Technische Universität Wien
Institut für Computergraphik und Algorithmen
Algorithms and Complexity Group

186.813 Algorithmen und Datenstrukturen 1 VU 6.0

1. Übungstest SS 2015

30. April 2015

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: | Vorname:

Matrikelnummer: | Unterschrift:

Legen Sie während der Prüfung Ihren Ausweis für Studierende vor sich auf das Pult.

Sie dürfen die Lösungen nur auf die Angabeblätter schreiben, die Sie von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden. Benutzen Sie bitte dokumentenechte Schreibgeräte (keine Bleistifte!).

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

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

Viel Erfolg!

Aufgabe 1.A: Ω/O/Θ-Notation **(16 Punkte)**

a) (8 Punkte)

Gegeben sei die folgende Funktion $f : \mathbb{N}^+ \to \mathbb{R}$

$$f(n) = \begin{cases} n! \cdot \ln n + \dfrac{1}{n^2} - 20 & \text{wenn } n \text{ prim und gerade} \\ 3n^6 + 2^{n+3} + \dfrac{n^7}{\ln n^3 \cdot 4n^2} & \text{wenn } n \text{ ungerade} \\ 4^n \cdot \dfrac{5}{2^n} + 200n^5 \cdot \ln n - 200 & \text{sonst} \end{cases}$$

Kreuzen Sie **in der folgenden Tabelle** die zutreffenden Felder an:

$f(n)$ ist	$O(.)$	$\Omega(.)$	$\Theta(.)$	keines
$\ln n$				
2^n				
$n!$				
3^n				

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

Platz für Notizen:

b) (8 Punkte)

Tragen Sie für die Codestücke A und B jeweils die **Laufzeit** in Abhängigkeit von n und die **Werte der Variablen** a **und** b nach dem Ausführen des Codes in Θ-Notation in die unten stehenden Tabellen ein.

A

$a = 1;$
$b = 1;$
für $c = 1, \ldots, \lfloor\sqrt{n}\rfloor$ **{**
  $a = a + a;$
  $b = b + c;$
}
$i = 2^{2b};$
solange $i > 1$ **{**
  $i = \lfloor\frac{i}{2}\rfloor;$
}

Laufzeit	a	b
$\Theta($ $)$	$\Theta($ $)$	$\Theta($ $)$

B

$a = n^2;$
$b = n;$
$c = n;$
wiederhole
  $a = a + c;$
  $c = \lfloor c - \frac{n}{4}\rfloor;$
  $b = b \cdot b;$
bis $c < 1$
solange $a > n^2$ **{**
  $a = \lfloor\sqrt{a}\rfloor;$
}

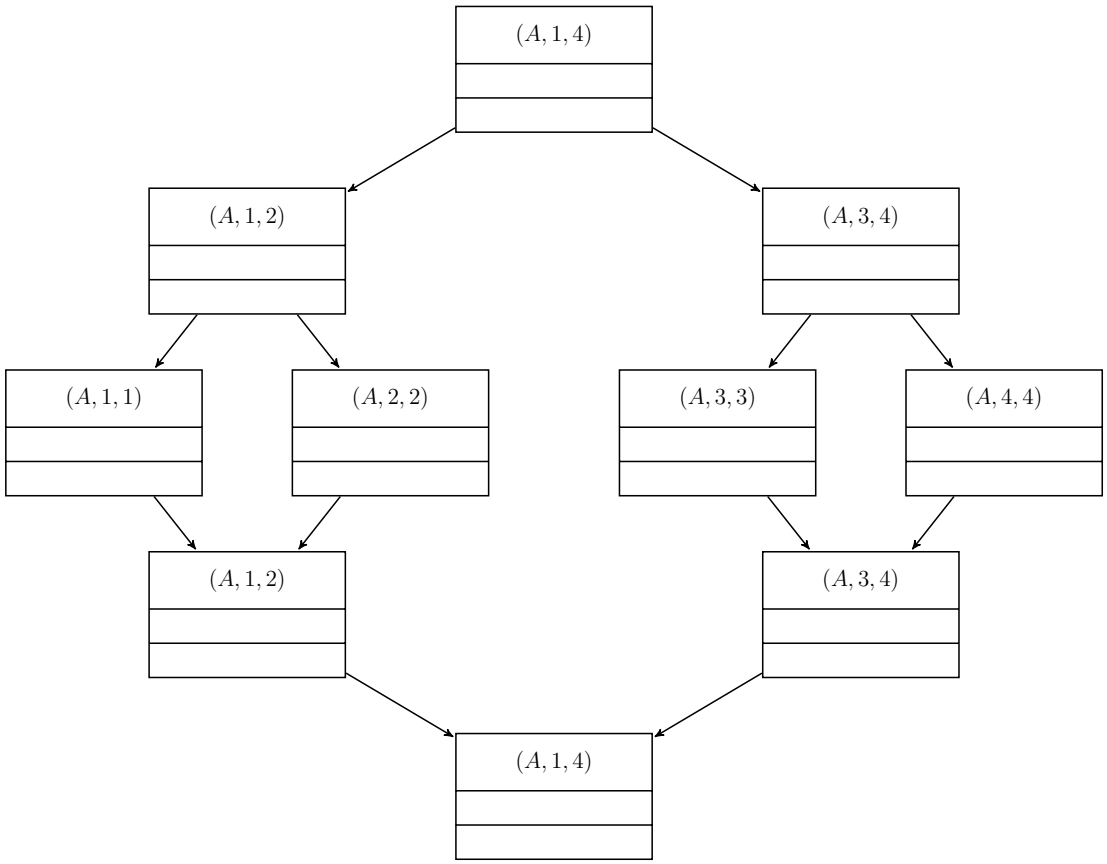
Laufzeit	a	b
$\Theta($ $)$	$\Theta($ $)$	$\Theta($ $)$

Platz für Notizen:

Aufgabe 2.A: Sortierverfahren **(18 Punkte)**

a) (8 Punkte)
Die folgende Abbildung zeigt den Ablauf der Funktion Merge-Sort beim aufsteigenden Sortieren durch Verschmelzen für ein Feld mit vier Elementen. Es soll das Feld $A = \{19, 8, 4, 3\}$ sortiert werden. Lösen Sie die folgenden Aufgaben entsprechend des Pseudocodes für Sortieren durch Verschmelzen, der in der Vorlesung bzw. dem Skriptum präsentiert wurde!

- Nummerieren Sie unter Verwendung der ersten Zeile in jedem Knoten die Aufrufe entsprechend der Vorgehensweise von Sortieren durch Verschmelzen. Die Nummerierung soll chronologisch im Bezug auf die Reihenfolge der Aufrufe erfolgen. Knoten, die nicht der maximalen Rekursionstiefe entsprechen, kommen jeweils zweimal vor. Die Nummer im ersten Vorkommnis soll dem Zeitpunkt des Aufrufs entsprechen und die Nummer des zweiten Vorkommens dem Zeitpunkt zu dem der Aufruf endet.
- Tragen Sie in der zweiten Zeile in jedem Knoten das aktuelle Feld A ein.



b) Ein Sortierverfahren heißt stabil, wenn die Reihenfolge der Elemente mit gleichem Schlüssel vor und nach dem Sortiervorgang gleich ist. Berücksichtigen Sie zum Lösen der folgenden Aufgaben jeweils den Pseudocode (und damit das Prinzip), der in der Vorlesung bzw. dem Skriptum für die betreffenden Sortierverfahren präsentiert wurde!

i) (4 Punkte) Kreuzen Sie in der folgenden Tabelle an, welche Sortierverfahren stabil sind und welche nicht.

	InsertionSort	SelectionSort	QuickSort	MergeSort
stabil				
nicht stabil				

(1 Punkt pro richtiger Spalte, -1 Punkt pro falscher Spalte, 0 Punkte für leere Spalten, Negativpunkte übertragen sich nicht auf die übrigen Beispiele)

ii) (6 Punkte) Wählen Sie ein nicht stabiles Sortierverfahren (nicht notwendigerweise aus der Tabelle) und geben sie eine Eingabesequenz an, für die der gewählte Algorithmus kein stabiles Ergebnis retourniert. Stellen Sie die einzelnen Schritte der Berechnung dar und kennzeichnen Sie, wo die Stabilität verloren geht.

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

Gegeben ist eine einfach verkettete, azyklische Zahlenliste A.

a) (12 Punkte)

Vervollständigen Sie den Pseudocode der Funktion `Oddeven(A, i, j)`, die die Reihenfolge der Elemente an den Positionen i bis j in der Liste A wie folgt ändert: Alle Elemente an ungeraden Positionen befinden sich danach vor allen Elementen an geraden Positionen. Die Reihenfolge *innerhalb* der Elemente an ungeraden Positionen darf sich gegenüber der ursprünglichen Reihenfolge *nicht* ändern.

b) (4 Punkte)

Geben Sie die minimale Anzahl an Vertauschungen an, die benötigt wird, um `Oddeven(A, i, j)` zu implementieren, wenn i und j gerade sind.

Sie können davon ausgehen, dass die Liste nicht leer ist und $1 \leq i < j \leq |A|$ gilt. Weiters kann die Länge der Liste $|A|$ in konstanter Zeit ermittelt werden. Der Zeiger `A` verweist auf das erste Element der Liste A. Die Funktion `ungerade(x)` testet, ob x ungerade ist. Die Funktion `vertausche(x, y)` vertauscht zwei Listenelemente (durch `x` und `y` referenziert) und danach auch die Zeiger, d.h. die Zeiger (`x` und `y`) bleiben in der ursprünglichen Reihenfolge erhalten.

Weitere Hinweise:

- Es ist nicht erlaubt, zusätzlichen Speicher zu verwenden.
- Die leeren Zeilen in der Schleife reichen für eine Lösung aus.

Oddeven(A, i, j)

```
p = 1;
a = A;
solange p < i {
  a = a.next;
}
falls ungerade(i) dann {
  a = a.next;
  p = p + 1;
}
b = a.next;
q = p + 1;
solange q ≤ j {
  vertausche(a, b);
}
```

Platz für Notizen: