Algorithmen und Datenstrukturen 1

Abschlussklausur SoSe 2013
08. Juli 2013

Prof. Dr. Carsten Lecon
Fachhochschule Aalen
Fakultät Elektronik und Informatik
Studiengang Informatik

Name:	
Vorname:	
Semester:	
Matrikelnummer:	

- Bitte geben Sie diesen Zettel sowie die übrigen Aufgabenzettel zusammen mit den Lösungsblättern ab.
- Das erste Blatt der abgegebenen Klausur muss dieses **Deckblatt** sein.
- Die Blätter sind nach Aufgaben **aufsteigend sortiert.**
- Die Blätter sind **zusammengeheftet.** Nicht angeheftete Blätter werden nicht gewertet!
- Alle Aufgaben können auf den vorliegenden Blättern gelöst werden (ggf. Rückseite verwenden).

Viel Erfolg!

Einträge in der folgenden Tabelle sind nicht vom Prüfling auszufüllen
(dennoch getätigte Angaben werden ggf. korrigiert …).

Aufgabe	Thema	Max. Punkte	Erreichte Punkte
1	Bin. Suchb.	7	
2	Rekursion	10	
3	Sortieralg.	40	
4	Sortieralg. II	28	
5	Rot-Schw.-B.	20	
6	Allg. Fragen	16	
Punkte aus Übungen		(10)	
SUMME		121	

1 Binäre Suchbäume [7]

a) Fügen Sie folgende Zahlen in einen anfangs leeren binären Suchbaum und zeichnen ihn [2].

2 – 3 – 10 – 4 – 6 – 5 – 1 – 17 – 11

b) Geben Sie die Reihenfolge der Knotenwerte an, wenn der Baum in *Preorder* durchlaufen wird [2].

b) Löschen Sie aus dem Suchbaum, den Sie in der obigen Aufgabe erstellt haben, den Knoten mit dem Wert 10. Zeichnen Sie **beide** Möglichkeiten. [3]

2 Rekursion [10]
Gegeben sei folgende rekursive Funktion:

$$f : \mathbf{N}_0 \to \mathbf{N}_0$$

$$f(n) = \begin{cases} 0 & \text{für } n = 0 \\ f(n-1) + n & \text{für } n > 0 \end{cases}$$

a) Geben Sie die Werte für folgende Funktionswerte an: [4]

n	**1**	**2**	**3**	**4**	**5**	**6**
f(n)						

b) Was bewirkt diese Methode? [2]

c) Stellen Sie den rekursiven Aufruf von 5 grafisch dar. [4]

3 Sortieralgorithmen [40]
Gegeben sei folgende Zahlenfolge aus neun Zahlen (die erste Zeile bezeichnet den Index):

#	1	2	3	4	5	6	7	8	9
	2	7	9	1	5	6	8	4	3

Sortieren Sie diese Zahlenfolge mittels

a) Insertionsort [7]

b) Bubblesort1 [7]

c) Bubblesort2 [7]

d) Mergesort [4]

e) Quicksort (erste Variante). Notieren Sie auch die rekursiven Aufrufe. Alternativ können Sie in der beiliegenden Tabelle in den Spalten „left“ und „right“ die entsprechenden linken und rechten Array-Indexe angeben. [15]

Tragen Sie die Zwischenergebnisse der folgenden Sortiervorgänge in die beiliegenden Tabellen ein (Anlage).

4 Sortieralgorithmen II [28]
Das Sortierverfahren *supersort* (Name geändert) arbeitet gemäß folgendem Algorithmus:

```
procedure supersort(A)
  anfang=0
  do
    anfang = anfang+1
    for i=anfang to length(A)-1
      if A[i] > A[i+1] then
        swap A[i] ↔ A[i+1 ]
      end if
    end for
    if keine-Vertauschung-mehr then
      brich-do-until-Schleife-ab
    end if
    for i=length(A)-1 to anfang
      if A[i] > A[i+1] then
        swap A[i] ↔ A[i+1]
      end if
    end for
  until keine-Vertauschung-mehr
end procedure
```

a) Wie ist die Laufzeit dieses Algorithmus (in O-Notation)? [2]

b) Sortieren Sie mit diesem Algorithmus die Zahlenfolge

2 7 9 1 5 6 8 4 3

Geben Sie jeweils die Zwischenstände nach jedem Durchlauf der äußeren `do`-Schleife an. [6]

c) Schreiben Sie eine entsprechende Java-Methode (auf die folgende Seite)

```
public static void sort(int[] A)
```

die diesen Algorithmus implementiert. Eine `swap`-Methode sei bereits mit folgender Signatur implementiert: [20]

```
swap(int[] A, int i, int j)
```

Aufgabe 4.c [Fortsetzung]

5 Rot-Schwarz-Baum [20]
Fügen Sie in einen anfangs leeren Rot-Schwarz-Baum nacheinander Knoten mit folgenden Schlüsseln ein:

2 7 9 1 5 6 8 4 3

Zeichnen Sie jeweils den resultierenden Baum. Geben Sie jeweils den Zustand direkt nach dem Einfügen und ggf. nach einer Wiederherstellung der Rot-Schwarz-Eigenschaft an. Notieren Sie, welche Aktionen Sie ggf. tätigen („Fall 1b“, „Linksrotation um x“, „Rechtsrotation um y“).

Aufgabe 5 [Fortsetzung]

6 Allgemeine Fragen [16]

6.1 Komplexität von Algorithmen [2]

Welche der folgenden Laufzeiten bezeichnet den schnellsten, den zweitschnellsten, den drittschnellsten und den viertschnellsten Algorithmus?

a. Kubische Laufzeit
b. Kubische Laufzeit
c. Logarithmische Laufzeit
d. Exponentielle Laufzeit

Schnellste Laufzeit: ___
Zweitschnellste Laufzeit: ___
Drittschnellste Laufzeit: ___
Viertschnellste Laufzeit: ___

6.2 Komplexität von Algorithmen II [3]

Gegeben sei eine logische Formel mit n Variablen. Gesucht ist eine Belegung der Variablen, so dass die Formel wahr (erfüllt) ist. Der Algorithmus probiert dazu alle Variablen-Belegungen.
Wie ist die Laufzeit dieses Algorithmus (in O-Notation)?

6.2 Sortieralgorithmen [1]

Welchen Vorteil hat der *Insertionsort* gegenüber dem *Shellsort*?

6.3 Heap [10]

Gegeben sei folgendes Array (die erste Zeile bezeichnet den Index):

#	1	2	3	4	5	6	7	8	9
	2	7	9	1	5	6	8	4	3

a) Zeichnen Sie das Array als Heap. [2]
b) Handelt es sich bei diesem Heap um einen Min-Heap? Begründen Sie Ihre Antwort. [2]
c) Erstellen Sie aus diesem Heap einen Max-Heap. Schreiben Sie alle Zwischenschritte auf. [6]

#	1	2	3	4	5	6	7	8	9
	2	**7**	**9**	**1**	**5**	**6**	**8**	**4**	**3**

Tabelle für Insertionsort

i=		1	2	3	4	5	6	7	8	9
		2	7	1	5	6	8	4	3	9

Tabelle für Bubblesort1

1	2	3	4	5	6	7	8	9
2	7	1	5	6	8	4	3	9

Tabelle für Bubblesort2

1	2	3	4	5	6	7	8	9
2	7	1	5	6	8	4	3	9

Tabelle für Quicksort

left	right	1	2	3	4	5	6	7	8	9
1	9	2	7	1	5	6	8	4	3	9

LÖSUNGEN

2 Rekursion [10]
Gegeben sei die folgende rekursive Funktion:

$$f : \mathbf{N}_0 \to \mathbf{N}_0$$

$$f(n) = \begin{cases} 0 & \text{für } n = 0 \\ f(n-1) + n & \text{für } n > 0 \end{cases}$$

a) Geben Sie die Werte für folgende Eingaben an:

n	**1**	**2**	**3**	**4**	**5**	**6**
f(n)	1	3	6	10	15	21

b) Was bewirkt diese Methode?
Summenfunktion

c) Stellen Sie den rekursiven Aufruf von 5 grafisch dar.

Rot-Schwarz-Baum

4 Rot-Schwarz-Baum [20]
Fügen Sie in einen anfangs leeren Rot-Schwarz-Baum nacheinander Knoten mit folgenden Schlüsseln ein:

	2	7	9	1	5	6	8	4	3
Punkte	1	1	4	2	1	2	1	1	7

6.1 Komplexität
Welche der folgenden Laufzeiten bezeichnet den schnellsten, den zweitschnellsten, den drittschnellsten und den viertschnellsten Algorithmus?

a. Kubische Laufzeit
b. Kubische Laufzeit
c. Logarithmische Laufzeit
d. Exponentielle Laufzeit

Schnellste Laufzeit: **c**
Zweitschnellste Laufzeit: **a / b**___
Drittschnellste Laufzeit: **a / b**___
Viertschnellste Laufzeit: **d**

6.2 Komplexität
Gegeben sei eine logische Formel mit n Variablen. Gesucht ist eine Belegung der Variablen so dass die Formel wahr (erfüllt) ist. Der Algorithmus probiert alle Variablen-Belegungen.
Wie ist die Laufzeit dieses Algorithmus (in O-Notation)?
→ **$O(2^n)$**

Sortieralgorithmen II

Das Sortierverfahren *supersort* (Name frei ausgedacht) arbeitet gemäß folgendem Algorithmus:

```
procedure supersort(A)
  anfang=0
  do
    anfang = anfang+1
    for i=anfang to length(A)-1
      if A[i] > A[i+1] then
        swap A[i] ↔ A[i+1 ]
      end if
    end for
    if keine-Vertauschung-mehr then
      brich-do-until-Schleife-ab
    end if
    for i=length(A)-1 to anfang
      if A[i] > A[i+1] then
        swap A[i] ↔ A[i+1]
      end if
    end for
  until keine-Vertauschung-mehr
end procedure
```

a) Wie ist die Laufzeit dieses Algorithmus (in O-Notation)?
O(n2) – bei fast sortierten Array O(n)

b) Sortieren Sie mit diesem Algorithmus die Zahlenfolge

2 7 9 1 5 6 8 4 3

Geben Sie jeweils die Zwischenstände nach jedem Durchlauf der äußeren `do`-Schleife an.

```
Vor Sortierung:
2 7 9 1 5 6 8 4 3
+++ Nach Durchlauf:
1 2 7 3 5 6 8 4 9
+++ Nach Durchlauf:
1 2 3 4 5 6 7 8 9
Nach Sortierung (Ripplesort):
1 2 3 4 5 6 7 8 9
```

c) Schreiben Sie eine entsprechende Java-Methode

```
public static void sort(int[] A)
```

die diesen Algorithmus implementiert. Eine `swap`-Methode sei bereits wie folgt implementiert:

```
swap(int[] A, int i, int j
```

Lösung:

Bei dem Verfahren handelt es sich um den Sortieralgorithmus *Ripplesort* (bzw. *Shakersort*).

```
public static void supersort() {
     boolean tausch;
     int anfang=-1;
     do {
          anfang++;
          tausch = false;
          for (int i=anfang; i<(A.length-1); i++) {
               if (A[i] > A[i+1]) {
                    swap(A, i, i+1);
                    /*
                    tausch = true;
               } // if
          } // for

          if (!tausch) break;

          for (int i=A.length-2; i>=anfang; i--) {
               if (A[i] > A[i+1]) {
                    swap(A, i, i+1);
                    tausch = true;
               } // if
          } // for
     } while (tausch);
}
```