21

Klausur Algorithmen

Name:_

Matrikelnummer: _

Aufgabe	Erreichte Punktzahl
1 (6 Punkte)	4
2 (9 Punkte)	0
3 (8 Punkte)	8
4 (6 Punkte)	6
5 (12 Punkte)	8,5
6 (15 Punkte)	12,5
7 (8 Punkte)	8
8 (8 Punkte)	6
9 (8 Punkte)	4
Summe 80 Punkte	57,0

Bearbeitungszeit: 90 Minuten

Note: 2,7

Gesamtnote: 2,3 Rip. 20.07.08

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

- ☐ **Letzter Versuch**
- ☐ **Zeitablauf**

Die Klausur ist bestanden, wenn mindestens 40 Punkte erreicht werden.

1. Aufgabe (6 Punkte) 4

Betrachten Sie nachfolgenden Algorithmus in Pseudocode:

```
// G = (V, E) sei Graph mit n Knoten und m Kanten
// S sei Stack mit Knoten
for (v ∈ V)
        S.push(v);

while (S != leer) {
        v = S.pop();
        markiere v als besucht;
        for (u ∈ Adj(v)) // fuer alle Nachbarn von v
                Gebe u aus
}
```

→ $O(V)$ Durchläufe ✓ 1

→ $O(n)$ Durchläufe ✓ 1

→ $O(n)$ Durchläufe zu groß noch 1

Analysieren Sie die Laufzeit unter der Annahme, dass die Stackoperationen optimal realisiert werden und der Graph geeignet durch Adjazenzlisten realisiert ist!
Schätzen Sie dazu mit Hilfe der O-Notation die Worst-Case-Laufzeit der Funktion in Abhängigkeit von der Problemgröße (Anzahl Kanten und Knoten) ab.
Begründen Sie Ihre Aussage.

Da for-Schleife in while-Schleife verschachtelt ist, beträgt die Laufzeit $O(n) + O(n \cdot n) = \underline{O(n^2)}$ im Worst-Case. (✓) 1

Stackop. ?

2. Aufgabe (9 = 6 + 3 Punkte) 0

a) Geben Sie an, welche der nachfolgenden Aussagen wahr, welche falsch sind. Für falsche Antworten werden jeweils 2 Punkte abgezogen!

- `floor(`$\sqrt{n}$`)` $\in$ `O(n )`, wobei `floor(x)` die kleinste ganze Zahl `>=` `x` ist ✓ wahr ✓
- $\log^2(n) \in \Theta(\log(n))$ ✓ wahr f.
- $4\ n^2 + \sqrt{n} \in O(n^2)$ ✓ wahr ✓
- $\frac{1}{3}\ n + \log(n) \in \Omega(n)$ ✗ falsch f.
- $n^{\log(n)} \in O(n^2)$ ✗ falsch ✓
- $n! \in \Omega(n^n)$ ✗ falsch ✓

Σ 0

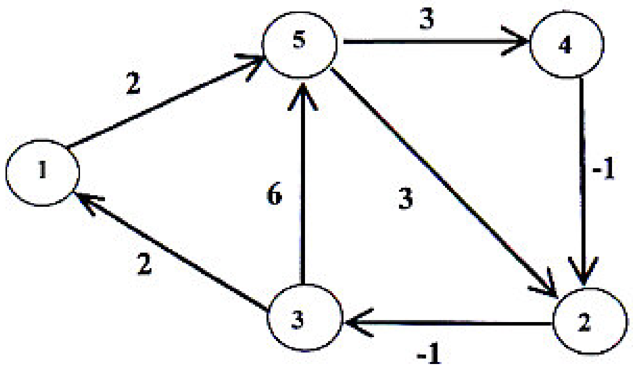
b) Geben Sie eine möglichst einfache, scharfe Funktion `g(n)` an, so dass $f(n) \in O(g(n))$ mit $f(n) = 3 \log n + \frac{1}{2}\sqrt{n}$.

$g(n) = \log n$ f.

c) Geben Sie eine möglichst einfache, scharfe Funktion `k(n)` an, so dass $h(n) \in \Omega(k(n))$ mit $h(n) = 3n! + 4n^2$.

$k(n) = n^2$ f.

3. Aufgabe (8 Punkte) 8



Betrachten Sie den oben aufgeführten Graphen.
Stellen Sie die ersten Schritte des Floyd-Warshall-Algorithmus zur Bestimmung aller kürzesten Wege dar. Füllen Sie dazu die nachfolgenden Matrizen mit der Initialisierung und den **ersten beiden** Durchläufen der äußeren **for-k**-Anweisung aus.

Initialisierung

$DIST^0$	1	2	3	4	5
1					2
2			-1		
3	2				6
4		-1			
5		3		3	

$PRED^0$	1	2	3	4	5
1					1
2			2		
3	3				3
4		4			
5		5		5	

nicht eingetragene: ∞

nicht eingetragene: null

kürzeste Wege

$DIST^1$	1	2	3	4	5
1					2
2			-1		
3	2				4
4		-1			
5		3		3	

$PRED^1$	1	2	3	4	5
1					1
2			2		
3	3				1
4		4			
5		5		5	

$DIST^2$	1	2	3	4	5
1					2
2			-1		
3	2				4
4		-1	-2		
5		3	2	2	

$PRED^2$	1	2	3	4	5
1					1
2			2		
3	3				1
4		4	2		
5		5	2	5	

4. Aufgabe (6 Punkte) 6

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) 3, 7, 24, 21, 8, 11, 9, 10

b) 7, 3, 9, 8, 2

c) 8, 3, 2, 9

a) 3, 7, 9, 21, 8, 11, 24, $\boxed{10}$ ✓

3, 7, 9, 8, 21, 11, 24, $\boxed{10}$ ✓

3, 7, 9, 8, $\boxed{10}$, 11, 24, 21 ✓

b) $\boxed{2}$, 3, 9, 8, 7 ✓

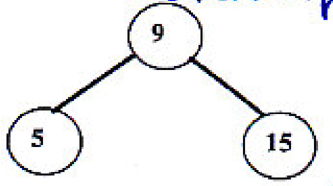
c) bleibt gleich: 8, 3, 2, $\boxed{9}$ ✓

5. Aufgabe (12 Punkte) 8,5

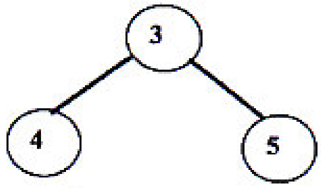
a) Betrachten Sie nachfolgende binäre Bäume, geben Sie zu jedem Baum an, ob er die Suchbaumeigenschaft, die Heapeigenschaft, beide Eigenschaften oder keine der beiden Eigenschaften hat: 4

Element linker NV: kleiner
rechter NV: größer

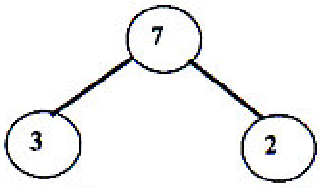
Knotenwert höchstens so groß wie Wert des Vorgängers



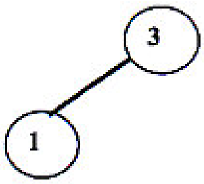
Suchbaum ✓



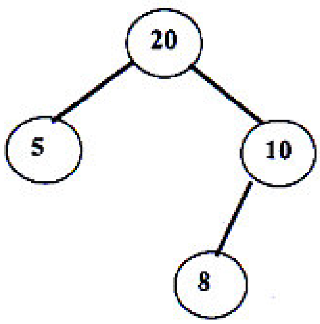
keine ✓



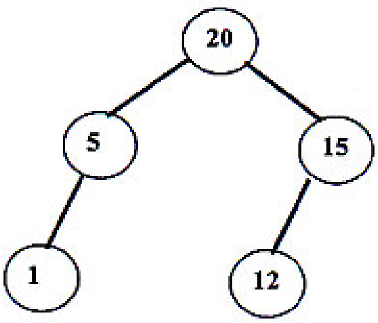
Heap ✓



Suchbaum
+ Heap ✓

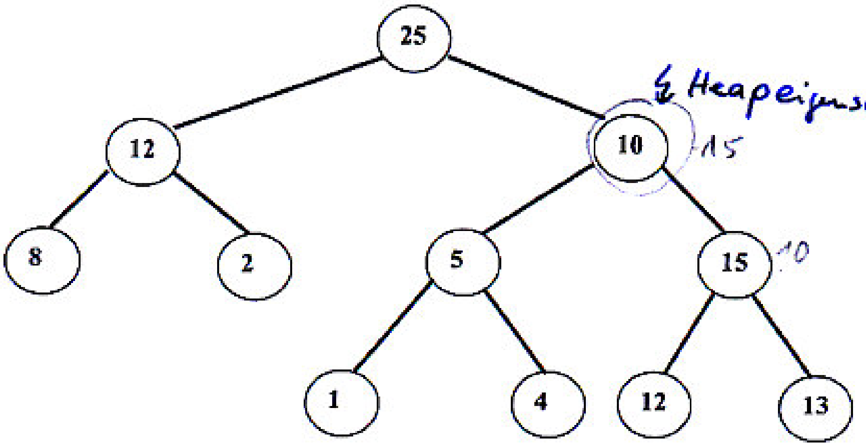


Heap f

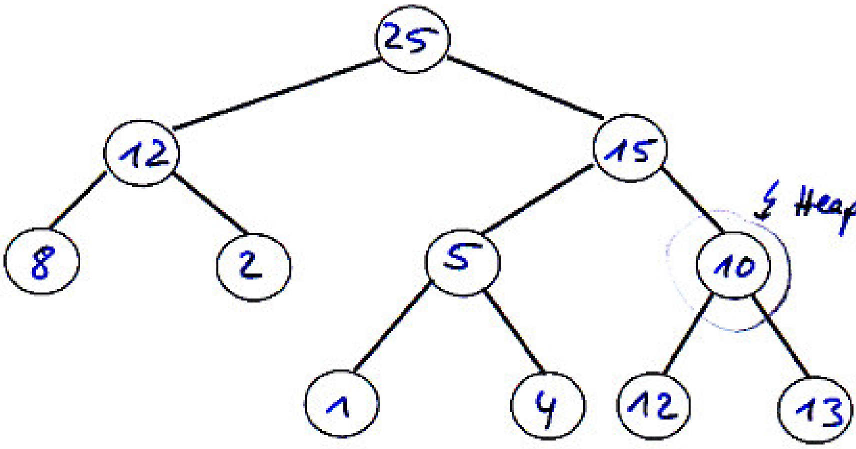


Heap f.

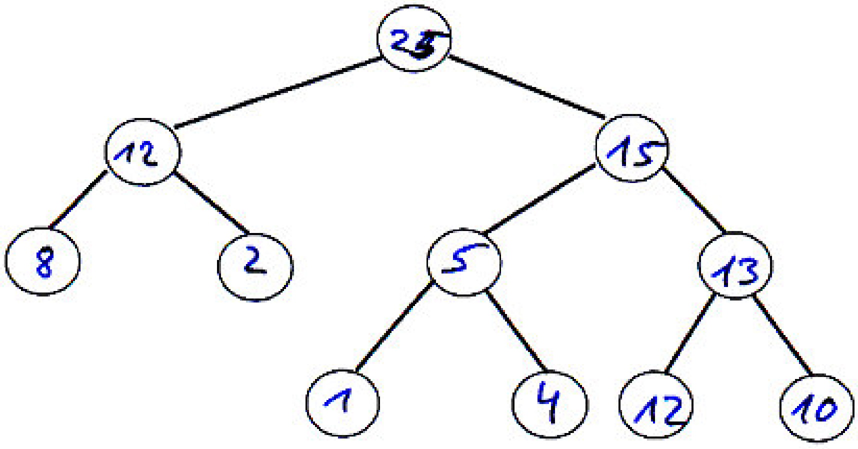
b) Geben Sie in nachfolgendem Binärbaum an, in welchem Knoten die Heapeigenschaft verletzt ist und stellen Sie mittels des Verfahrens `Heapify` einen korrekten Heap her. Stellen Sie den entstandenen Baum nach jedem Austauschschritt dar. 3



↯ Heapeigenschaft verletzt



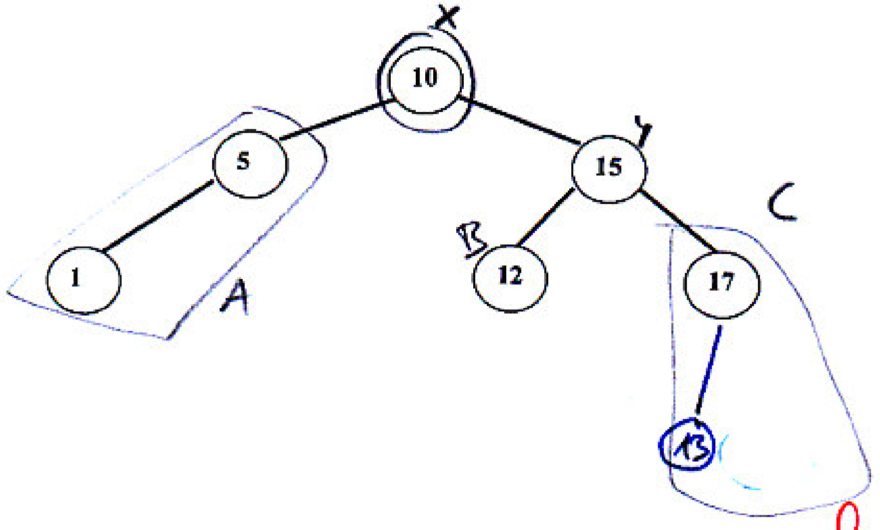
↯ Heapeigenschaft verletzt



Heap.

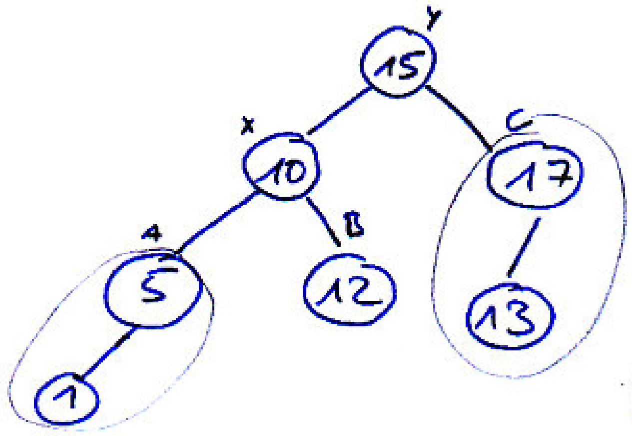
c) Betrachten Sie nachfolgenden Suchbaum. Fügen Sie in dem Baum den Knoten 13 ein (ohne den Baum zu balancieren).

1,5



f.

Führen Sie nun in dem oben aufgeführten Baum (mit dem neu eingefügten Knoten) um den Knoten 10 eine Linksrotation durch und geben Sie den entstandenen Suchbaum an.



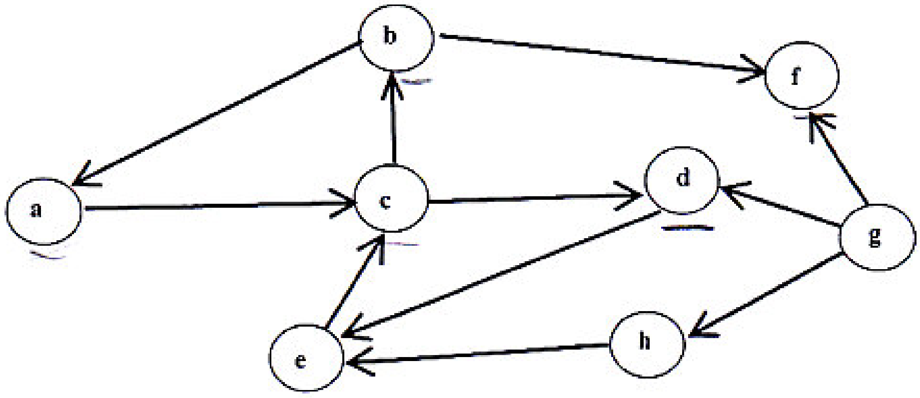
(v)

wenn 1. Teil nicht falsch wäre

1,5

6. Aufgabe (15 Punkte) 12,5

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	null	c	a	c	d	b	null	null
dist	0	2	1	2	3	3	∞	∞

5

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:

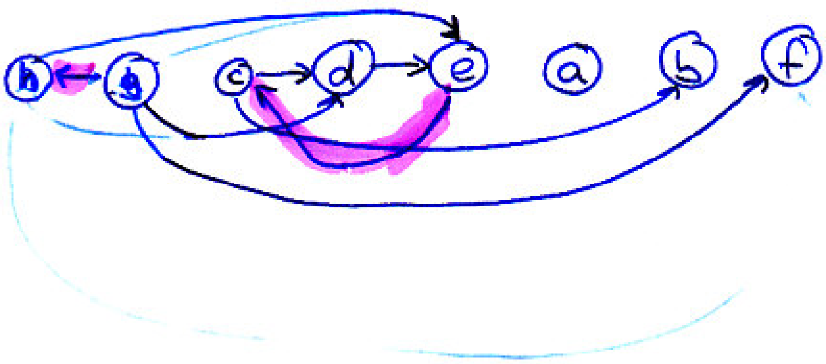
	a	b	c	d	e	f	g	h
first	1	3	2	8	9	4	13	15
last	7 f.	6	12	11	10	5	14	16
pred	null	c	a	c	d	b	null	null f
	-1							-0,5

6,5

c) Geben Sie eine topologische Sortierung des Graphen an bzw. begründen Sie, falls dieser Graph nicht topologisch sortierbar ist.

↳ ist nicht topologisch sortierbar, da Kanten {g,h}, {e,c} nach links zeigen.

