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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

1. Übungstest WS 2008

07. November 2008

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: Ω/O/Θ-Notation **(18 Punkte)**

a) (6 Punkte)

Gegeben sei die folgende Funktion:

$$f(n) = \begin{cases} \frac{5}{3}n^2 \log_3 n - 3^9 \cdot n\sqrt{n}, & \text{falls } n \text{ durch 3 teilbar} \\ \frac{2}{3} + 3n - 9n^2 + \frac{1}{3}n^3, & \text{sonst} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	$\Omega(\cdot)$	$\Theta(\cdot)$	$O(\cdot)$	keines
$n^2\sqrt{n}$				
$n^2 \log_3 n + n^3$				
$n^2 \log_2 n^n$				

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

b) (6 Punkte)

Seien $f(n)$, $g(n)$ und $h(n)$ Funktionen mit positivem Wertebereich. Kreuzen Sie in der folgenden Tabelle jeweils alle richtigen Folgerungen an. Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

Annahme	Folgerung: $f(n)$ ist in		
	$O(h(n))$	$\Theta(g(n))$	$O\left(\frac{2g(n)}{3}\right)$
$f(n) = O(g(n)) \quad \wedge \quad h(n) = \Omega(g(n))$			
$f(n) = O(g(n)^2) \quad \wedge \quad h(n) = \Theta(f(n))$			
$f(n) = O(g(n)) \quad \wedge \quad f(n) = \Omega(g(n))$	undefiniert		

c) (6 Punkte)

- Bestimmen Sie die Laufzeiten des unten angegebenen Algorithmus A in Abhängigkeit von n in Θ-Notation.
- Ergänzen sie den Algorithmus C so, dass er in Abhängigkeit von n die angegebene Laufzeit hat.

A

```
a = n^2;
j = log n;
b = 2^n;
solange a > 0 {
  b = ∛b;
  für i = j, j−1, ..., 1 {
    b = 2b + i;
  }
  a = a − n;
}
```

C $\Theta(n^2 \cdot \sqrt{n})$

```
für l = 1, ..., [      ] {
  k = n;
  wiederhole {
    k = [      ];
    m = n · k;
  } bis k ≤ 1
}
```

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

a) (8 Punkte)

- Sortieren Sie die nachfolgenden Zahlen **aufsteigend** mittels **Quick-Sort**:

  $$\langle 30, 10, 20, 25, 50, 70, 45, 40 \rangle.$$

  Geben Sie die Zahlenfolge nach **jedem Aufruf** von `Partition()` an. Markieren Sie in jeder dieser Folgen die Zahlen, die sich bereits an ihrer endgültigen Position befinden.
- Beschreiben Sie eine Zahlenfolge, die bei Quick-Sort **immer** eine quadratische Laufzeit $\Theta(n^2)$, in Abhängigkeit der Größe der Zahlenfolge n, erfordert. Begründen Sie ihre Antwort in maximal 2-3 Sätzen.

b) (8 Punkte)

Gegeben sei der unten angeführte Code zum Algorithmus `Sortiere()` in Pseudocode, der für ein Feld $A = (A[1], \ldots, A[n])$ aufgerufen wird.

Vervollständigen Sie den Algorithmus `Sortiere()` so, dass er eine aufsteigend zu sortierende Zahlenfolge durchläuft und dabei jeweils Paare direkt benachbarter Zahlen vergleicht und nötigenfalls vertauscht.

Achten Sie darauf, dass der fertige Algorithmus möglichst effizient arbeitet.

```
Sortiere(var A, n)
 1: i = 0;
 2: wiederhole {
 3:     i = i + 1;
 4:     [                                   ]
 5:     für j = 1...[      ] {
 6:         falls A[j + 1] [  ] A[j] dann {
 7:             vertausche [                ]
 8:             vertauscht = wahr;
 9:         }
10:     }
11: } bis [                                ]
```

Bestimmen Sie die Laufzeit im Worst- und Best-Case ihrer Version von `Sortiere()` in Θ-Notation in Abhängigkeit der Eingabegröße n.

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) (4 Punkte)

Für die Analyse von Sortierverfahren sind neben der reinen Laufzeit auch zwei weitere Kenngrößen von großer Bedeutung: Schlüsselvergleiche und Datenbewegungen.

Kreuzen Sie in der folgenden Tabelle die zutreffenden Anworten an. Die angekreuzte Antwort muss **sowohl** für den **Best- als auch den Worst-Case** gelten!

	Bewegungen		Vergleiche	
	$O(n)$	$\Omega(n \log n)$	$\Omega(n \log n)$	$O(n \log n)$
Selection-Sort	☐ ja ☐ nein	☐ ja ☐ nein	☐ ja ☐ nein	☐ ja ☐ nein
Quick-Sort	☐ ja ☐ nein	☐ ja ☐ nein	☐ ja ☐ nein	☐ ja ☐ nein

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

b) (2 Punkte)

In eine bereits absteigend sortierte Folge wird ein beliebig großes Element am Ende angehängt. Die neu entstandene Folge soll nun erneut absteigend sortiert werden. Welches Sortierverfahren eignet sich bei diesem Szenario am Besten? Begründen Sie *kurz* ihre Antwort.

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) (4 Punkte)

Eine Folge von Zahlen soll mittels Heapsort **absteigend** sortiert werden. Geben Sie an, ob dafür ein Maximum- oder ein Minimum-Heap nötig ist. Begründen Sie *kurz* Ihre Antwort.

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

186.172 Algorithmen und Datenstrukturen 1 VL 4.0

1. Übungstest WS 2008

07. November 2008

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 Glück!

Aufgabe 1.B: Ω/O/Θ-Notation **(18 Punkte)**

a) (6 Punkte)

Seien $f(n)$, $g(n)$ und $h(n)$ Funktionen mit positivem Wertebereich. Kreuzen Sie in der folgenden Tabelle jeweils alle richtigen Folgerungen an. Jede Zeile wird nur dann gewertet, wenn sie vollständig richtig ist.

Annahme	Folgerung: $f(n)$ ist in		
	$\Omega(h(n))$	$\Theta(g(n))$	$\Omega\left(\frac{3g(n)}{2}\right)$
$f(n) = \Omega(\sqrt[3]{g(n)}) \quad \wedge \quad h(n) = \Theta(f(n))$			
$f(n) = \Omega(g(n)) \quad \wedge \quad h(n) = O(g(n))$			
$f(n) = \Omega(g(n)) \quad \wedge \quad f(n) = O(g(n))$	undefiniert		

b) (6 Punkte)

Gegeben sei die folgende Funktion:

$$f(n) = \begin{cases} \frac{5}{2} n \log_2 n - 2^9 \cdot \sqrt{n}, & \text{falls } n \text{ durch 2 teilbar} \\ \frac{3}{2} - 2n + 3n^2, & \text{sonst} \end{cases}$$

Kreuzen Sie in der folgenden Tabelle die zutreffenden Felder an:

$f(n)$ ist	$\Omega(\cdot)$	$\Theta(\cdot)$	$O(\cdot)$	keines
$n \log_2 n^n$				
$n\sqrt{n}$				
$n \log_2 n + n^2$				

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

c) (6 Punkte)

- Bestimmen Sie die Laufzeiten des unten angegebenen Algorithmus A in Abhängigkeit von n in Θ-Notation.
- Ergänzen sie den Algorithmus C so, dass er in Abhängigkeit von n die angegebene Laufzeit hat.

A

```
j = √n;
a = n²;
b = 2ⁿ;
wiederhole {
  für i = j, j−1, ..., 1 {
    b = 2b + i;
  }
  a = a − n;
  b = ∛b;
} bis a > 0
```

C $\Theta(n^3 \cdot \log n)$

```
für l = 1, ..., [      ] {
  k = n;
  wiederhole {
    k = [      ];
    m = n · k;
  } bis k ≤ 1
}
```

Aufgabe 2.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) (2 Punkte)

Eine beliebige Folge soll aufsteigend sortiert werden. Welche Sortierverfahren eignen sich bei diesem Szenario, wenn im Worst-Case eine Laufzeit, in Abhängigkeit der Größe der Zahlenfolge n, von $\Theta(n \log n)$ nicht überschritten werden sollte? Begründen Sie *kurz* ihre Antwort.

b) (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
Quick-Sort	☐ ja ☐ nein		
Insertion-Sort	☐ ja ☐ nein		
Fachverteilung	☐ ja ☐ nein		

c) (4 Punkte)

Eine Folge von Zahlen soll mittels Heapsort **aufsteigend** sortiert werden. Geben Sie an, ob dafür ein Maximum- oder ein Minimum-Heap nötig ist. Begründen Sie kurz Ihre Antwort.

d) (4 Punkte)

Für die Analyse von Sortierverfahren sind neben der reinen Laufzeit auch zwei weitere Kenngrößen von großer Bedeutung: Schlüsselvergleiche und Datenbewegungen.

Kreuzen Sie in der folgenden Tabelle die zutreffenden Anworten an. Die angekreuzte Antwort muss **sowohl** für den **Best- als auch den Worst-Case** gelten!

	Bewegungen		Vergleiche	
	$O(n)$	$\Omega(n \log n)$	$\Omega(n \log n)$	$O(n \log n)$
Selection-Sort	☐ ja ☐ nein	☐ ja ☐ nein	☐ ja ☐ nein	☐ ja ☐ nein
Merge-Sort	☐ ja ☐ nein	☐ ja ☐ nein	☐ ja ☐ nein	☐ ja ☐ nein

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

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

a) (8 Punkte)

- Sortieren Sie die nachfolgenden Zahlen **aufsteigend** mittels **Quick-Sort**:

  $$\langle 40, 20, 30, 35, 60, 80, 55, 50 \rangle.$$

  Geben Sie die Zahlenfolge nach **jedem Aufruf** von `Partition()` an. Markieren Sie in jeder dieser Folgen die Zahlen, die sich bereits an ihrer endgültigen Position befinden.
- Beschreiben Sie eine Zahlenfolge, die bei Quick-Sort **immer** eine quadratische Laufzeit $\Theta(n^2)$, in Abhängigkeit der Größe der Zahlenfolge n, erfordert. Begründen Sie ihre Antwort in maximal 2-3 Sätzen.

b) (8 Punkte)

Gegeben sei der unten angeführte Code zum Algorithmus `Sortiere()` in Pseudocode, der für ein Feld $A = (A[1], \ldots, A[n])$ aufgerufen wird.

Vervollständigen Sie den Algorithmus `Sortiere()` so, dass er eine aufsteigend zu sortierende Zahlenfolge durchläuft und dabei jeweils Paare direkt benachbarter Zahlen vergleicht und nötigenfalls vertauscht.

Achten Sie darauf, dass der fertige Algorithmus möglichst effizient arbeitet.

```
Sortiere(var A, n)
 1: i = 1;
 2: wiederhole {
 3:     [                                   ]
 4:     für j = 2...[      ] {
 5:         falls A[j − 1] [ ] A[j] dann {
 6:             vertausche [                ]
 7:             vertauscht = wahr;
 8:         }
 9:     }
10:     i = i + 1;
11: } bis [                                 ]
```

Bestimmen Sie die Laufzeit im Worst- und Best-Case ihrer Version von `Sortiere()` in Θ-Notation in Abhängigkeit der Eingabegröße n.