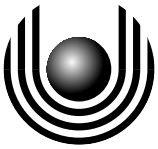


FernUniversität in Hagen

Lösungsvorschläge
zur Klausur
63113 „Datenstrukturen und Algorithmen“

25.03.2021

Aufgabe 1

(a)

ops	*produktVorhanden:*	$katalog \times produkt$	$\rightarrow bool$
	neuerKatalog:		$\rightarrow katalog$
	neuesProdukt:	$ID \times int_0$	$\rightarrow produkt$
	produktEinfügen:	$katalog \times produkt$	$\rightarrow katalog$
	verkäufeÄndern:	$produkt \times int_+ \times int_+$	$\rightarrow produkt$
	bestandÜberErmitteln:	$katalog \times int_0$	$\rightarrow katalog$
	gesamtbestandBerechnen:	$katalog$	$\rightarrow int_0$
	produktOhneBestandLöschen:	$katalog \times produkt$	$\rightarrow katalog$
	katalogLeeren:	$katalog$	$\rightarrow katalog$

(b)

functions

$$produktVorhanden(K,p) = \begin{cases} true & \text{falls } \exists pk \in K : \pi_1(pk) = \pi_1(p) \\ false & \text{sonst} \end{cases}$$

$$neuerKatalog() = \emptyset$$

$$neuesProdukt(id,b) = (id, \emptyset, b)$$

$$produktEinfügen(K,p) = \begin{cases} K \cup \{p\} & \text{falls } \neg produktVorhanden(K,p) \\ K & \text{sonst} \end{cases}$$

$$verkäufeÄndern(p,l,z) = \begin{cases} (\pi_1(p), \pi_2(p) \cup \{(l,z)\}, \pi_3(p)) \\ \quad \text{falls } \nexists t \in \pi_2(p) : \pi_1(t) = l \\ (\pi_1(p), \pi_2(p) \setminus \{t \mid t \in \pi_2(p) \wedge \pi_1(t) = l\} \cup \{(l,z)\}, \pi_3(p)) \\ \quad \text{sonst} \end{cases}$$

$$bestandÜberErmitteln(K,b) = \{p \in K : \pi_3(p) > b\}$$

$$gesamtbestandBerechnen(K) = \left(\sum_{p \in K} \pi_3(p) \right)$$

$$produktOhneBestandLöschen(K,p) = \{p \in K : produktVorhanden(K,p) \wedge \pi_3(p) > 0\}$$

$$katalogLeeren(K) = \emptyset$$

Aufgabe 2

(a)

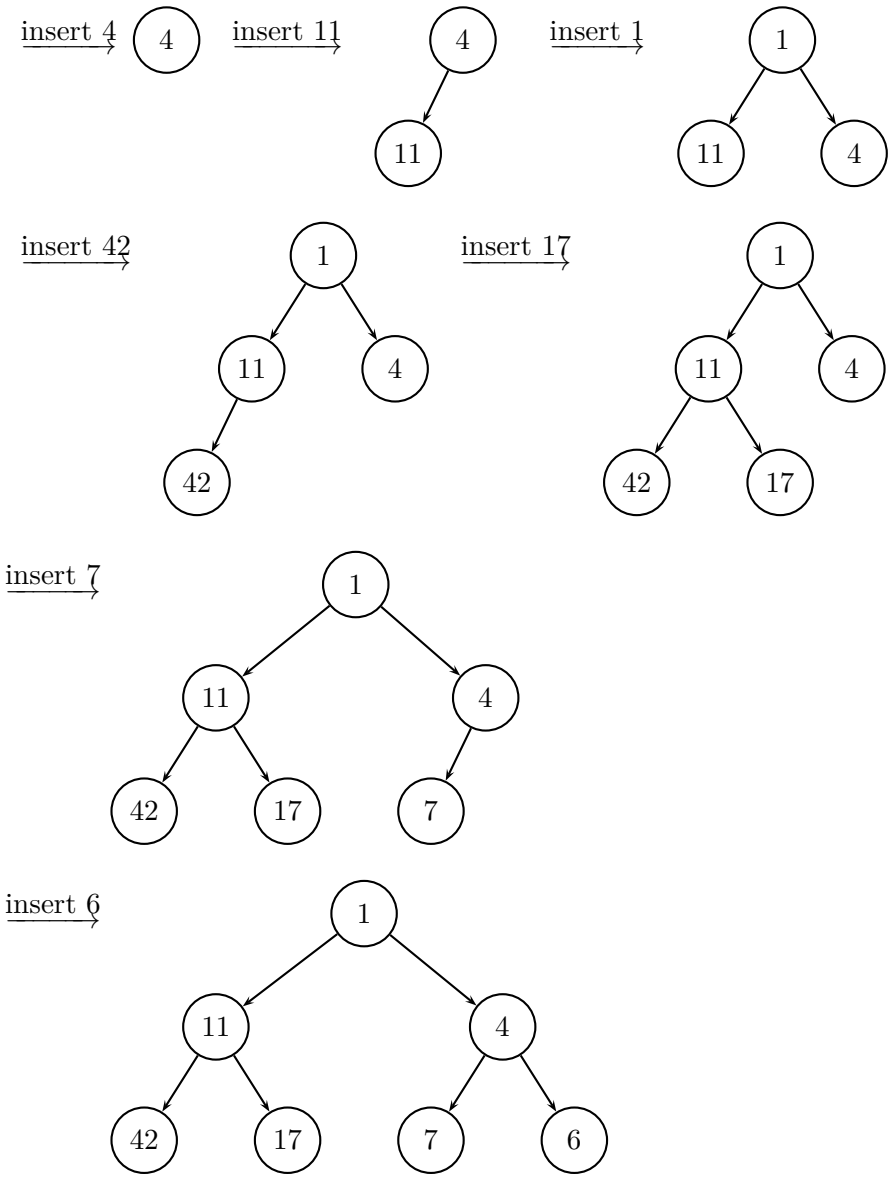
(i) Der Baum ist kein Minimum-Heap, da die Heap-Eigenschaft am Knoten mit dem Eintrag 3 verletzt ist ($3 > 2$).