1



reicht als Begründung nicht!

-1

Bitte Zyklus angeben!

7. Aufgabe (8 Punkte) 8

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

- Lineares Sondieren
- Quadratisches Sondieren

Dabei sollen folgende Schlüssel in der angegebenen Reihenfolge eingefügt werden:
16, 32, 22, 20, 14, 27, 54.

Lineares Sondieren

0	1	2	3	4	5	6	7	8	9	10
22	54		14		16	27			20	32

✓

Quadratisches Sondieren

0	1	2	3	4	5	6	7	8	9	10
22			14		16	27		54	20	32

✓

	linear	Quadratisch
16 mod 11 = 5	5	5
32 mod 11 = 10	10	10
22 mod 11 = 0	0	0
20 mod 11 = 9	9	9
14 mod 11 = 3	3	3
27 mod 11 = 5	5 ↯ 6	5 ↯ 6
54 mod 11 = 10	10 ↯ 0 ↯ 1	10 ↯ 0 ↯ 9 ↯ 3 ↯ 6 ↯ 8 (+1, −1, +4, −4, +9)

8. Aufgabe (8 Punkte) 6

Betrachten Sie nachfolgenden Graphen. Bestimmen Sie in diesem Graphen einen minimal spannenden Baum.

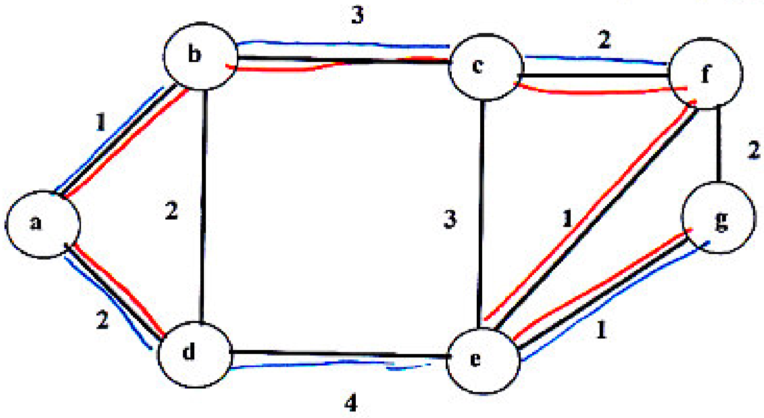
Verwenden Sie dazu

a) Kuskals Algorithmus

b) Prims Algorithmus; starten Sie dabei bei Knoten a.

unterschiedlich

Geben Sie jeweils 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.



a) Kruskal:

{a, b}
{e, f}
{e, g}
{a, d}
{c, f}
{b, c} V

b) Prim:

{a, b}
{a, d}
{b, c}
{d, e}
{c, f} f.

zu wenig

	a	b	c	d	e	f	g
pred	null	a	b	a	d	c	e
minW	0	1	4	2	6	6	7

?

-2

9. Aufgabe (8 Punkte) 4

Betrachten Sie eine Realisierung einer Queue, in der folgende Operationen implementiert sind:

- `void Enqueue(Queue Q, T w)` : fügt Wert `w` (vom Typ `T`) am Ende der Queue `Q` an
- `T Dequeue(Queue Q)` : entfernt Wert (vom Typ `T`) am Anfang der Queue `Q`; Rückgabe: der entfernte Wert
- `int Size(Queue Q)` : Rückgabe: Anzahl der Elemente in der Queue `Q`

Formulieren Sie nun mit Hilfe dieser Queue-Operationen eine neue Queue-Operation `Later(Queue Q, int n)`, die folgendes bewirkt: sie verschiebt den Wert am Anfang der Queue `Q` um `n` Positionen nach hinten. Zur Veranschaulichung soll folgendes Beispiel dienen:

head

↓

8	7	6	5	4	3	2	1

`Later(Q, 3);`

7	6	5	8	4	3	2	1

Bestimmen Sie die Laufzeit für diese Methode!

```
Later (Queue Q, int n){
   wert = Dequeue Q;
   Queue hilfe1;
   Queue hilfe2;
   for (i=0; i<n; i++){
      temp = Dequeue (Q);
      Enqueue (hilfe1, temp);
   }
   for (i=head+n; i< Q.length; i++){
      temp = Dequeue (Q);
      Enqueue (hilfe2, temp);
   }
   Enqueue (Q, hilfe1);
   Enqueue (Q, wert);
   Enqueue (Q, hilfe2);
}
```

1

2

Enque noch → Operationen, die als Argument eine Queue bekommt war nicht def.!

Laufzeit?