

## Klausur Algorithmen

Name: \_\_\_\_\_

Matrikelnummer: \_\_\_\_\_

Aufgabe	Erreichte Punktzahl
1 ( 5 Punkte)	2,0
2 ( 10 Punkte)	6,0
3 ( 6 Punkte)	4,0
4 ( 6 Punkte)	2,0
5 ( 8 Punkte)	4,5
6 ( 11 Punkte)	7,5
7 ( 10 Punkte)	8,0
8 ( 7 Punkte)	5,5
9 ( 13 Punkte)	5,0
Summe 76 Punkte	44,5

Bearbeitungszeit: 90 Minuten

Note: 3,3 Rip. 16.07.12

Gesamtnote: \_\_\_\_\_

Bitte kreuzen Sie an, wenn einer der beiden nachfolgenden Fälle auf Sie zutrifft:

Letzter Versuch

Die Klausur ist bestanden, wenn mindestens 38 Punkte erreicht werden.

## 1. Aufgabe (5 Punkte)

(2)

Betrachten Sie die nachfolgend aufgeführte Funktion **mysterious**.

---

```

1: static int mysterious(int n) {
2:
3:     int i = 1;
4:     int j = 0;
5:
6:     while (i < n) {
7:         i *= 2;
8:         j++;
9:     }
10:    return j;
11: }
```

---

- Schätzen Sie die Worst-Case-Laufzeit der Funktion mit Hilfe der O-Notation in Abhängigkeit von dem Parameter  $n$  möglichst genau ab.
- Gibt es einen Unterschied zwischen der Worst-Case-Laufzeit und der Best-Case-Laufzeit? Wenn ja, geben Sie die Best-Case-Laufzeit an!

Begründen Sie jeweils Ihre Aussagen!

noch 1

while ~~hat die Laufzeit  $\Theta(n)$~~  hat Laufzeit von  $\frac{n}{2}$  zu ungenau  
 Es gibt keine Unterschied zwischen best-Case und Worst-Case. ✓ 1

*Was ist mit den anderen Anweisungen?*

## 2. Aufgabe (10 Punkte)

(6)

a) Geben Sie an, welche der nachfolgenden Aussagen wahr, welche falsch sind. Für falsche Antworten wird jeweils 1 Punkt abgezogen!

- $\log n \in \Omega(\log n^2)$  falsch wahr f. -1
  - $n^n \in O(n!)$  falsch<sup>v</sup>, da  $n^n$  schneller wächst als  $n!$  1,5
  - $12 \cdot n^3 + 3 \cdot n^4 \in \Omega(n^3)$  wahr ✓ 1
  - $2^n + 4 \cdot n^2 \in \Theta(n^2)$  falsch<sup>v</sup>, da  $2^n$  schneller wächst als  $n^2$  1,5
  - $n^{\log(n)} \in O(n^{\sqrt{n}})$  wahr ✓ mit der Annahme, dass  $n^{\sqrt{n}}$  schneller wächst als  $n^{\log(n)}$  1,5  
 $\sqrt{n}$
- $\Sigma 4,5$

b) Geben Sie eine möglichst einfache, scharfe Funktion  $g(n)$  an, so dass  $f(n) \in \Omega(g(n))$  mit  $f(n) = 4^n + 5 \cdot n!$

$$g(n) = 4^n \quad \checkmark$$

1,5

c) Geben Sie eine möglichst einfache, scharfe Funktion  $k(n)$  an, so dass  $h(n) \in O(k(n))$  mit  $h(n) = 5/2 \cdot n \cdot \log n + 3 \cdot n/\sqrt{n}$

$$k(n) = \log n \quad f. \quad \text{welches wächst schneller?} \quad 0$$

**3. Aufgabe (6 Punkte)**

(4)

Stellen Sie den Verlauf des Aufteilungsschrittes von Quicksort (ohne die nachfolgenden Rekursionen) für folgende Zahlenfolgen dar. Dabei soll als Pivotelement das Element am rechten Rand gewählt werden. Stellen Sie die Zahlenfolgen nach jedem Austauschschritt dar.

- a) 1, 5, 19, 17, 6, 9, 7, 10

~~1, 5, 19, 17, 6, 9, 7, 10~~

1, 5, 7, 17, 6, 9, 19, 10 ✓

1, 5, 7, 9, 6, 17, 19, 10 ✓

1, 5, 7, 9, 6, 10, 17, 19 f.

- 1

- b) 5, 6, 7, 9, 4

4, 6, 7, 9, 5 ✓

Ende

- c) 5, 4, 1, 6

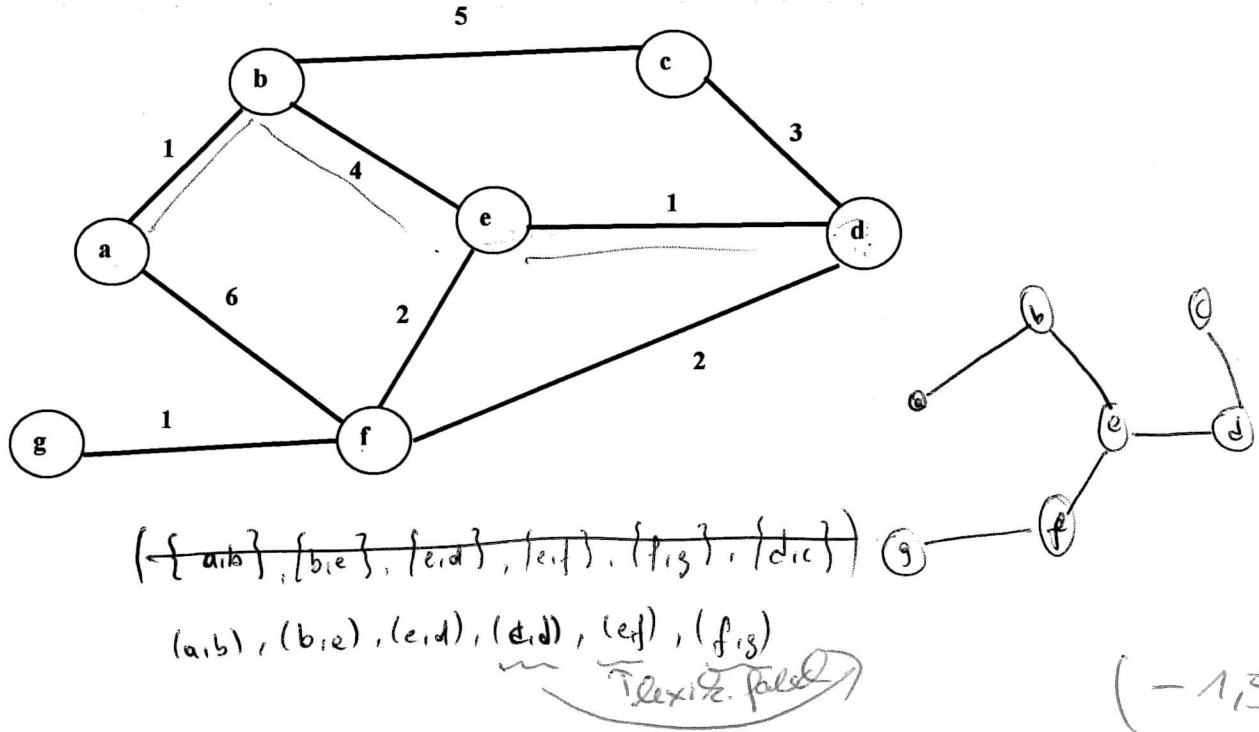
da es schon sortiert ist 2  
worst case

- 1

## 4. Aufgabe (6 Punkte) (2)

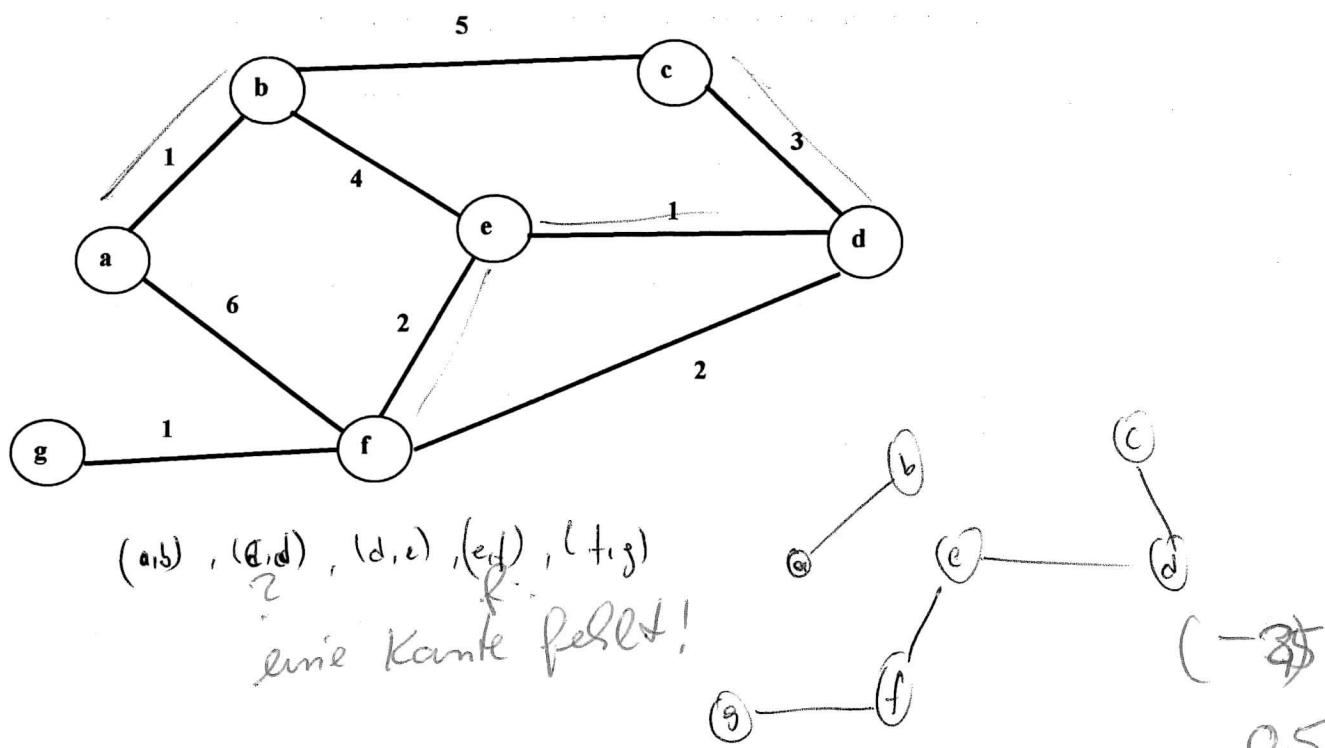
a) Bestimmen Sie mit **Prims Algorithmus** einen minimal spannenden Baum.

Starten Sie dabei bei Knoten **a**. Geben Sie die Reihenfolge an, in der die Kanten des minimal spannenden Baumes ausgewählt werden. Sind bei der Wahl der nächsten Kante aufgrund gleicher Kantengewichte mehrere Kanten möglich, so soll zuerst die lexikographisch kleinste Kante ausgewählt werden. Zeichnen Sie den minimal spannenden Baum ein.



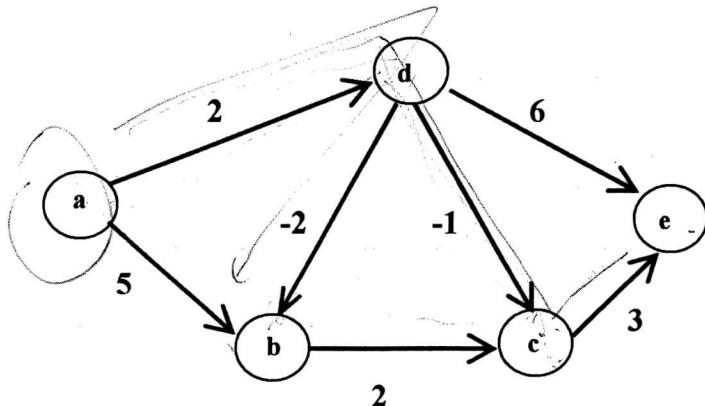
b) Bestimmen Sie mit **Kruskals Algorithmus** einen minimal spannenden Baum.

Geben Sie dabei die Reihenfolge an, in der die Kanten des minimal spannenden Baumes ausgewählt werden. Bei Auswahlmöglichkeit zwischen mehreren Kanten soll wiederum die lexikographisch kleinste Kante ausgewählt werden. Zeichnen Sie den minimal spannenden Baum ein.



## 5. Aufgabe (8 Punkte)

4,5



Betrachten Sie den oben aufgeführten Graph. Stellen Sie die ersten Phasen des Verlaufs des **Bellman-Ford-Algorithmus** zur Bestimmung der kürzesten Wege dar. Der Startknoten, d.h. der Knoten von dem aus die kürzesten Wege bestimmt werden sollen, sei der Knoten **a**. Geben Sie die **pred**- und **dist**-Werte nach jedem Durchlauf des Rumpfes der äußeren **for**-Anweisung an, d.h. nachdem alle Kanten des Graphen einmal betrachteten wurden. Durchlaufen Sie dabei die Kanten in lexikographischer Reihenfolge.

Es sind nur die Initialisierung und die Werte nach den ersten beiden Durchläufen durch die äußere **for**-Anweisung anzugeben.

**Initialisierung**

	a	b	c	d	e
pred	∅	∅	∅	∅	∅
dist	0	∞	∞	∞	∞

→ man erkennt kaum den Unterschied zw. 0 u. ∞

**Werte nach 1. Durchlauf äußere for-Anweisung**

	a	b	c	d	e
pred	∅	a	b	a	c
dist	0	5	7	2	10

1,5

1,5

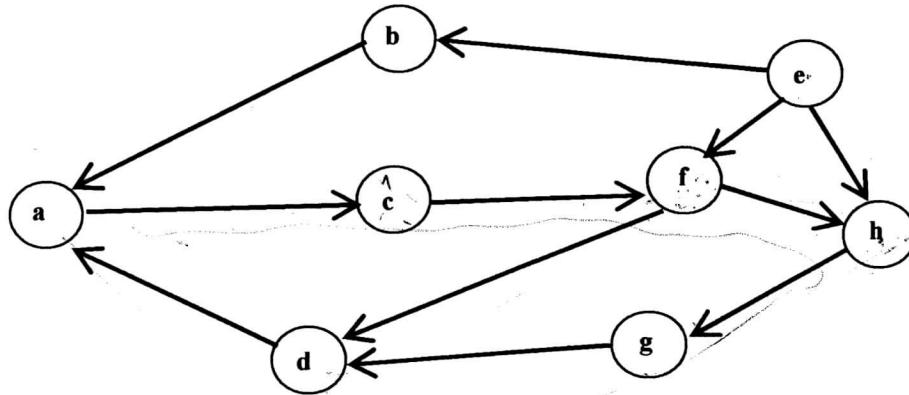
**Werte nach 2. Durchlauf äußere for-Anweisung**

	a	b	c	d	e
pred	—	a	a	a	c
dist	0	0	1	2	4

1,5

## 6. Aufgabe (11 Punkte)

Betrachten Sie nachfolgenden Graphen:



- a) Führen Sie eine **Breitensuche** startend bei Knoten **a** durch. Bei Auswahlmöglichkeit zwischen mehreren Knoten ist immer der lexikographisch kleinste zu wählen.

	a	b	c	d	e	f	g	h
pred	-	e	a	f	-	c	h	f
dist	1	f	1	3	2	2	4	3

- b) Führen Sie eine **Tiefensuche** startend mit Knoten **a** durch. Bei der mehreren Auswahlmöglichkeiten eines Knotens wählen Sie immer den Knoten mit der lexikographisch kleinsten Beschriftung. Ermitteln Sie die **first**-, **last**- und **pred**-Werte und tragen Sie sie in nachfolgender Tabelle ein:

	a	b	c	d	e	f	g	h
first	1 ✓	14	2 ✓	4 ✓	13	3 ✓	10 f	9 f
last	5 f	15	6 f	8 f	16	7 f	11 f	12 f
pred	d f	e	a ✓	f ✓	-	c ✓	h ✓	f ✓

- c) Bestimmen Sie eine topologische Sortierung des Graphen, sofern dies möglich ist. Wenn dies nicht möglich sein sollte, begründen Sie dies.

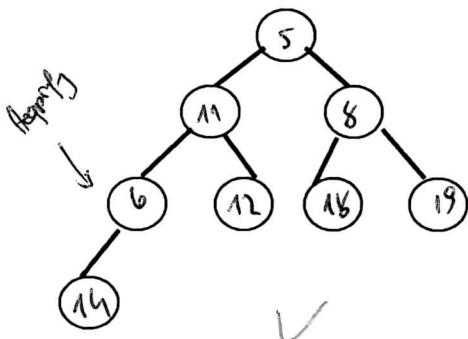
Nicht topologisch sortierbar weil der Graph nicht  
acyklisch ist also er hat  
ein Zyklus.

## 7. Aufgabe (10 Punkte)

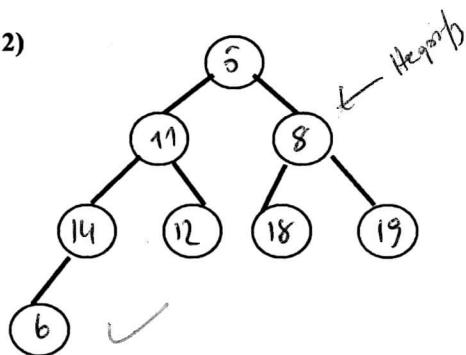
(8)

- a) Stellen Sie den Verlauf des Verfahrens **BuildHeap** zum Aufbau eines Heaps für die Zahlenfolge 5, 11, 8, 6, 12, 18, 19, 14 mit Hilfe eines Binärbaumes dar. In diesem Verfahren wird für bestimmte Knoten das Verfahren **Heapify** durchgeführt. Geben Sie zur Darstellung des Verfahrens jeweils an, auf welchen Knoten Sie **Heapify** anwenden und anschließend das Ergebnis des vollständigen Durchlaufs (inklusive aller Rekursionen) von **Heapify**.

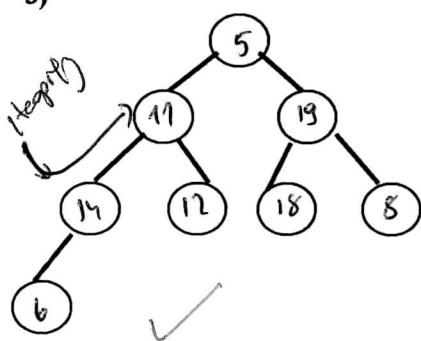
1)



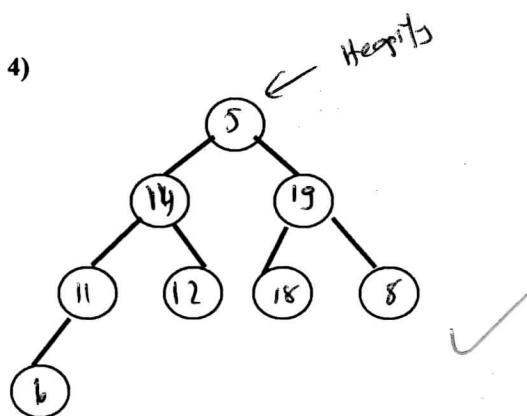
2)



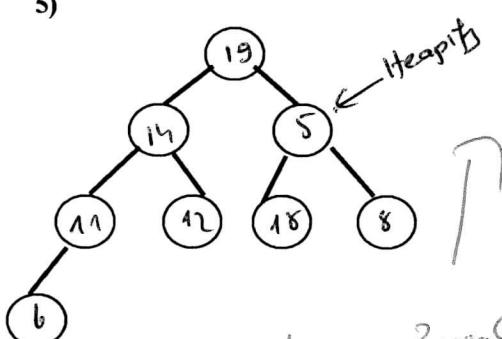
3)



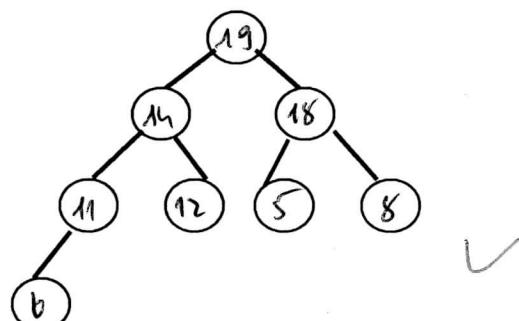
4)



5)



6)



diese Zwischen-  
schritt sollte nicht  
angegeben werden!

-0,5

b) Stellen Sie alle möglichen Heaps bestehend aus den Werten 1, 2, 3, 4 dar.



ein Heap fehlt! wegen falsche Heaps! -0,5  
-1

**8. Aufgabe (7 Punkte)**

5,5

Tragen Sie in nachfolgenden Hashtabellen die Schlüssel ein, wenn als Hashverfahren die Divisionsrestmethode angewendet wird und als Sondierungsverfahren

- Lineares Sondieren
- Quadratisches Sondieren
- Verkettung Überläufer

Dabei sollen folgende Schlüssel in der angegebenen Reihenfolge eingefügt werden:  
10, 21, 17, 8, 9, 3.

Geben Sie zu diesem Zweck die Hashfunktion an, sowie für jeden Schlüssel den Wert, auf den er gemäß der Hashfunktion abgebildet wird.

**Lineares Sondieren**

0	1	2	3	4	5	6
21	17	8	10	9	3	

$$m = 7$$

$$h(k) = k \bmod 7$$

$$h(10) = 3$$

$$h(21) = 0$$

$$h(17) = 1 \quad f$$

$$h(8) = 1 \quad \checkmark$$

$$h(9) = 2 \quad \checkmark$$

$$h(3) = 3 \quad \checkmark$$

**Quadratisches Sondieren**

0	1	2	3	4	5	6
21	17	8	10	9	3	

(v)

**Verkettung Überläufer**

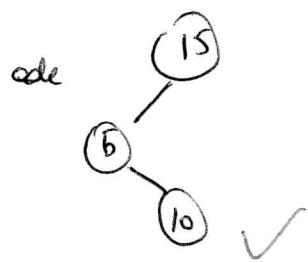
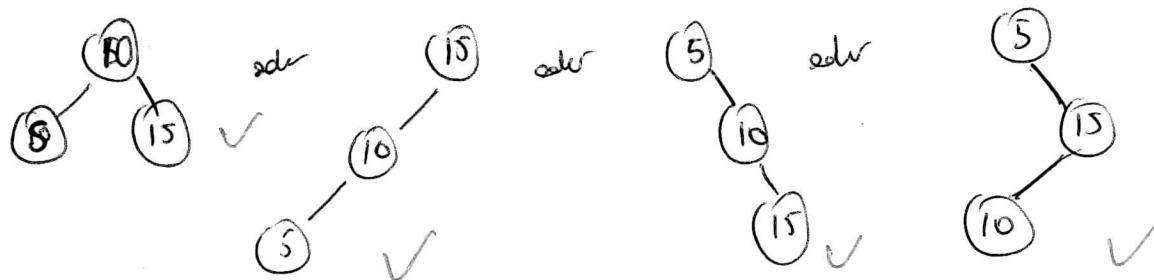
0	1	2	3	4	5	6
21	17	8	10			

(v)

**9. Aufgabe (13 Punkte)**

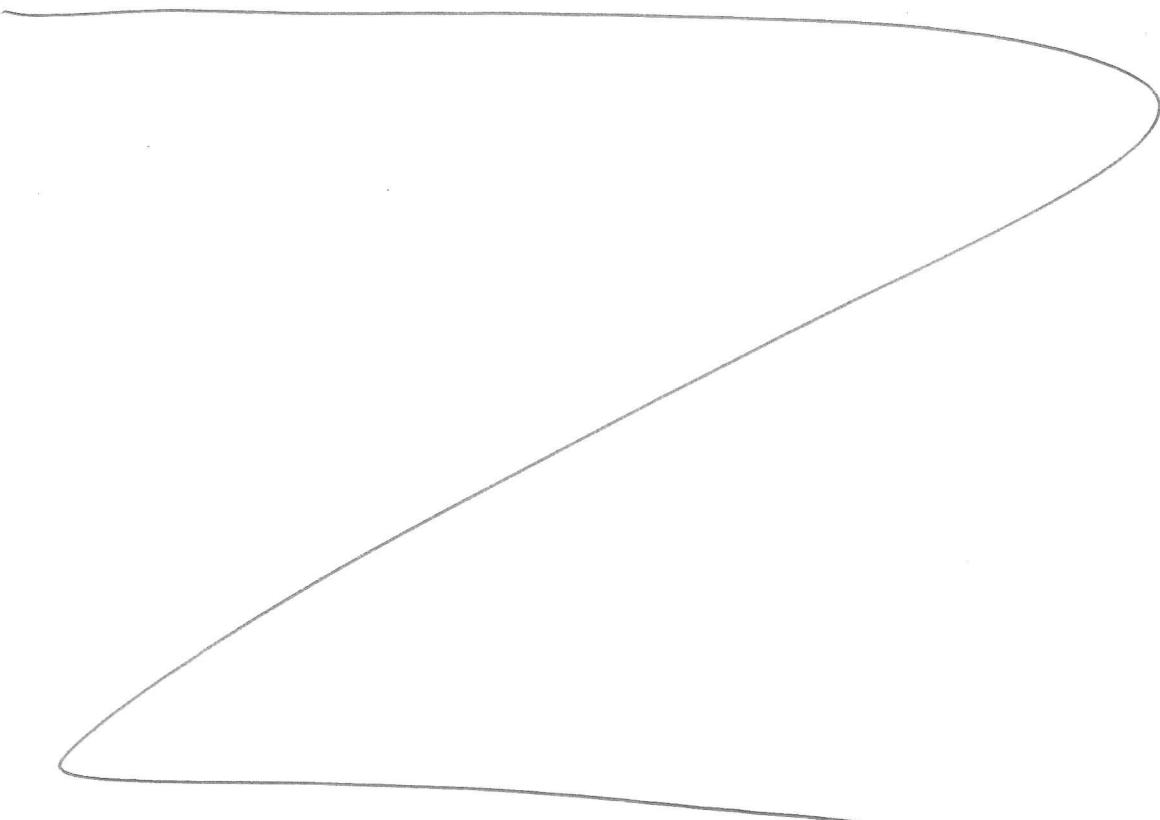
(5)

a) Geben Sie für die Schlüsselwerte {5, 10, 15} alle möglichen Suchbäume an!

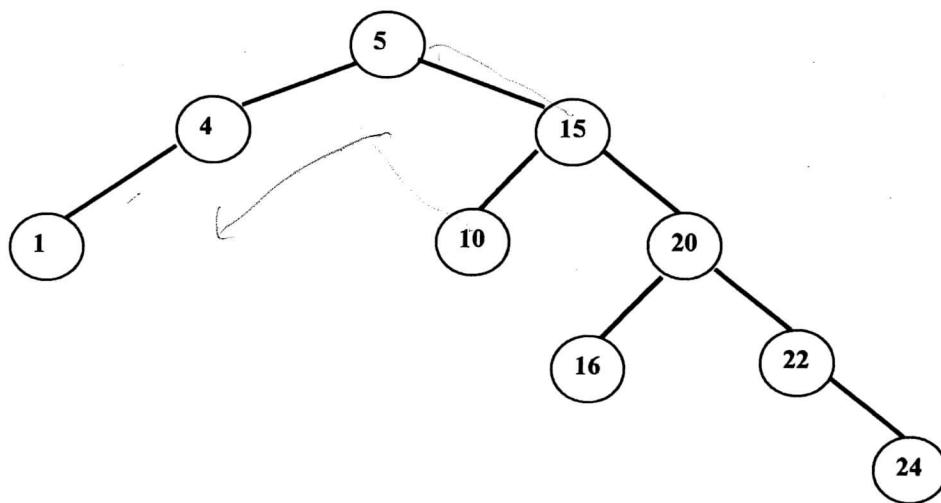


4

b) Geben Sie ein möglichst effizientes Verfahren an, um in einem Suchbaum den kleinsten Wert zu finden. Geben Sie die Laufzeit im worst-case und im best-case in Abhängigkeit von der Anzahl der Schlüsselwerte (Knoten) an. Skizzieren Sie die Situation, in der der worst-case und in der der best-case eintritt.



c) Betrachten Sie nachfolgenden Suchbaum.



Führen Sie nun eine Linksrotation um den Knoten mit Wert 5 aus und zeichnen Sie den entstandenen Suchbaum auf.