(ii) Dieser Baum ist kein links-vollständiger Minimum-Heap, weil der Knoten mit dem Eintrag 3 zwar einen rechten, aber keinen linken Sohn hat. Er stellt aber einen Minimum-Heap dar.

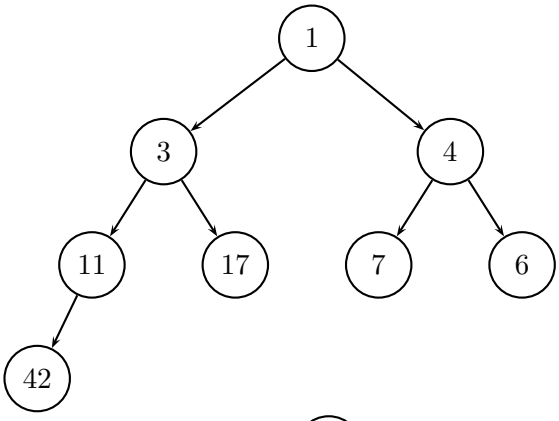
(iii) Dieser Baum ist auch kein links-vollständiger Minimum-Heap, da die vorletzte Ebene nicht aufgefüllt ist. Er ist auch kein Minimum-Heap, da die Heap-Bedingung am Knoten mit dem Eintrag 4 verletzt ist ($4 > 3$).

(iv) Dieser Baum ist ein links-vollständiger Minimum-Heap.

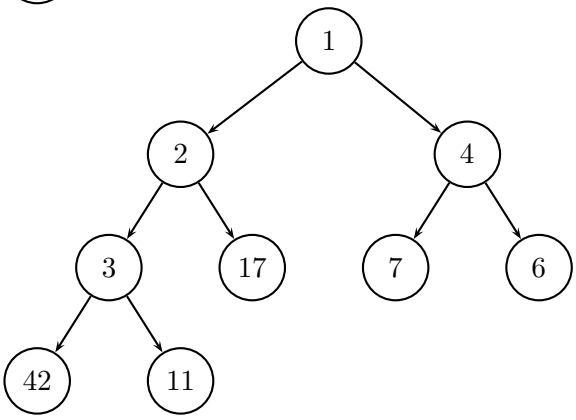
(b)



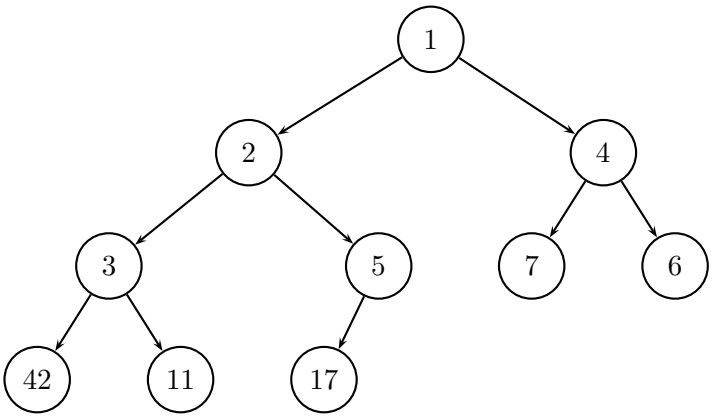
insert 3 $\longrightarrow$



insert 2 $\longrightarrow$

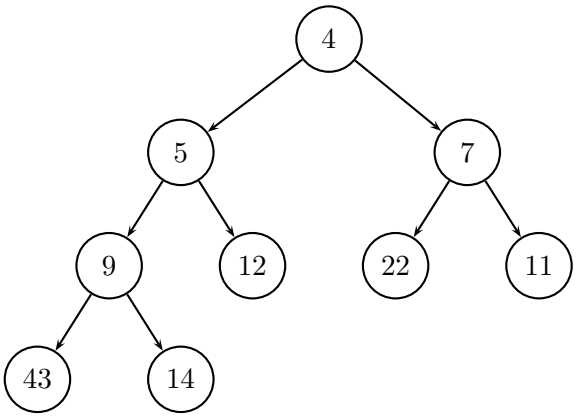


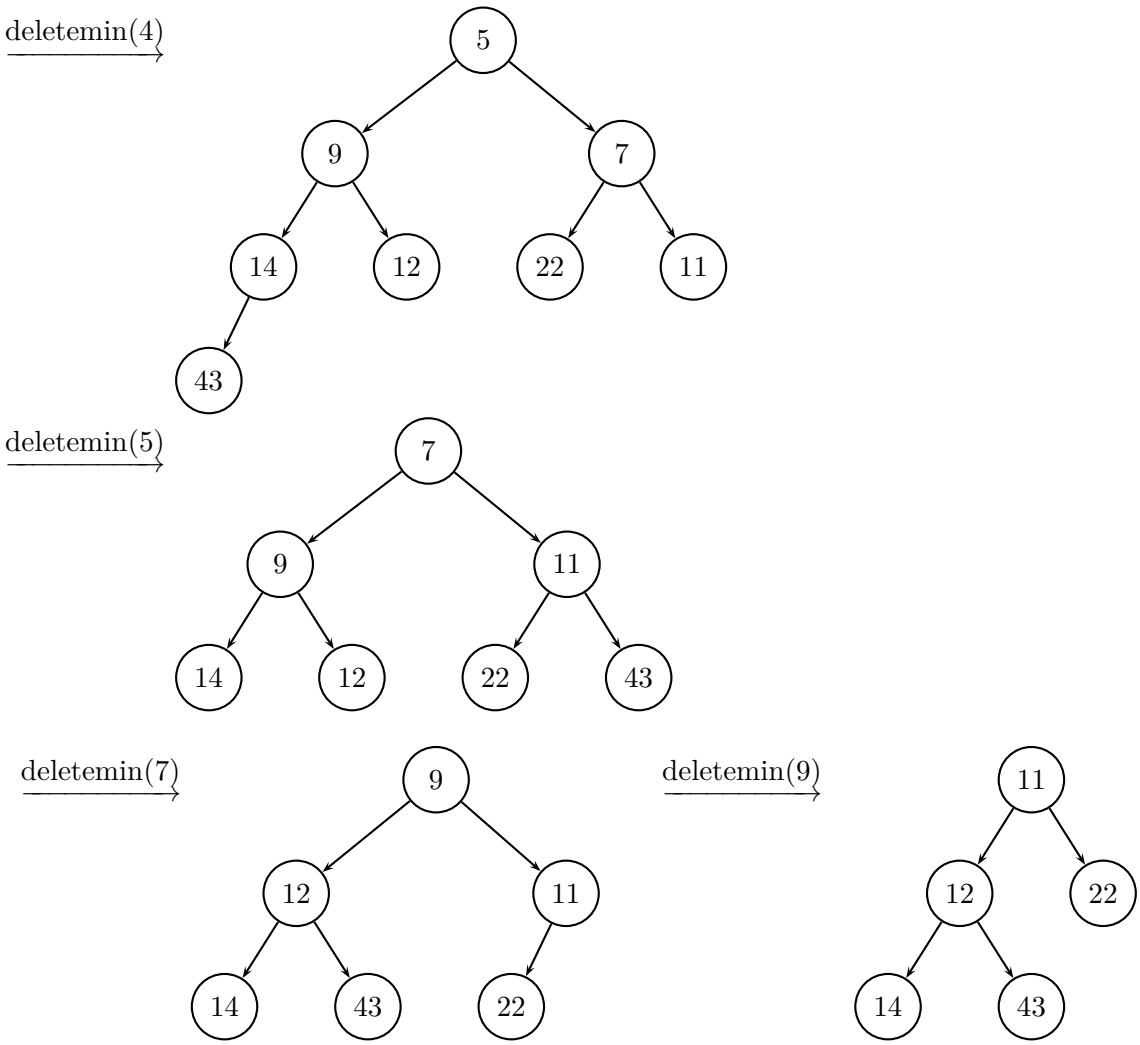
insert 5 $\longrightarrow$



(c)

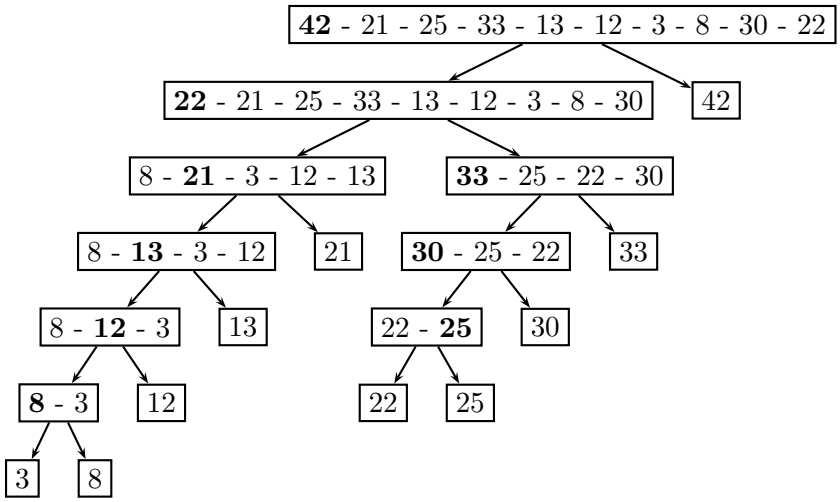
deletemin (3) $\longrightarrow$





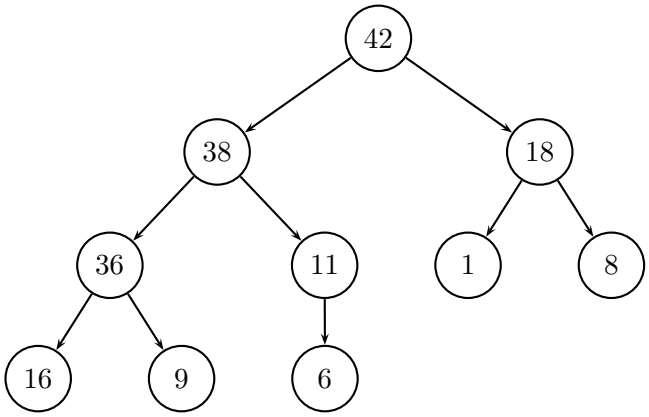
Aufgabe 3 Sortieren

(a)



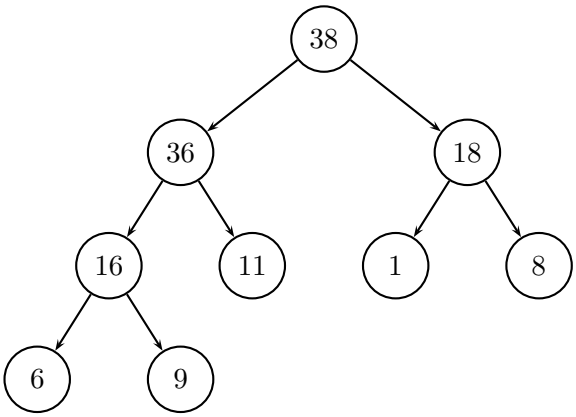
(b)

Der initiale Heap sieht wie folgt aus:



Arrayeinbettung: | 42 | 38 | 18 | 36 | 11 | 1 | 8 | 16 | 9 | 6 |

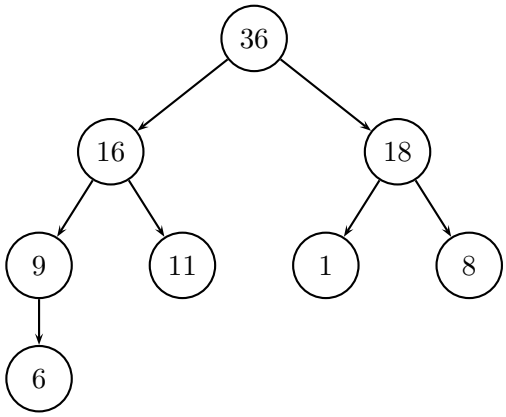
- Nach Verarbeitung des Elements 42 ergibt sich:



Sortierte Folge: 42

Arrayeinbettung: | 38 | 36 | 18 | 16 | 11 | 1 | 8 | 6 | 9 ‖ 42 |

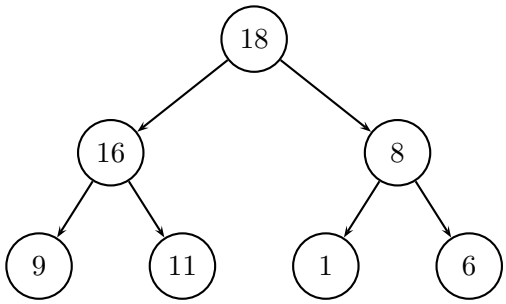
- Nach Verarbeitung des Elements 38 ergibt sich:



Sortierte Folge: 38, 42

Arrayeinbettung: | 36 | 16 | 18 | 9 | 11 | 1 | 8 | 6 ‖ 38 | 42 |

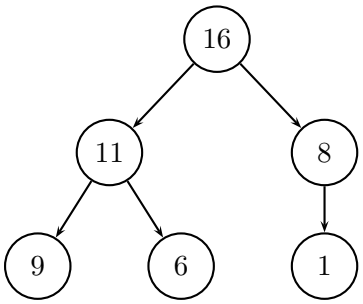
- Nach Verarbeitung des Elements 36 ergibt sich:



Sortierte Folge: 36, 38, 42

Arrayeinbettung: | 18 | 16 | 8 | 9 | 11 | 1 | 6 ‖ 36 | 38 | 42 |

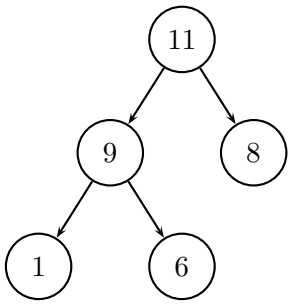
- Nach Verarbeitung des Elements 18 ergibt sich:



Sortierte Folge: 18, 36, 38, 42

Arrayeinbettung: | 16 | 11 | 8 | 9 | 6 | 1 ‖ 18 | 36 | 38 | 42 |

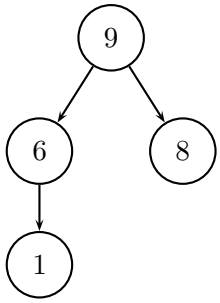
- Nach Verarbeitung des Elements 16 ergibt sich:



  Sortierte Folge: 16, 18, 36, 38, 42

  Arrayeinbettung: | 11 | 9 | 8 | 1 | 6 || 16 | 18 | 36 | 38 | 42 |

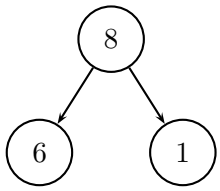
- Nach Verarbeitung des Elements 11 ergibt sich:



  Sortierte Folge: 11, 16, 18, 36, 38, 42

  Arrayeinbettung: | 9 | 6 | 8 | 1 || 11 | 16 | 18 | 36 | 38 | 42 |

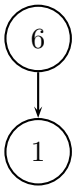
- Nach Verarbeitung des Elements 9 ergibt sich:



  Sortierte Folge: 9, 11, 16, 18, 36, 38, 42

  Arrayeinbettung: | 8 | 6 | 1 || 9 | 11 | 16 | 18 | 36 | 38 | 42 |

- Nach Verarbeitung des Elements 8 ergibt sich:



  Sortierte Folge: 8, 9, 11, 16, 18, 36, 38, 42

  Arrayeinbettung: | 6 | 1 ‖ 8 | 9 | 11 | 16 | 18 | 36 | 38 | 42 |

- Nach Verarbeitung des Elements 6 ergibt sich:

  Sortierte Folge: 6, 8, 9, 11, 16, 18, 36, 38, 42

  Arrayeinbettung: | 1 ‖ 6 | 8 | 9 | 11 | 16 | 18 | 36 | 38 | 42 |

- Final erhalten wir:

  Sortierte Folge: 1, 6, 8, 9, 11, 16, 18, 36, 38, 42

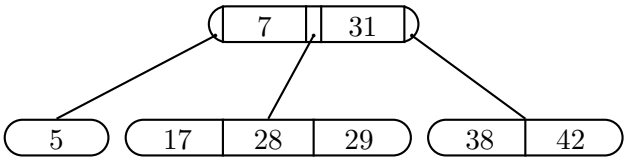
  Arrayeinbettung: ‖ 1 | 6 | 8 | 9 | 11 | 16 | 18 | 36 | 38 | 42 |

Aufgabe 4

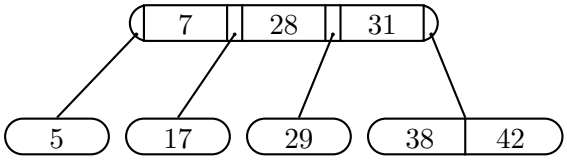
(a)

Die Elemente 28 und 29 werden eingefügt. Beim Einfügen des Elements 29 tritt ein Überlauf auf. Es wird ein Split notwendig.

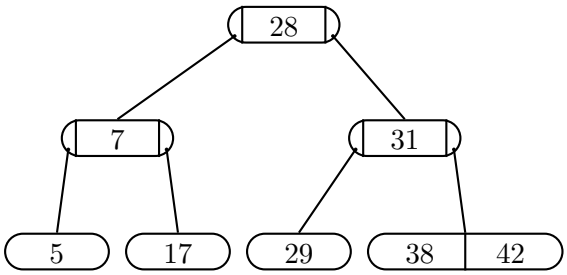
Baum vor dem Split:



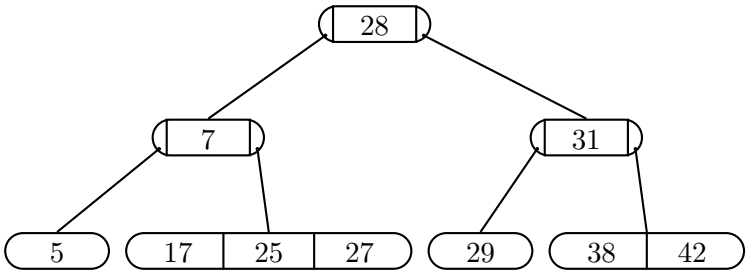
Durch den Split kommt es zu einem Überlauf in der Wurzel.



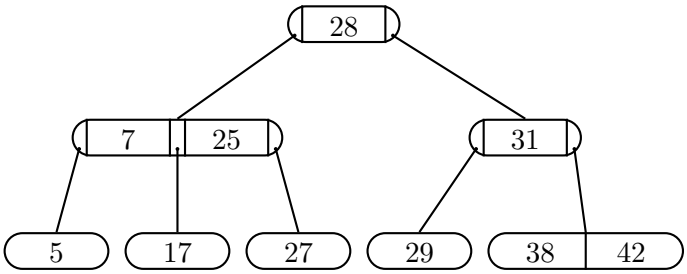
Ein erneuter Split ist durchzuführen. Das Ergebnis ist:



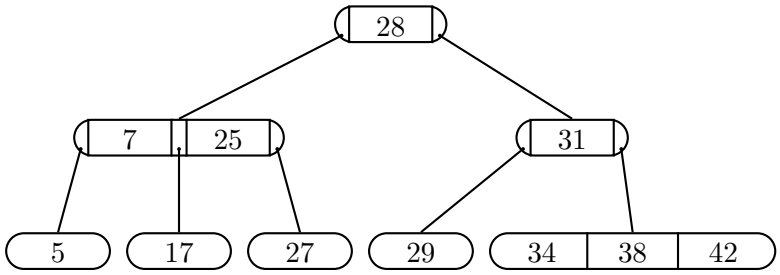
Die Element 25 und 27 werden eingefügt. Die 27 verursacht einen Überlauf:



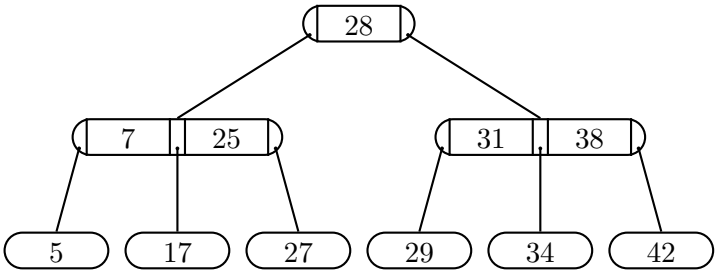
Es wird ein Split durchgeführt:



Das Element 34 wird eingefügt und verursacht einen Überlauf.

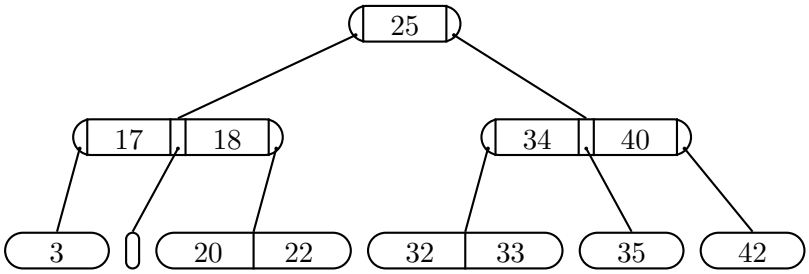


Es wird ein Split durchgeführt. Das Ergebnis aller Einfügungen ist somit der folgende Baum.

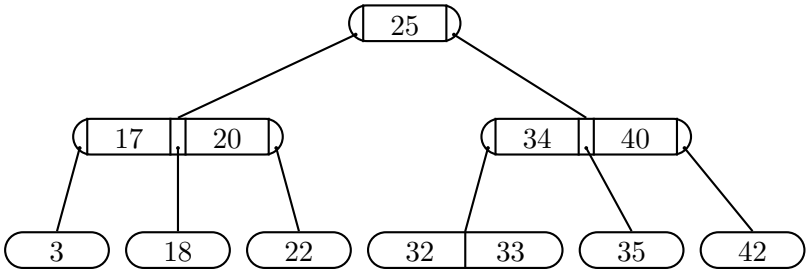


(b)

Das Löschen des Elements 6 verursacht einen Unterlauf. Eine Balance-Operation wird notwendig.
Baum vor der Balance:

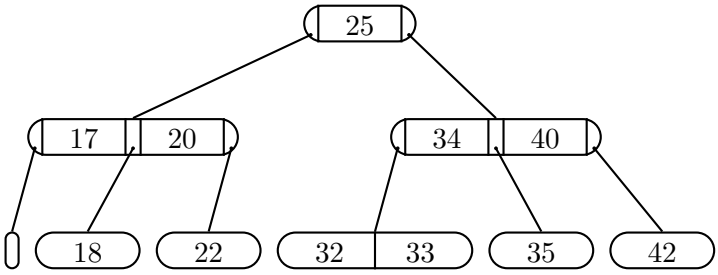


Baum nach der Balance:

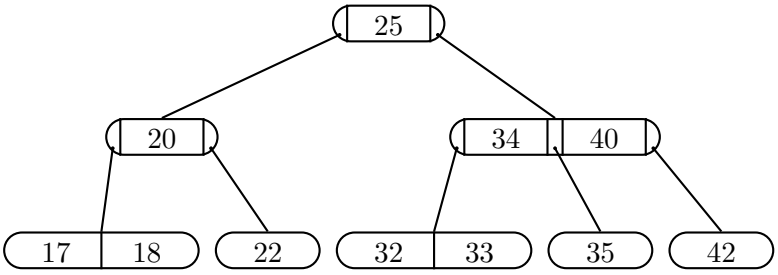


Das Element 3 wird gelöscht, was eine Merge-Operation zur Folge hat.

Baum vor dem Merge:

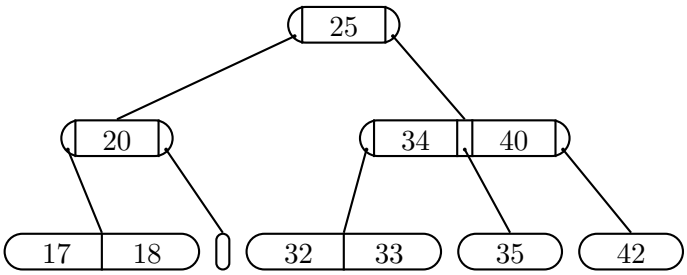


Baum nach dem Merge:

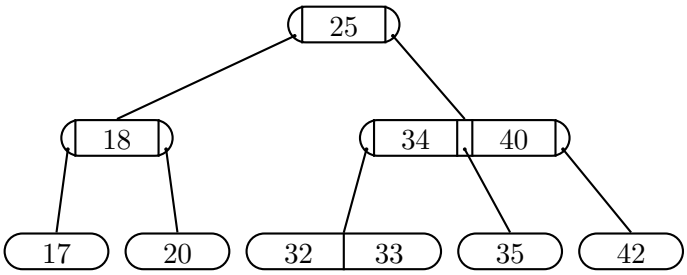


Dem Löschen von 22 folgt eine Balance-Operation.

Baum vor der Balance:

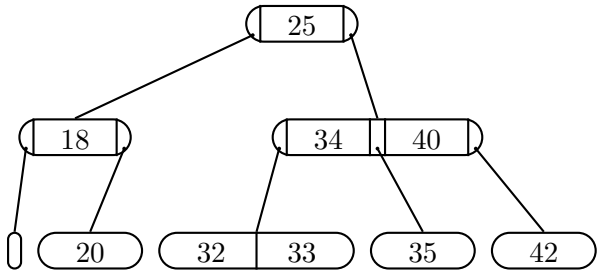


Baum nach der Balance:

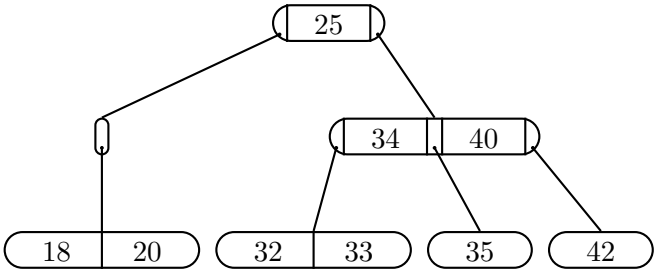


Element 17 wird gelöscht, anschließend muss gemerged werden.

Baum vor dem Merge:

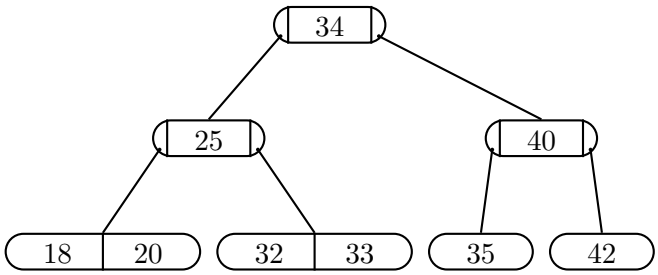


Baum nach Merge:



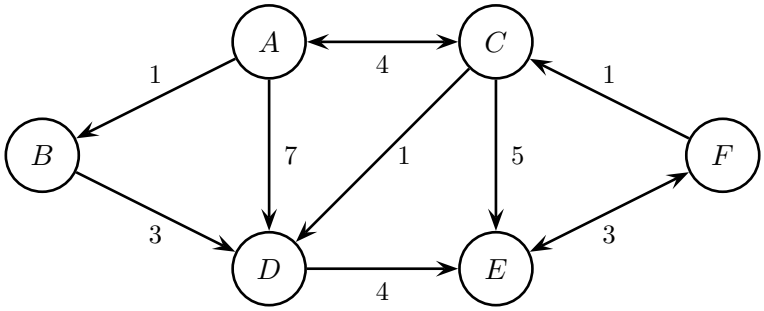
Die Merge-Operation zieht eine weitere Balance-Operation nach sich.

Baum nach der Balance (finaler Baum):



Aufgabe 5 Lösung

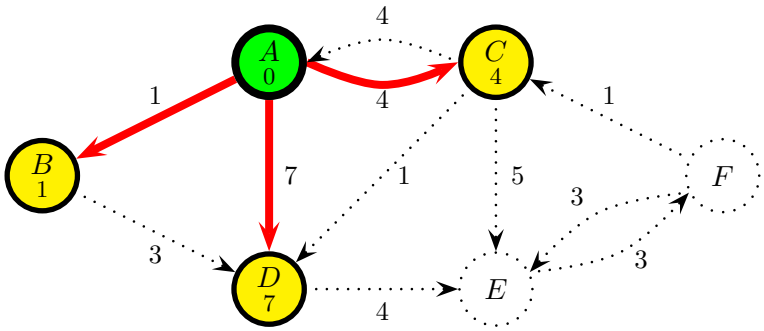
Ausgangsgraph :



Aktionen beim Verarbeiten des Knotens A

- Färbe Knoten A grün
- Färbe Knoten B gelb, setze Distanz auf 1
- Färbe Kante (A,B) rot
- Färbe Knoten C gelb, setze Distanz auf 4
- Färbe Kante (A,C) rot
- Färbe Knoten D gelb, setze Distanz auf 7
- Färbe Kante (A,D) rot

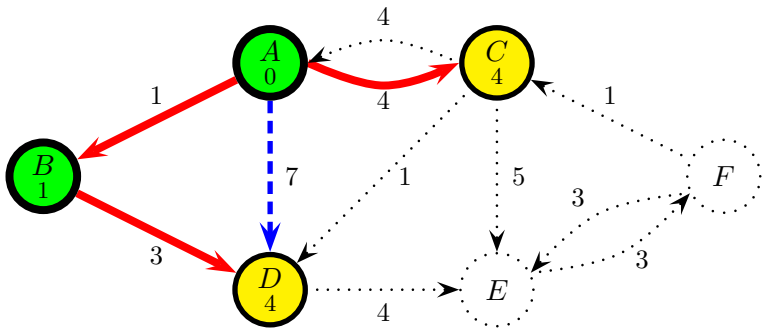
Nach Verarbeitung des Knotens A



Aktionen beim Verarbeiten des Knotens B

- Färbe Knoten B grün
- Kürzerer Weg zu Knoten D gefunden, die neue Distanz beträgt 4
- Färbe Kante (A,D) gelb
- Färbe Kante (B,D) rot

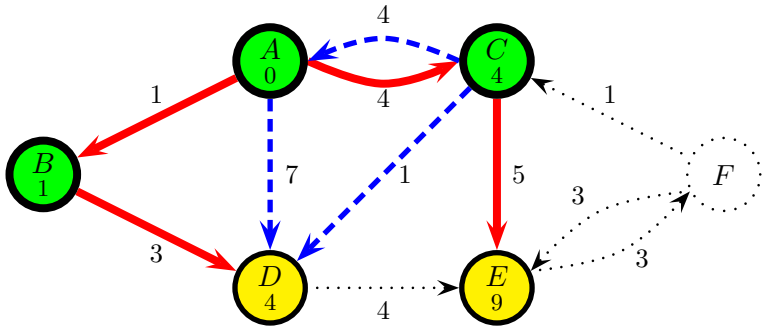
Nach Verarbeitung des Knotens B



Aktionen beim Verarbeiten des Knotens C

- Färbe Knoten C grün
- Färbe Kante (C,A) gelb
- Färbe Kante (C,D) gelb
- Färbe Knoten E gelb, setze Distanz auf 9
- Färbe Kante (C,E) rot

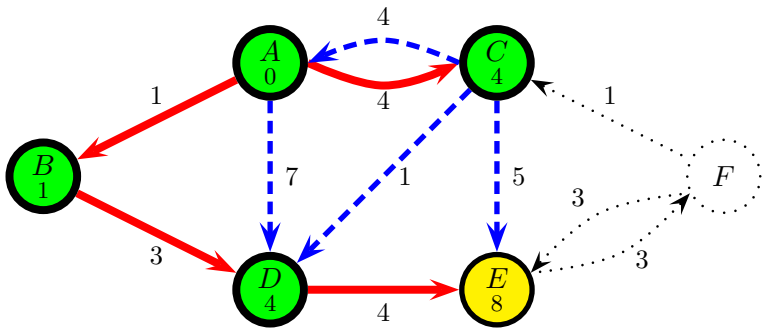
Nach Verarbeitung des Knotens C



Aktionen beim Verarbeiten des Knotens D

- Färbe Knoten D grün
- Kürzerer Weg zu Knoten E gefunden, die neue Distanz beträgt 8
- Färbe Kante (C,E) gelb
- Färbe Kante (D,E) rot

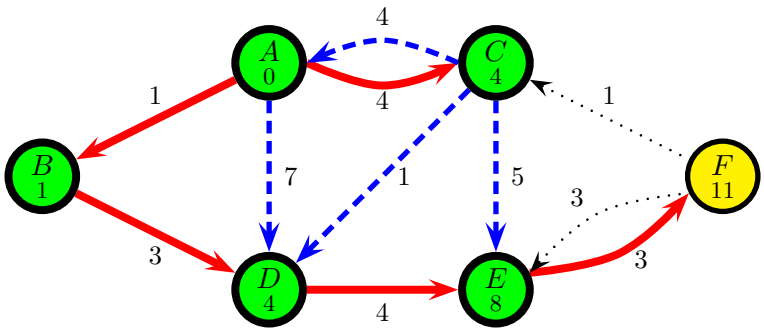
Nach Verarbeitung des Knotens D



Aktionen beim Verarbeiten des Knotens E

- Färbe Knoten E grün
- Färbe Knoten F gelb, setze Distanz auf 11
- Färbe Kante (E,F) rot

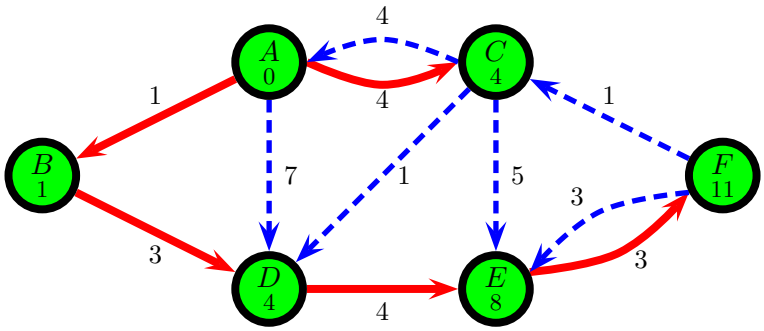
Nach Verarbeitung des Knotens E



Aktionen beim Verarbeiten des Knotens F

- Färbe Knoten F grün
- Färbe Kante (F,C) gelb
- Färbe Kante (F,E) gelb

Nach Verarbeitung des letzten Knotens ergibt sich folgendes Bild:



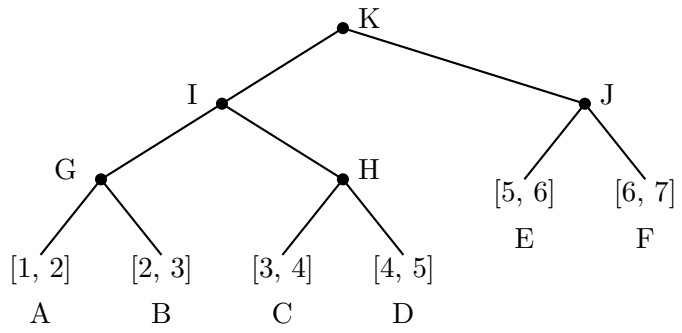
Eine tabellarische Lösung kann wie folgt aussehen:

Schritt	grün	gelbe Knoten	rote Kanten	gelbe Kanten	Updates
1	A	$B(1)$, $C(4)$, $D(7)$	(A, B), (A, C), (A, D)		
2	B		(B, D)	(A, D)	$D(4)$
3	C	$E(9)$	(C, E)	(C, A), (C, D)	
4	D		(D, E)	(C, E)	$E(8)$
5	E	$F(11)$	(E, F)		
6	F			(F, C), (F, E)	

Aufgabe 6 Lösung

(a)

Der Segmentbaum besitzt die folgende Gestalt:



(b)

Die Events sind die folgenden:

$1 : (2, left, 3, 7, r_4)$	$2 : (3, left, 2, 5, r_2)$	$3 : (3, point, 7, p_4)$	$4 : (4, point, 4, p_2)$
$5 : (5, right, 3, 7, r_4)$	$6 : (6, left, 1, 3, r_1)$	$7 : (6, point, 2, p_1)$	$8 : (6, point, 6, p_5)$
$9 : (6, point, 4, p_6)$	$10 : (7, left, 4, 6, r_3)$	$11 : (7, right, 1, 3, r_1)$	$12 : (9, point, 5, p_3)$
$13 : (9, right, 2, 5, r_2)$	$14 : (10, right, 4, 6, r_3)$		

Bei der Sortierung muss man darauf achten, dass bei gleichen x-Koordinaten linke Rechteckseiten vor den Punkten und Punkte vor den rechten Rechteckseiten verarbeitet werden.

(c)

1. $(2, left, 3, 7, r_4)$: Das Intervall $[3, 7]$ wird in den Baum eingetragen. Daher haben die Knoten H und J nun r_4 als Intervallliste.
2. $(3, left, 2, 5, r_2)$: Das Intervall $[2, 5]$ wird in den Baum eingetragen. Die Intervalllisten der Knoten B und H werden um r_2 erweitert.
3. $(3, point, 7, p_4)$: Die Suche im Baum nach Koordinate 7 liefert das Ergebnis $\{(p_4, r_4)\}$.
4. $(4, point, 4, p_2)$: Die Suche im Baum nach Koordinate 4 liefert das Ergebnis $\{(p_2, r_4), (p_2, r_2)\}$.
5. $(5, right, 3, 7, r_4)$: Der Eintrag r_4 wird aus der Intervallliste der Knoten H und J entfernt.
6. $(6, left, 1, 3, r_1)$: Das Intervall $[1, 3]$ wird eingetragen. Der Knoten G erhält r_1 als Intervallliste.
7. $(6, point, 2, p_1)$: Die Suche im Baum nach Koordinate 2 liefert das Ergebnis $\{(p_1, r_1), (p_1, r_2)\}$.
8. $(6, point, 4, p_6)$: Die Suche im Baum nach Koordinate 4 liefert das Ergebnis $\{(p_6, r_2)\}$.
9. $(6, point, 6, p_5)$: Die Suche im Baum nach Koordinate 6 liefert kein Ergebnis.
10. $(7, left, 4, 6, r_3)$: Das Intervall $[4, 6]$ wird eingetragen. Die Knoten D und E erhalten r_3 als Intervallliste.

11. $(7, right, 1, 3, r_1)$: Der Eintrag r_1 wird aus der Intervallliste des Knoten G entfernt.
12. $(9, point, 5, p_3)$: Die Suche im Baum nach Koordinate 5 liefert das Ergebnis $\{(p_3, r_2), (p_3, r_3)\}$.
13. $(9, right, 2, 5, r_2)$: Der Eintrag r_2 wird aus der Intervallliste der Knoten B und H entfernt.
14. $(10, right, 4, 6, r_3)$: Der Eintrag r_3 wird aus der Intervallliste der Knoten D und E entfernt. Nach Verarbeitung dieses Segments ist der Intervallbaum leer.

Anmerkung: Die Schritte 7, 8 und 9 können beliebig vertauscht werden.

Aufgabe 7 Lösung

(a) Anfangsläufe
f1: 21 | 17 | 38 | 2 | 16 | 55 | 12 |
f2: 3 | 29 | 5 | 71 | 25 | 99 |

Phase 1
g1: 3 21 | 5 38 | 16 25 | 12 |
g2: 17 29 | 2 71 | 55 99 |

Phase 2
f1: 3 17 21 29 | 16 25 55 99 |
f2: 2 5 38 71 | 12 |

Phase 3
g1: 2 3 5 17 21 29 38 71 |
g2: 12 16 25 55 99 |

Phase 4
f1: 2 3 5 12 16 17 21 25 29 38 55 71 99 |
f2:

(b) Anfangsläufe
f1: 3 17 21 | 2 16 71 | 12 |
f2: 5 29 38 | 25 55 99 |

Phase 1
g1: 3 5 17 21 29 38 | 12 |
g2: 2 16 25 55 71 99 |

Phase 2
f1: 2 3 5 16 17 21 25 29 38 55 71 99 |
f2: 12 |

Phase 3
g1: 2 3 5 12 16 17 21 25 29 38 55 71 99 |
g2:

(c)

Erster Lösungsweg (passend zum Kurstext):

1	2	3	1. Lauf
21	3	17	3
21	29	17	17
21	29	38	21
5	29	38	29
5	2	38	38
5	2	71	71
5	2	16	

1	2	3	2. Lauf
5	2	16	2
5	25	16	5
55	25	16	16
55	25	99	25
55	12	99	55
	12	99	99

1	2	3	3. Lauf
	12		12

Anfangsläufe
f1: 3 17 21 29 38 71 | 12 |
f2: 2 5 16 25 55 99 |

Phase 1
g1: 2 3 5 16 17 21 25 29 38 55 71 99 |
g2: 12 |

Phase 2
f1: 2 3 5 12 16 17 21 25 29 38 55 71 99 |
f2:

Zweiter Lösungsweg (mit Array-Heapdarstellung):

1	2	3	1. Lauf
3	21	17	3
17	21	29	17
21	29	38	21
29	38	5	29
38	2	5	38
71	2	5	71
16	2	5	

1	2	3	2. Lauf
2	16	5	2
5	16	25	5
16	25	55	16
25	55	99	25
55	99	12	55
99		12	99

1	2	3	3. Lauf
12			12

Anfangsläufe
f1: 3 17 21 29 38 71 | 12 |
f2: 2 5 16 25 55 99 |

Phase 1
g1: 2 3 5 16 17 21 25 29 38 55 71 99 |
g2: 12 |

Phase 2
f1: 2 3 5 12 16 17 21 25 29 38 55 71 99 |
f2: