Algorithmen und Datenstrukturen 1

Abschlussklausur WS 2012 / 2013

05. Februar 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!
- Bis auf die Aufgaben 4 (Rot-Schwarz-Baum) und 5.3 (Heap) können Sie alle Aufgaben auf den vorliegenden Blättern lösen.

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	BST	5	
2	Rekursion	10	
3	Sortieralg.	51	
4	Rot-Schw.-B.	20	
5	Allg. Fragen	35	
Punkte aus Übungen			
SUMME		121	

1 Binäre Suchbäume [5]

a) Fügen Sie die Zahlen der folgenden Tabelle in einen anfangs leeren binären Suchbaum und zeichnen ihn.

5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8

b) Löschen Sie aus dem Suchbaum, den Sie in der obigen Aufgabe erstellt haben, den Wurzelknoten. Zeichnen Sie den resultierenden Baum und beschreiben Sie Ihr Vorgehen.

c) Geben Sie eine Einfügereihenfolge der obigen Zahlen an, damit ein Baum mit minimaler Höhe resultiert. Sie müssen diesen Baum nicht zeichnen.

2 Rekursion [10]
Hinweis: Diese Aufgabe bitte direkt auf dem Blatt lösen.

Gegeben sei die folgende rekursive Java-Methode:

```
public static String algo1(String s) {

      if (s.isEmpty()) {
            return s;
      }
      if (s.length() == 1) {
            return s;
      }
      return s.charAt(s.length() - 1)
                  + algo1(s.substring(1, s.length() - 1))
                  + s.charAt(0);
}
```

a) Geben Sie die Rückgabewerte für folgende Eingaben an:

s	**Rückgabewert**
„A“	
„CCC“	
„HALLO“	
„PRÜFUNG“	
„ALGORITHMEN UND DS 1“	

b) Was bewirkt diese Methode?

c) Stellen Sie den Aufruf von „PRÜFUNG“ grafisch dar.

3 Sortieralgorithmen [51]
Gegeben sei folgende Zahlenfolge aus 16 Zahlen (die erste Zeile bezeichnet den Index):

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8

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

a) Sortieren Sie diese Zahlenfolge mit dem **Shellsort**. **[10]**

b) Sortieren Sie diese Zahlenfolge mit dem **Insertionsort**. **[8]**

c) Vergleichen Sie die Anzahl der Verschiebungen von Shellsort und Insertionsort. **[1]**

d) Sortieren Sie die Zahlenfolge mit dem **Bubblesort2**. **[8]**

e) Vergleichen Sie die Anzahl der Schleifendurchläufe von Bubblesort2 mit der Anzahl der Schleifendurchläufe von Bubblesort (erste Variante). (Hinweis: Sie müssen den Algorithmus Bubblesort (erste Variante) nicht anwenden!) **[2]**

f) Sortieren Sie die Zahlenfolge mit dem **Quicksort2**-Algorithmus. Notieren Sie alle rekursiven Aufrufe. Alternativ können Sie in der beiliegenden Tabelle in den Spalten „left“ und „right“ die entsprechenden linken und rechten Array-Indexe angeben. **[22]**

Beispiel: (die Zahlenwerte in der dritten Zeile sind vielleicht nicht korrekt…):

left	right	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	16	5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1	7	1	2	7	4	7	5	3									

In diesem Fall befinden sich in der letzten Zeile die Zahlen beim Aufruf von **`quicksort2(1,7)`**.

4 Rot-Schwarz-Baum [20]

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

5 – 4 – 7 – 10 – 6 – 8 – 9 – 11 – 12 – 20 – 21

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).

5 Allgemeine Fragen [35]

5.1 Komplexität von Algorithmen [1]

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

a. quadratische Laufzeit
b. kubische Laufzeit
c. Exponentielle Laufzeit
d. Logarithmische Laufzeit

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

5.2 Sortieralgorithmen [2]

Kreuzen Sie an, welche der folgenden Sortieralgorithmen stabil sind.
Insertionsort. Bei falsch angekreuzten Antworten werden 0,5 Punkte abgezogen; eine negative Punktsumme wird auf 0 Punkte aufgerundet.

- ○ Shellsort
- ○ Bubblesort (erste Variante)
- ○ Bubblesort2
- ○ Quicksort
- ○ Mergesort
- ○ Heapsort
- ○ Selectionsort

5 Allgemeine Fragen (Forts)

5.3 Heap [12]

Die Elemente eines Heaps sind in folgenden Array gespeichert:

9	15	13	52	38	70	64	78	73

a) Handelt es bei diesem Heap und einen Min-Heap, einen Max-Heap oder weder um einen Min-, noch um einen Max-Heap?
b) Zeichnen Sie den Heap-Baum.
c) Entfernen Sie das Minimum und stellen aus dem Rest einen Min-Heap her.
d) Stellen Sie aus dem Ergebnis von c) einen Max-Heap her. Zeichnen Sie anschließend den resultierenden Heap-Baum.

5.4 Rekursion vs. Iteration [10]

Gegeben sei folgende Java-Methode:

```
public static int rek(int a, int b) {
  if (b == 0) return a;
  else
    return rek(b, a % b);
}
```

a) Was berechnet diese Methode?

b) Wandeln Sie diese rekursive Java-Methode in eine äquivalente iterative Methode um.

5 Allgemeine Fragen (Forts)

5.5 Schleifeninvariante [10]

Gegeben sei folgender Java-Code zu Berechnung von Zweierpotenzen:

```
public static int pot2 (int n) {
  int r = 1;
  int i = 0;
  while (i < n) {
    r *= 2;
    i++;
  }
  return r;
}
```

Beweisen Sie mit vollständiger Induktion, dass die Schleifeninvariante

$r = 2^i$

lautet.

Tabelle für Shellsort

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8

Tabelle für Insertionsort

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8

Tabelle für Bubblesort2

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8

Tabelle für Quicksort2

left	right	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	16	5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8

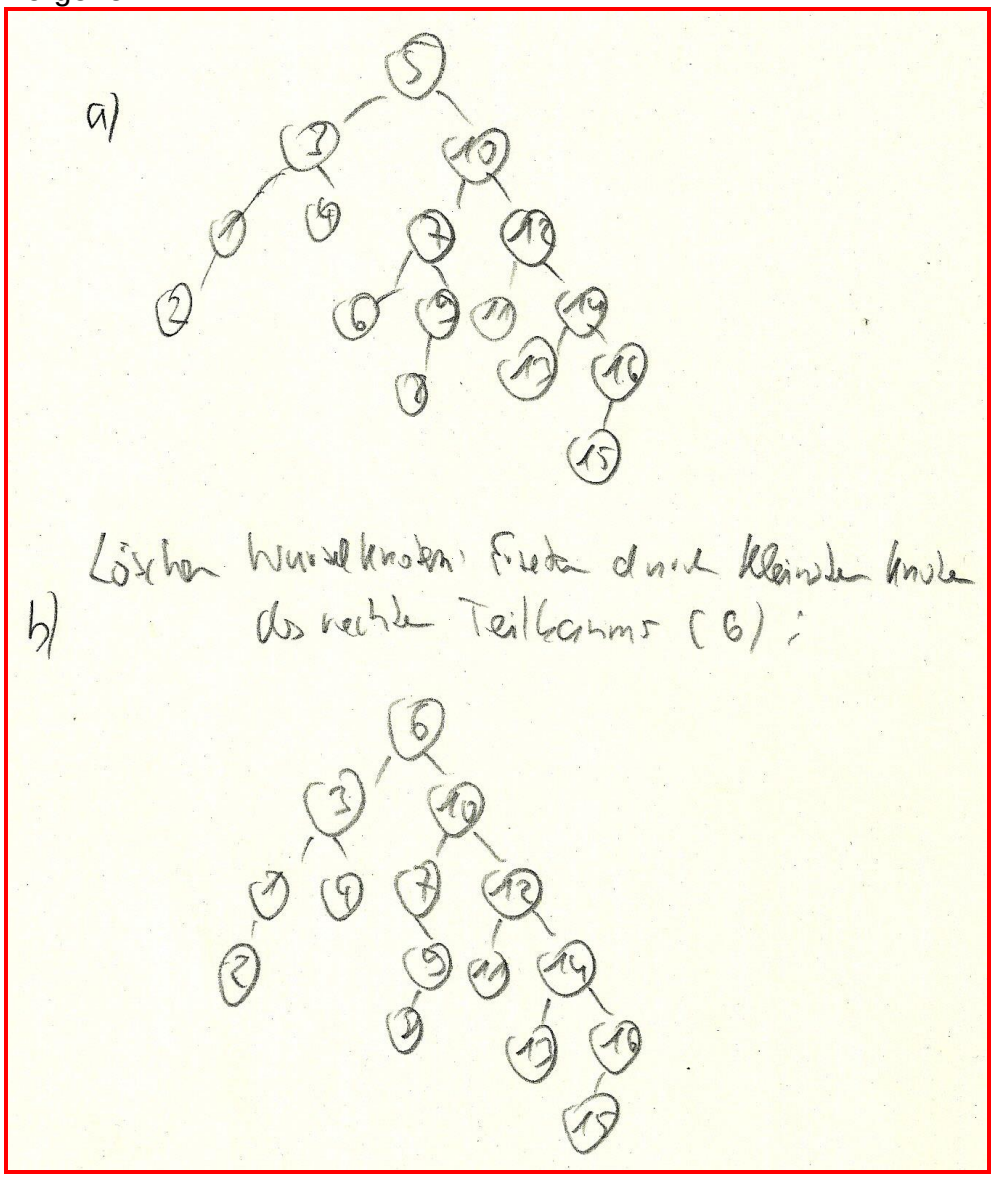
MUSTERLÖSUNGEN

1 Binäre Suchbäume [5]

a) Fügen Sie die Zahlen der folgenden Tabelle in einen anfangs leeren binären Suchbaum und zeichnen ihn.

5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8

b) Löschen Sie aus dem Suchbaum, den Sie in der obigen Aufgabe erstellt haben, den Wurzelknoten. Zeichnen Sie den resultierenden Baum und beschreiben Sie Ihr Vorgehen.



c) Geben Sie eine Einfügereihenfolge der obigen Zahlen an, damit ein Baum mit minimaler Höhe resultiert. Sie müssen diesen Baum nicht zeichnen.

Musterlösung

2 Rekursion [] - MUSTERLÖSUNG

Hinweis: Diese Aufgabe bitte direkt auf dem Blatt lösen.

Gegeben sei die folgende rekursive Java-Methode:

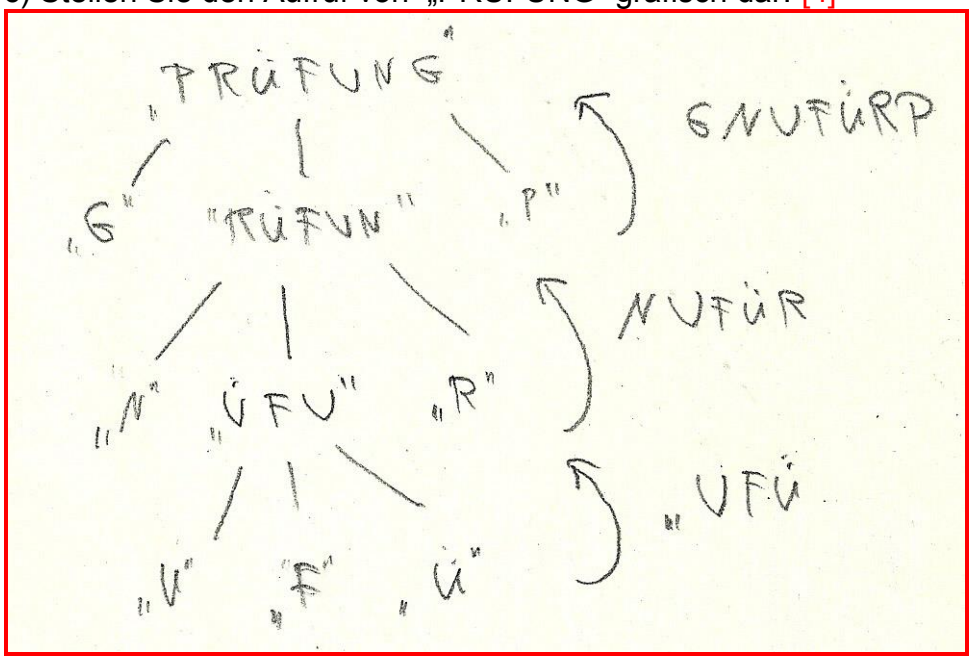
```
public static String algo1(String s) {

      if (s.isEmpty()) {
            return s;
      }
      if (s.length() == 1) {
            return s;
      }
      return s.charAt(s.length() - 1)
                  + algo1(s.substring(1, s.length() - 1))
                  + s.charAt(0);
}
```

a) Geben Sie die Rückgabewerte für folgende Eingaben an: [5]

s	**Rückgabewert**
„A“	„A“
„CCC“	„CCC“
„HALLO“	„OLLAH“
„PRÜFUNG“	„GNUFÜRP“
„ALGORITHMEN UND DS 1“	„1 SD DNU NEMHTIROGLA“

b) Was bewirkt diese Methode? [1]
Invertierung einer Zeichenkette

c) Stellen Sie den Aufruf von „PRÜFUNG“ grafisch dar. [4]



Gegeben sei folgende Zahlenfolge aus 16 Zahlen (die erste Zeile bezeichnet den Index):

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8

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

a) Sortieren Sie diese Zahlenfolge mit dem **Shellsort**. **[10]**

b) Sortieren Sie diese Zahlenfolge mit dem **Insertionsort**. **[8]**

c) Vergleichen Sie die Anzahl der Verschiebungen von Shellsort und Insertionsort. **[1]**

d) Sortieren Sie die Zahlenfolge mit dem **Bubblesort2**. **[8]**

e) Vergleichen Sie die Anzahl der Schleifendurchläufe von Bubblesort2 mit der Anzahl der Schleifendurchläufe von Bubblesort (erste Variante). (Hinweis: Sie müssen den Algorithmus Bubblesort (erste Variante) nicht anwenden!) **[2]**

f) Sortieren Sie die Zahlenfolge mit dem **Quicksort2**-Algorithmus. Notieren Sie alle rekursiven Aufrufe. Alternativ können Sie in der beiliegenden Tabelle in den Spalten „left“ und „right“ die entsprechenden linken und rechten Array-Indexe angeben. **[22]**

Beispiel: (die Zahlenwerte in der dritten Zeile sind vielleicht nicht korrekt…):

left	right	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	16	5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1	7	1	2	7	4	7	5	3									

In diesem Fall befinden sich in der letzten Zeile die Zahlen beim Aufruf von **`quicksort2(1,7)`**.

Tabelle für Shellsort

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8
	5				9				1				13			
	1				5				9				13			
		10				14				6				2		
		2				6				10				14		
			7				3				15			7	11	
			3				7				11				15	
				12				16				4				8
				4				8				12				16
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
								↓	keine Änderung (keine Tausch) mehr							

Tabelle für Insertionsort

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8
	5	7	10	12	9	14	3	16	1	6	15	4	13	2	11	8
	5	7	9	10	12	14	3	16	1	6	15	4	13	2	11	8
	3	5	7	9	10	12	14	16	1	6	15	4	13	2	11	8
	1	3	5	7	9	10	12	14	16	6	15	4	13	2	11	8
	1	3	5	6	7	9	10	12	14	16	15	4	13	2	11	8
	1	3	5	6	7	9	10	12	14	15	16	4	13	2	11	8
	1	3	4	5	6	7	9	10	12	14	15	16	13	2	11	8
	1	3	4	5	6	7	9	10	12	13	14	15	16	2	11	8
	1	2	3	4	5	6	7	9	10	12	13	14	15	16	11	8
	1	2	3	4	5	6	7	9	10	11	12	13	14	15	16	8
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

1 1
11 2
111 5
1111 8
11 6
1 1
10
3
12
5
8
6 IV.

Tabelle für Bubblesort2

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8
	5	7	10	9	12	3	14	1	6	15	4	13	2	11	8	16
2	5	7	9	10	3	12	1	6	14	4	13	2	11	8	15	16
3	5	7	9	3	10	1	6	12	4	13	2	11	8	14	15	16
4	5	7	3	9	1	6	10	4	12	2	11	8	13	14	15	16
5	5	3	7	1	6	9	4	10	2	11	8	12	13	14	15	16
6	3	5	1	6	7	4	9	2	10	8	11	12	13	14	15	16
7	3	1	5	6	4	7	2	9	8	10	11	12	13	14	15	16
8	1	3	5	4	6	2	7	8	9	10	11	12	13	14	15	16
9	1	3	4	5	2	6	7	8	9	10	11	12	13	14	15	16
10	1	3	4	2	5	6	7	8	9	10	11	12	13	14	15	16
11	1	3	2	4	5	6	7	8	9	10	11	12	13	14	15	16
12	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
13	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

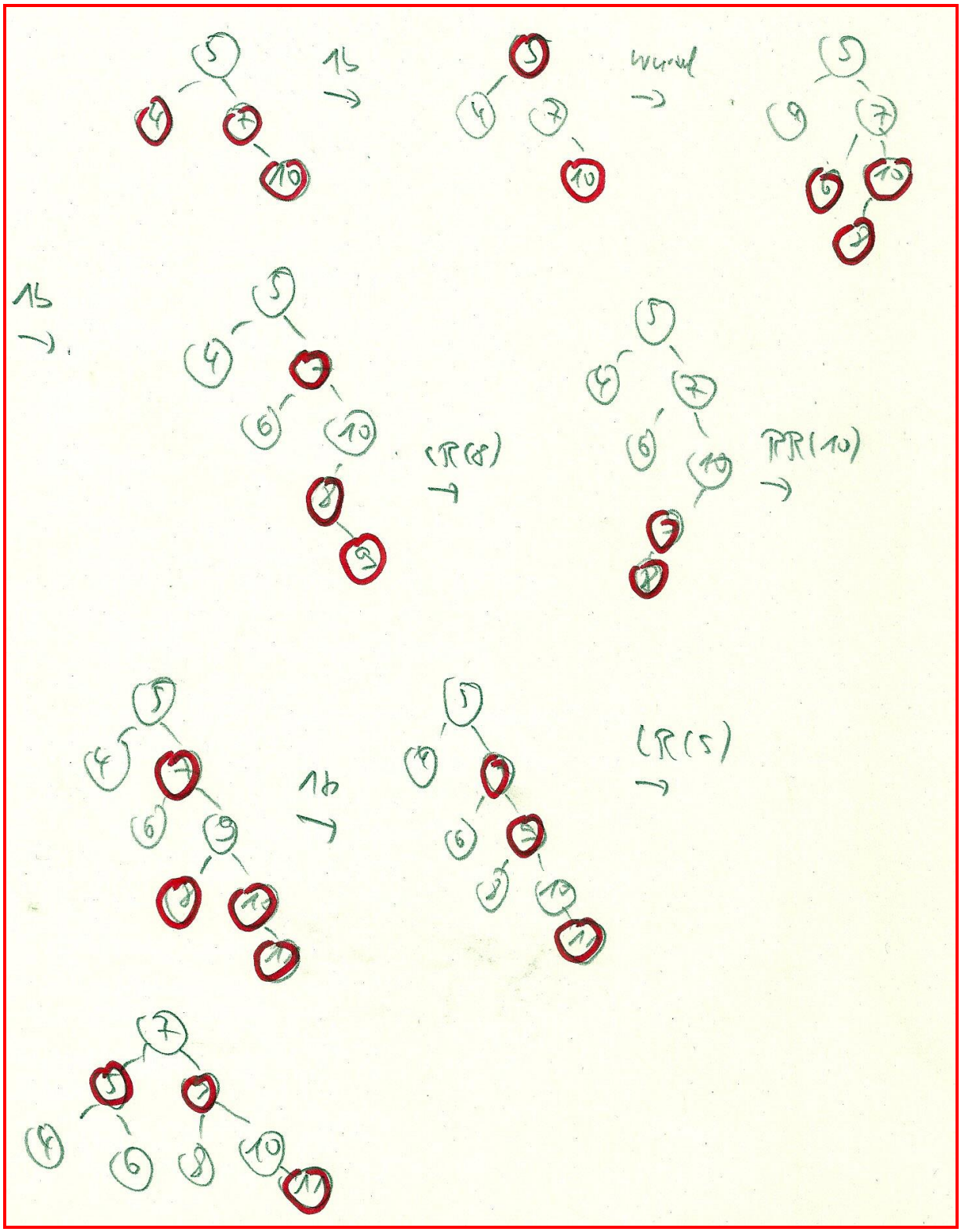
Tabelle für Quicksort2

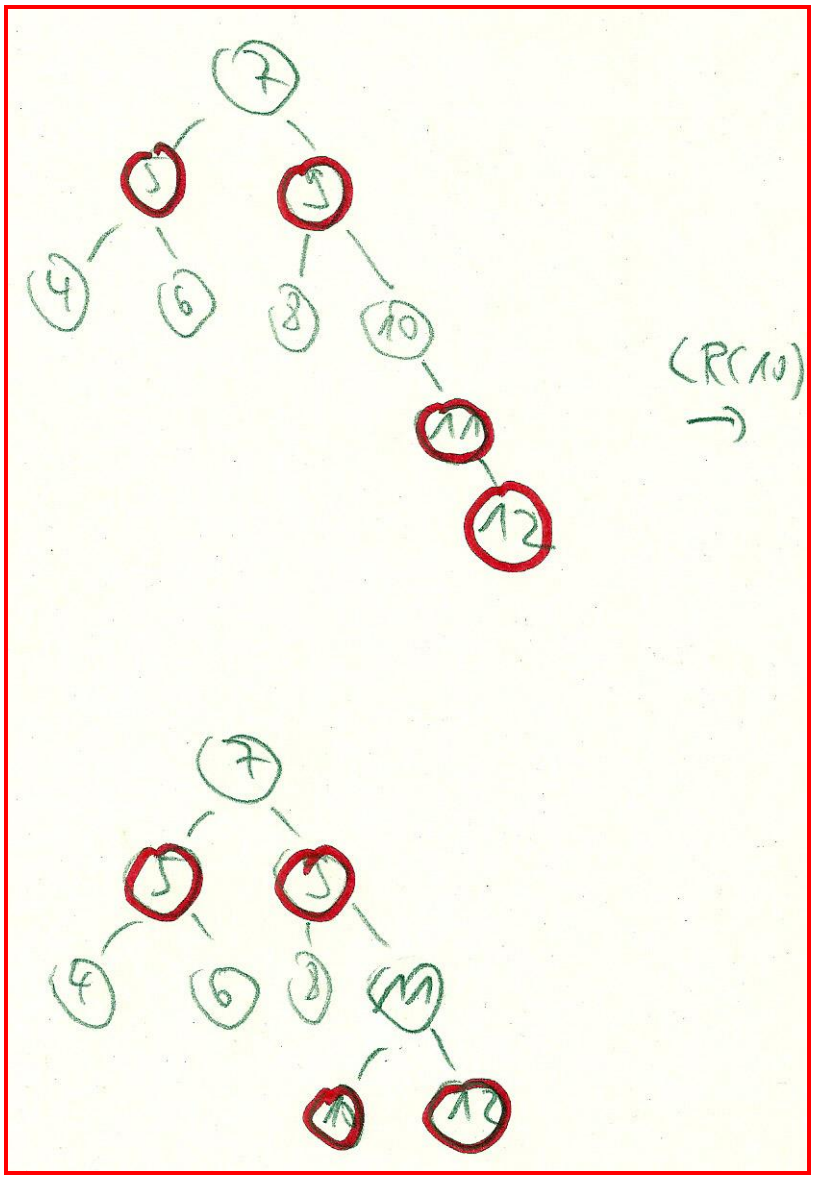
left	right	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
1	16	5	10	7	12	9	14	3	16	1	6	15	4	13	2	11	8	
		5	2	7	12	9	14	3	16	1	6	15	4	13	10	11	8	
		5	2	7	4	9	14	3	16	1	6	15	12	13	10	11	8	
		5	2	7	4	6	14	3	16	1	9	15	12	13	10	11	8	
		5	2	7	4	6	1	3	16	14	9	15	12	13	10	11	8	swap
		5	2	7	4	6	1	3	8	14	9	15	12	13	10	11	16	(14,16), (9,16)
1	7	5	2	7	4	6	1	3										
		1	2	7	4	6	5	3										swap
		1	2	3	4	6	5	7										(12)-(4,7)
1	2	1	2															✓
4	7				4	6	5	7										(4,6), (5,7)
4	6				4	5	6											swap
					4	5	6											(4,4), (6,6)
9	16									14	9	15	12	13	10	11	16	(9,15), (13,14)
9	15									14	9	15	12	13	10	11		
										10	9	15	12	13	14	11		swap
										10	9	11	12	13	14	15		(9,10), (11,15)
9	10									10	9							swap
										9	10							(9,9), (10,10)
11	15											11	12	13	14	15		(11,14), (15,15)
11	14											11	12	13	14			(11,13), (14,14)
11	13											11	12	13				(11,12), (13,13)
11	12											11	12					(11,11), (12,12)

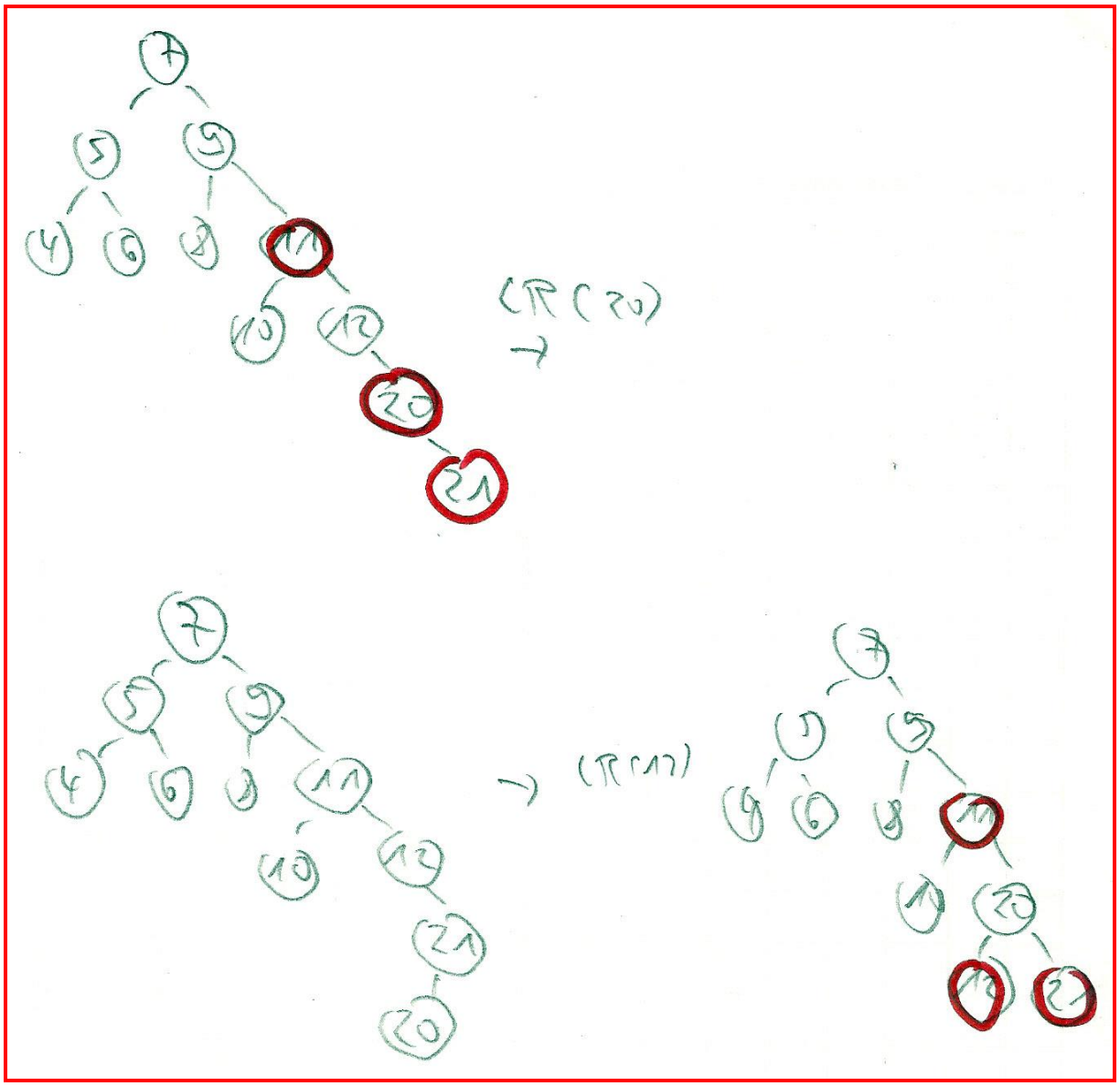
4 Rot-Schwarz-Baum [20]
Fügen Sie in einen anfangs leeren Rot-Schwarz-Baum nacheinander Knoten mit folgenden Schlüsseln ein:

5 – 4 – 7 – 10 – 6 – 8 – 9 – 11 – 12 – 20 – 21
1 – 1 – 1 – 02 – 1 – 3 – 3 – 02 – 02 – 02 – 03 (Punkte)

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).


15
wurde
15
LR(8)
RR(10)
16
LR(5)


7
5
9
4
6
8
10
11
12
RR(10)
7
5
9
4
6
8
11



Musterlösung

5.1 Komplexität von Algorithmen [1]

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

a. quadratische Laufzeit
b. kubische Laufzeit
c. Exponentielle Laufzeit
d. Logarithmische Laufzeit

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

5.2 Sortieralgorithmen

Kreuzen Sie an, welche von den folgenden Sortieralgorithmen stabil sind. []

Insertionsort

- Shellsort
- **Bubblesort (erste Variante)**
- **Bubblesort2**
- Quicksort
- **Mergesort**
- Heapsort
- Selectionsort

5.4 Rekursion vs. Iteration

Wandeln Sie folgenden Java-Code in ein äquivalentes iterative Programm um.

```
public static int rek(int a, int b) {
  if (b == 0) return a;
  else
    return rek(b, a % b);
}
```

```
public static int iterativ(int a, int b) {
  int rest = 0;
  while (b != 0) {
    rest = a % b;
    a = b;
    b = rest;
  }
  return a;
}
```

Gegeben sei folgender Java-Code zu Berechnung von Zweierpotenzen:

```
public static int pot2 (int n) {
  int r = 1;
  int i = 0;
  while (i < n) {
    r *= 2;
    i++;
  }
  return r;
}
```

Beweisen Sie mit vollständiger Induktion, dass die Schleifeninvariante

$r = 2^i$

lautet.

Schleifeninvariante - Beweis

Induktionsanfang:

Vor Beginn des ersten Schleifendurchlaufs:

$r_1 = 2^{i_1}$ $(i_1 = 0) \rightarrow r_1 = 2^0 = 1$ ✓

Induktionsannahme:

Beim n-ten Durchlauf gelte bei Schleifeneintritt:

$r_n = 2^i$

Induktionsschluss $n \rightarrow n+1$:

Es gelten folgende Zusicherungen (Anweisungen):

(1) $r_n = 2^{i_n}$ (1. Zeile der Schleife)

(2) $r_{n+1} = r_n * 2$ (2. Zeile)

(3) $i_{n+1} = i_n + 1$ (3. Zeile)

$\rightarrow r_{n+1} = \underbrace{r_n}_{(2)} * 2 = \underbrace{2^{i_n}}_{(1)} * 2 = \underbrace{2^{i_n+1}}_{(3)} = 2^{i_{n+1}}$ ✓

Terminierung:

Solange $n < \infty$

i wird in jedem Schleifendurchlauf erhöht,

bis die obere Grenze n erreicht ist. ✓