`:

Algorithmus: `Sortiere`(**var** A, l, r)
Eingabe: Feld $A[1, ..., n]$; Indexgrenzen l und r
Ausgabe: Feld A innerhalb der Grenzen l und r aufsteigend sortiert

```
minpos = l;
für i = l + 1, ..., r {
  falls A[i].key < A[minpos].key dann {
    minpos = i;
  }
}
falls minpos > l dann {
  Vertausche A[minpos] mit A[l];
}
Sortiere(A, l + 1, r);
```

Der Algorithmus wird für ein Feld A mit n Elementen durch den Aufruf von `Sortiere`$(A, 1, n)$ gestartet. *Hinweis:* Das Feld beginnt mit Index 1.

- Auf welchem aus der Vorlesung bekannten Sortierverfahren beruht der oben angeführte Algorithmus? Ist diese Implementierung korrekt? Falls nein, dann beschreiben Sie kurz das Problem.

- Stimmen die folgenden Aussagen über den Algorithmus `Sortiere()`?

Der Algorithmus beruht auf dem *Teile und Erobere*-Prinzip:	☐ ja ☐ nein
Das Sortierverfahren ist stabil:	☐ ja ☐ nein
Die max. Anzahl der Schlüsselbewegungen ist in $O(n)$:	☐ ja ☐ nein
Die max. Anzahl der Schlüsselvergleiche ist in $O(n \log n)$:	☐ ja ☐ nein

b) (6 Punkte) Auf die unten angeführten Folgen A und B wird jeweils der Algorithmus `Partition()` des *Quicksort* Verfahrens angewendet um sie aufsteigend zu sortieren. Kreuzen Sie jeweils das Ergebnis an, das nach der Ausführung von `Partition()` entstanden ist (Implementierung wie im Skriptum, Pivotelement rechts).

Folge A:
$\langle 9, 8, 7, 6, 5, 4, 3, 2 \rangle$

Ergebnis A:

$\langle 2, 9, 8, 7, 6, 5, 4, 3 \rangle$	☐
$\langle 3, 4, 5, 2, 7, 8, 9, 6 \rangle$	☐
$\langle 2, 6, 5, 9, 3, 4, 7, 8 \rangle$	☐
$\langle 2, 3, 7, 6, 5, 4, 8, 9 \rangle$	☐
keine dieser Folgen	☐

Folge B:
$\langle 2, 6, 9, 8, 3, 4, 7, 5 \rangle$

Ergebnis B:

$\langle 2, 3, 4, 5, 9, 6, 7, 8 \rangle$	☐
$\langle 2, 3, 4, 5, 6, 7, 8, 9 \rangle$	☐
$\langle 2, 4, 3, 5, 9, 6, 7, 8 \rangle$	☐
$\langle 2, 4, 3, 5, 8, 6, 7, 9 \rangle$	☐
keine dieser Folgen	☐