

Klausur Algorithmen

21

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 8,5
6 (15 Punkte)	8 12,5
7 (8 Punkte)	8
8 (8 Punkte)	6
9 (8 Punkte)	4
Summe 80 Punkte	57,0

Bearbeitungszeit: 90 Minuten

Note: 2,7Gesamtnote: 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:

```

1: // G = (V, E) sei Graph mit n Knoten und m Kanten
2: // S sei Stack mit Knoten
3: for (v ∈ V) → O(n) Durchläufe ✓ 1
4:     S.push(v);
5:
6: while (S != leer) { → O(n) Durchläufe ✓ 1
7:     v = S.pop();
8:     markiere v als besucht;
9:     for (u ∈ Adj(v)) // fuer alle Nachbarn von v → O(n) Durchläufe zu groß noch 1
10:         Gebe u aus
11: }

```

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 $O(n) + O(n \cdot n) = \underline{O(n^2)}$ (V) 1 im Worst-Case.

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!

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

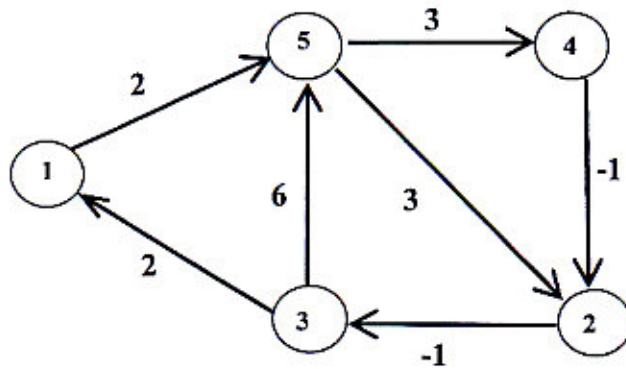
$$g(n) = \log n \quad \text{f.}$$

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

$$k(n) = n^2 \quad \text{f.}$$

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	

nicht eingetragen: ∞

PRED ⁰	1	2	3	4	5
1					1
2			2		
3	3				3
4		4			
5		5		5	

nicht eingetragen: null

kürzeste Weg

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

b) 2, 3, 9, 8, 7 ✓

c) bleibt gleich: 8, 3, 2, 9 ✓

5. Aufgabe (12 Punkte)

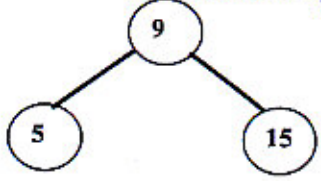
8,5

- 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.

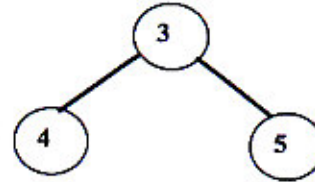
Element linker NV: kleiner
 rechter NV: größer

Knotenwert höchstens so groß wie Wert des Vorgängers

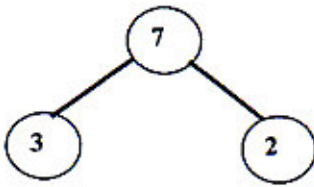
4



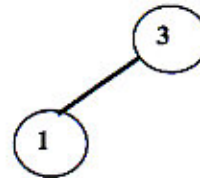
Suchbaum ✓



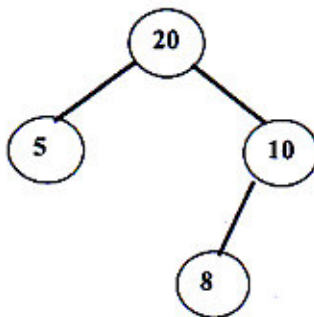
keine ✓



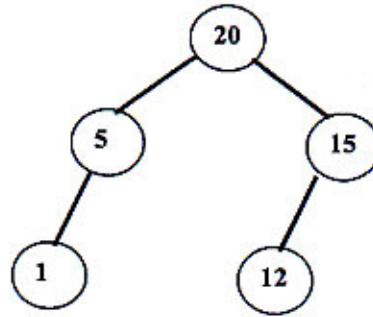
Heap ✓



Suchbaum + Heap ✓



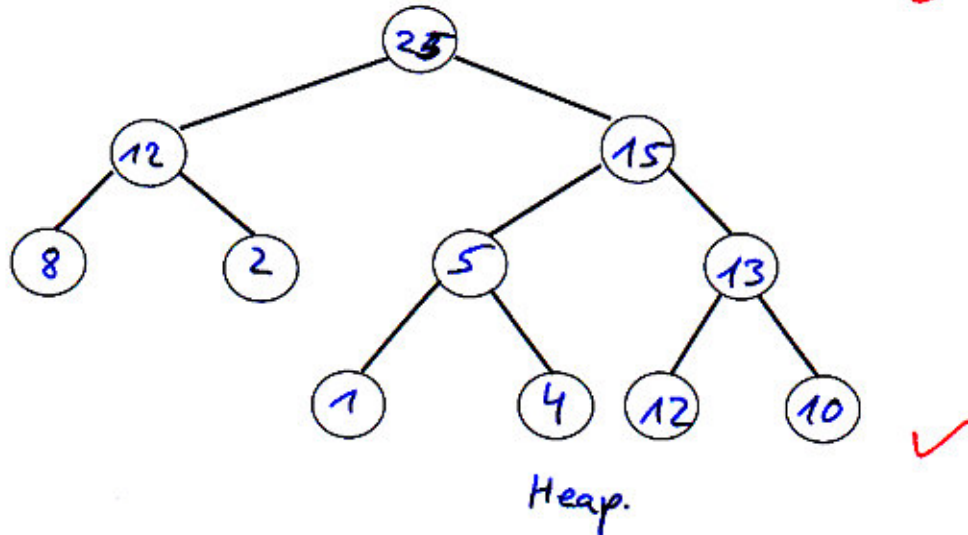
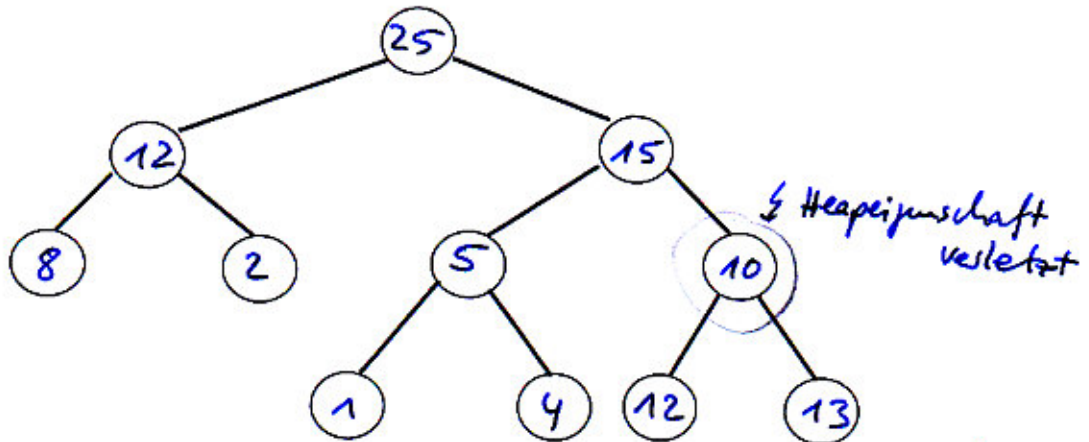
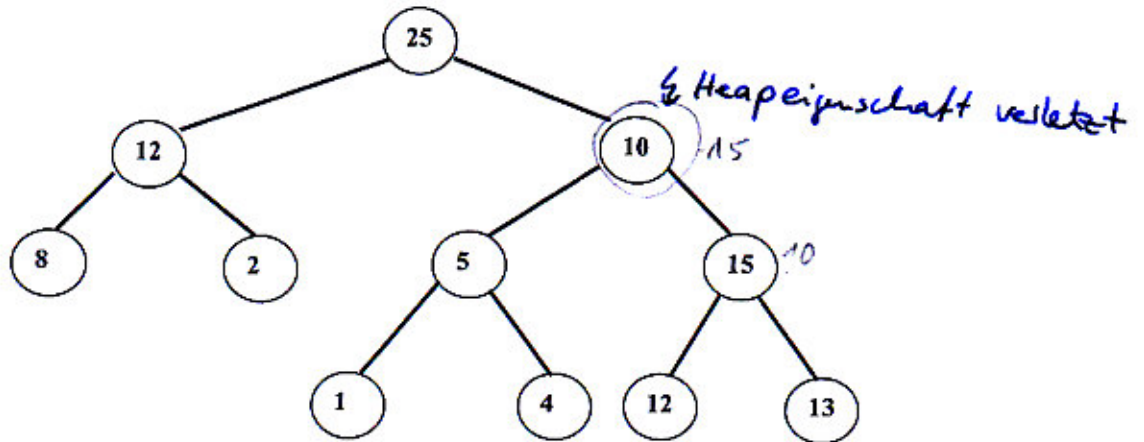
Heap f



Heap f.

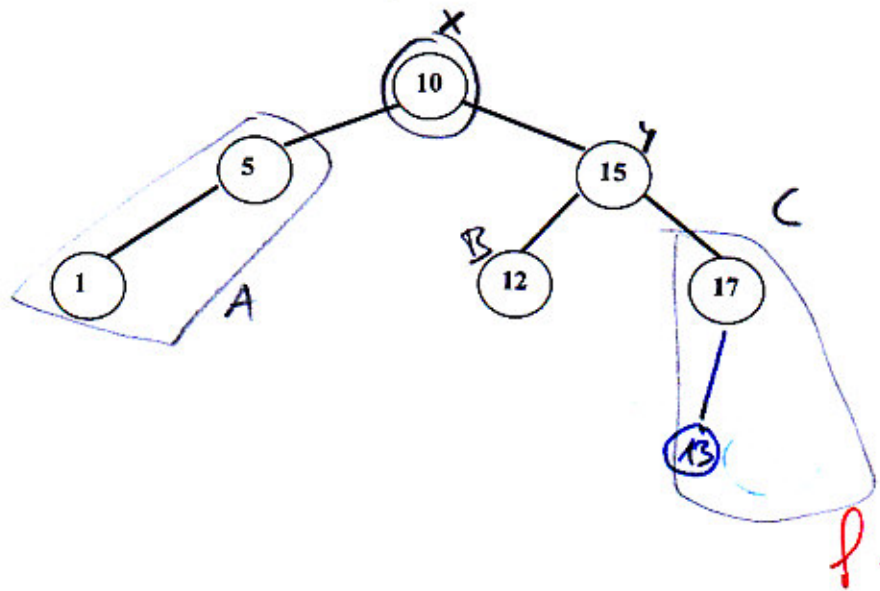
b) Geben Sie in nachfolgendem Binärbaum an, in welchem Knoten die Heapeigenschaft verletzt ist und stellen Sie mittels des Verfahrens **Heapify** einen **korrekten Heap** her. Stellen Sie den entstandenen Baum nach jedem Austauschschritt dar.

3

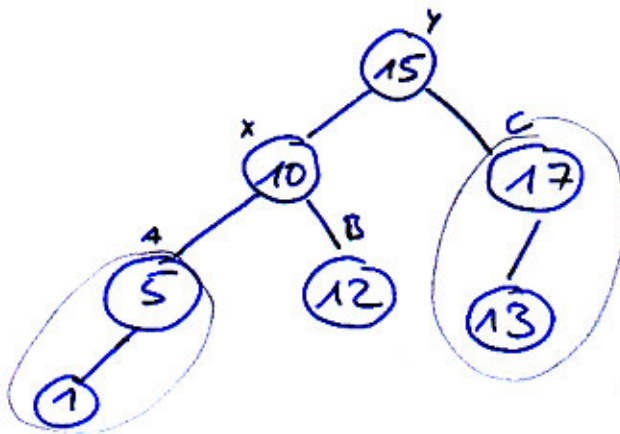


c) Betrachten Sie nachfolgenden Suchbaum. Fügen Sie in dem Baum den Knoten 13 ein (ohne den Baum zu balancieren).

1,5



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)

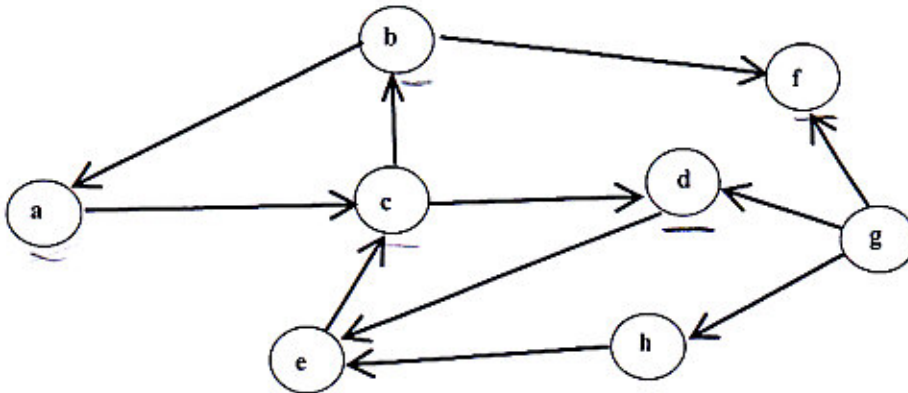
wenn 1. Teil nicht
falsch wäre

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:

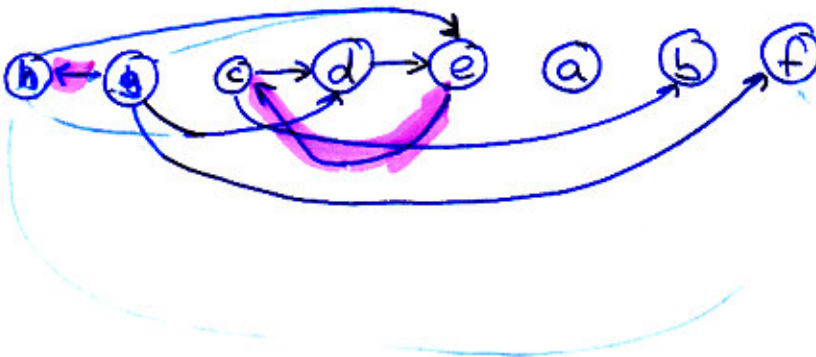
	a	b	c	d	e	f	g	h
first	1	3	2	8	9	4	13	15
last	7	6	12	11	10	5	14	16
pred	null	c	a	c	d	b	null	null

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, Knoten {g,h}, {e,c} nach vorne liegen.

1



reicht als Begründung nicht!

Bitte Zylinder angeben!

-1

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

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

Quadratisches Sondieren

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

	linear	Quadratisch
$16 \bmod M = 5$	5	5
$32 \bmod M = 10$	10	10
$22 \bmod M = 0$	0	0
$20 \bmod M = 9$	9	9
$14 \bmod M = 3$	3	3
$27 \bmod M = 5$	5 ↘ 6	5 ↘ 6
$54 \bmod M = 10$	10 ↘ 0 ↘ 1	10 ↘ 0 ↘ 9 ↘ 3 ↘ 6 ↘ 8 +1 +1 +4 -4 +3

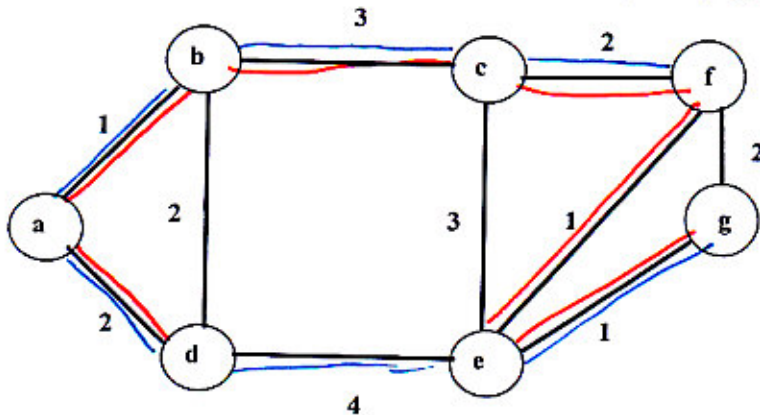
8. Aufgabe (8 Punkte) 6

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

Verwenden Sie dazu

- a) ~~Kruskal~~ Algorithmus
- b) ~~Prim~~ Algorithmus; starten Sie dabei bei Knoten **a**.

Geben Sie jeweils die ^{unterschiedlich} 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} ✓

b) Prim:

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

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

?

-2

Zu wenig

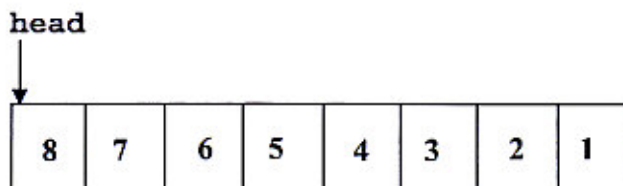
9. Aufgabe (8 Punkte)

4

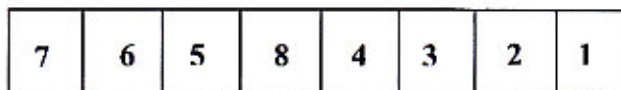
Betrachten Sie eine Realisierung einer Queue, in der folgende Operationen implementiert sind:

- `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:



`Later(Q, 3);`



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);`

`}`

15.07.08

noch ~~es~~ →
Operationen die als Argument eine Queue bekommt was nicht def.!
Laufzeit?