Ludwig-Maximilians-Universität München
Institut für Informatik
PD Dr. Matthias Renz
Dr. Erich Schubert, Daniel Kaltenthaler

München, 6.7.2015

Algorithmen und Datenstrukturen

SS 2015

Übungsblatt 12: Klausurvorbereitung

Wird nicht besprochen – Musterlösung ab 15.7.2015

Die Aufgaben auf diesem Blatt sind zusätzliches Übungsmaterial, und Wiederholungen früherer Aufgaben.

Diese Aufgaben decken nur einen kleinen Teil des Vorlesungsstoff ab, und sind vom Umfang bewusst groß gewählt. Sie können aber mit diesen Aufgaben üben, ob sie die Algorithmischen Abläufe dieser Verfahren verstanden haben, sie sollten aber zur Klausurvorbereitung dennoch auch alle anderen Übungsblätter wiederholen, und das Skript studieren. Es handelt sich bei diesen Aufgaben nur um Zusatzmaterial, dass wir mit geringem Aufwand bereitstellen können!

Aufgabe 12-1 HeapSort

Sortieren Sie das folgende Array mit dem HeapSort-Verfahren *exakt wie es in der Vorlesung besprochen wurde*.

Geben Sie dazu eine Tabelle ab, mit:

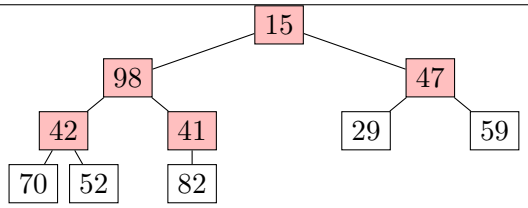
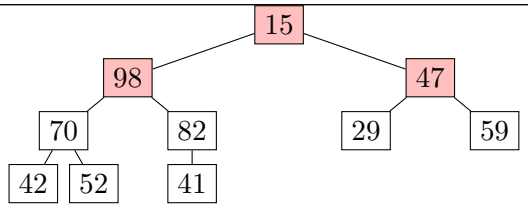
- 1. Spalte: Interne Repräsentation als Array (inkl. serialisiertem Heap und bereits sortierten Daten)
- 2. Spalte: Interpretation als Baumstruktur (graphisch, ohne bereits sortiere Daten)

In die erste Zeile tragen Sie die initialen Daten ein (inkl. graphischer Interpretation als Baum!)

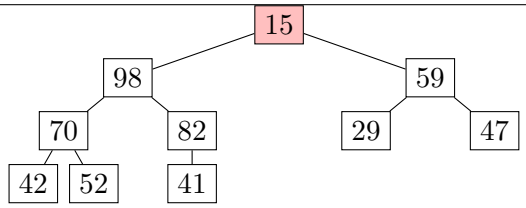
Nun erzeugen Sie den Heap, indem Sie ihn von unten nach oben korrigieren. Geben Sie die Daten in einer neuen Zeile immer dann an, *nachdem* ein Level *fertig korrigiert* ist. Kennzeichnen Sie in beiden Repräsentationen, welcher Teil des Heaps noch nicht korrigiert wurde. Alle Korrekturen im gleichen Level können Sie zusammen in einem Schritt durchführen, da diese voneinander unabhängig sind.

Sie sollten jetzt 4 Zeilen gefüllt haben. Kontrollieren Sie, ob Sie einen korrekten Heap haben!
Wenn Sie mit einem fehlerhaften Heap fortfahren, wird der Rest der Aufgabe mit 0 Punkten bewertet!

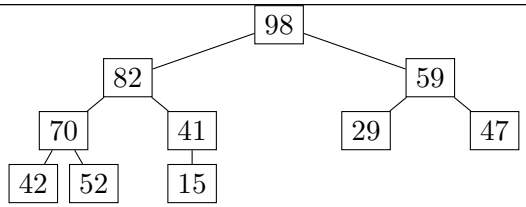
Anschließend führen Sie die zweite Phase des HeapSort durch, und zeichnen den Heap *nach* jedem Sortierschritt (d.h. auch *nach* dem Korrigieren des Heaps). Beachten Sie dass nach jedem Schritt ein korrekter Heap vorhanden sein sollte, wenn Sie keinen Fehler gemacht haben.
Wenn Sie mit einem fehlerhaften Heap fortfahren, wird der Rest der Aufgabe mit 0 Punkten bewertet!

Array (Echte Daten)	Baum (Interpretation)
15, 98, 47, 42, 41, 29, 59, 70, 52, 82	
15, 98, 47, 70, 82, 29, 59, 42, 52, 41	

15, 98, 59, 70, 82, 29, 47, 42, 52, 41

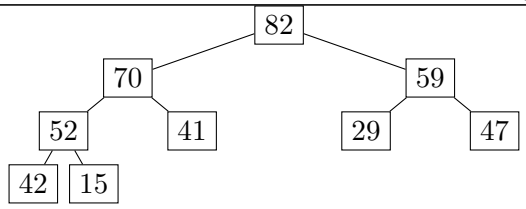


98, 82, 59, 70, 41, 29, 47, 42, 52, 15

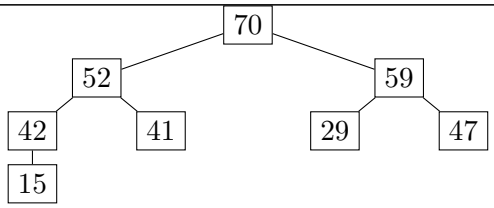


Heap ist jetzt vollständig aufgebaut.
Kontrollieren Sie die Heap-Eigenschaft! Arbeiten Sie keinesfalls mit einem fehlerhaften Heap weiter!

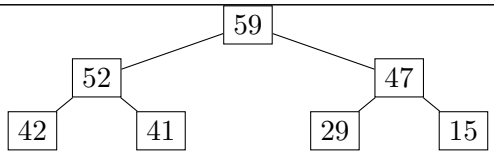
82, 70, 59, 52, 41, 29, 47, 42, 15, **98**



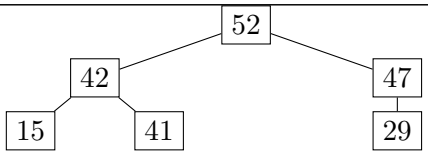
70, 52, 59, 42, 41, 29, 47, 15, **82**, **98**



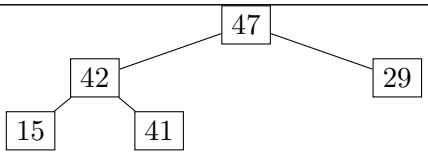
59, 52, 47, 42, 41, 29, 15, **70**, **82**, **98**



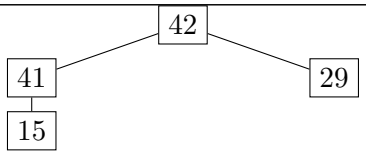
52, 42, 47, 15, 41, 29, **59**, **70**, **82**, **98**



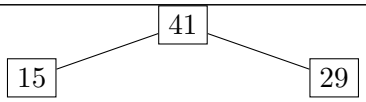
47, 42, 29, 15, 41, **52**, **59**, **70**, **82**, **98**



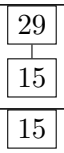
42, 41, 29, 15, **47**, **52**, **59**, **70**, **82**, **98**



41, 15, 29, **42**, **47**, **52**, **59**, **70**, **82**, **98**



29, 15, **41**, **42**, **47**, **52**, **59**, **70**, **82**, **98**



15, **29**, **41**, **42**, **47**, **52**, **59**, **70**, **82**, **98**

Aufgabe 12-2 Quick-Sort

Sortieren Sie die folgenden Zahlen mit dem Quick-Sort Algorithmus aus der Vorlesung:
36, 9, 58, 6, 57, 34, 27, 67, 38, 80, 79, 55, 53, 47, 72, 2, 5, 11, 3, 48

Anfang:	**36**	9	58	6	57	34	27	67	38	80	79	55	53	47	72	2	5	11	3	48
Vor Pivot:	**36**		3		11			5	**2**							38	67	57	58	
Nach Pivot:	**2**								**36**											
Vor Pivot:	**2**																			
Nach Pivot:	**2**																			
Vor Pivot:		**9**			**5**			11												
Nach Pivot:		**5**			**9**															
Vor Pivot:		**5**	**3**																	
Nach Pivot:		**3**	**5**																	
Vor Pivot:						**34**		**11**												
Nach Pivot:						**11**		**34**												
Vor Pivot:						**11**														
Nach Pivot:						**11**														
Vor Pivot:										**80**										**48**
Nach Pivot:										**48**										**80**
Vor Pivot:										**48**	38	**47**		55		79				
Nach Pivot:										**47**		**48**								
Vor Pivot:										**47**	**38**									
Nach Pivot:										**38**	**47**									
Vor Pivot:													**53**							
Nach Pivot:													**53**							
Vor Pivot:														**55**						
Nach Pivot:														**55**						
Vor Pivot:															**72**	58		**57**	79	
Nach Pivot:															**57**			**72**		
Vor Pivot:															**57**					
Nach Pivot:															**57**					
Vor Pivot:																**58**				
Nach Pivot:																**58**				
Ende:	**2**	3	5	6	9	11	27	34	36	38	47	48	53	55	57	58	67	72	79	80

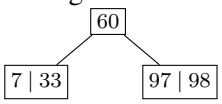
Aufgabe 12-3 B-Baum

Gegeben Sei ein leerer B-Baum der Ordnung 2.

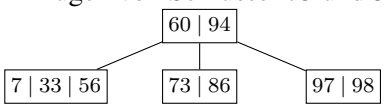
- Fügen Sie folgende Schlüssel (in dieser Reihenfolge) ein:
  98, 7, 97, 60, 33, 73, 56, 94, 86, 95, 67, 3, 44, 23, 19, 61, 25, 22, 4, 47, 79, 21
- Entfernen Sie folgende Schlüssel (in dieser Reihenfolge):
  60, 19, 3, 4, 98, 25, 33, 56, 86, 94, 23, 21, 67, 95, 97, 73, 79, 61, 44, 47, 7

Einzufügende Schlüssel: 98**,** 7**,** 97**,** 60**,** 33**,** 73**,** 56**,** 94**,** 86**,** 95**,** 67**,** 3**,** 44**,** 23**,** 19**,** 61**,** 25**,** 22**,** 4**,** 47**,** 79**,** 21

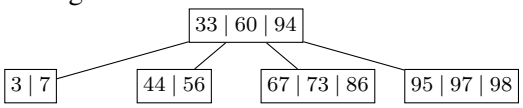
Einfügen von Schlüssel 98 und 7 und 97 und 60 und 33 (1 overflow)



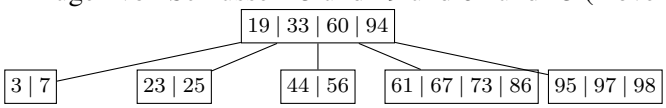
Einfügen von Schlüssel 73 und 56 und 94 und 86 (1 overflow)



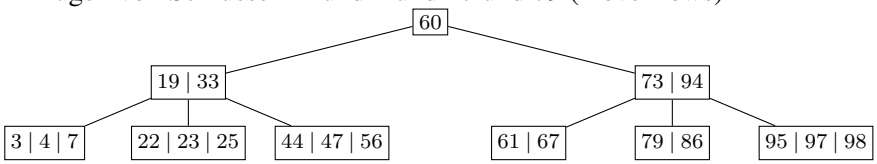
Einfügen von Schlüssel 95 und 67 und 3 und 44 (1 overflow)



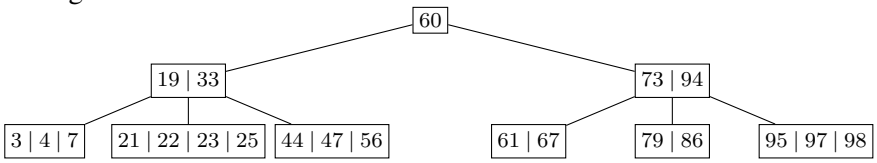
Einfügen von Schlüssel 23 und 19 und 61 und 25 (1 overflow)



Einfügen von Schlüssel 22 und 4 und 47 und 79 (2 overflows)

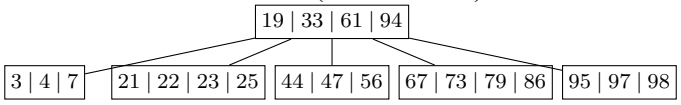


Einfügen von Schlüssel 21

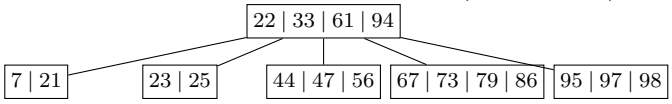


Zu Löschende Schlüssel: 60**,** 19**,** 3**,** 4**,** 98**,** 25**,** 33**,** 56**,** 86**,** 94**,** 23**,** 21**,** 67**,** 95**,** 97**,** 73**,** 79**,** 61**,** 44**,** 47**,** 7

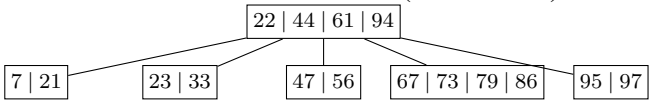
Löschen von Schlüssel 60 (3 underflows)



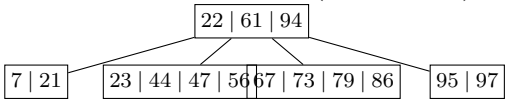
Löschen von Schlüssel 19 und 3 und 4 (1 underflow)



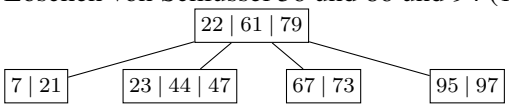
Löschen von Schlüssel 98 und 25 (1 underflow)



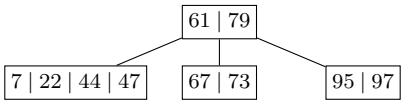
Löschen von Schlüssel 33 (1 underflow)



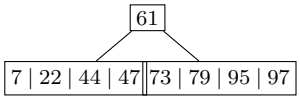
Löschen von Schlüssel 56 und 86 und 94 (1 underflow)



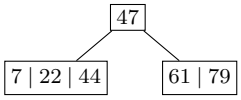
Löschen von Schlüssel 23 und 21 (1 underflow)



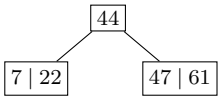
Löschen von Schlüssel 67 (2 underflows)



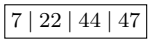
Löschen von Schlüssel 95 und 97 und 73 (1 underflow)



Löschen von Schlüssel 79 (1 underflow)



Löschen von Schlüssel 61 (2 underflows)



Löschen von Schlüssel 44 und 47 und 7 (1 underflow)

Aufgabe 12-4 Lineares Hashing mit Partiellen Erweiterungen

Gegeben Sei eine leere Hashtabelle mit der Strategie "Lineares Hashing mit Partiellen Erweiterungen" und folgenden Parametern:

Hashfunktion: $h_L(1,k) := k \bmod (2 \cdot 2^L)$
Hashfunktion: $h_L(2,k) := k \bmod (3 \cdot 2^L)$
Anfangsgröße: 2, Seitengröße: 2, Overflowseitengröße: 1
Füllgrad maximal: 0.70

Fügen Sie (in dieser Reihenfolge) folgende Objekte in die Hashtabelle ein:
35, 87, 78, 52, 22, 83, 40, 20, 27, 88, 32, 49, 4, 15, 14, 97, 95, 45, 76, 48

Einfügen von Schlüssel 35. Belegung nach Einfügen 0.25, nach Behandlung 0.25

_	35
_	_

Einfügen von Schlüssel 87. Belegung nach Einfügen 0.50, nach Behandlung 0.50

_	35
_	87

Einfügen von Schlüssel 78. Belegung nach Einfügen 0.75, nach Behandlung 0.50

78	_	35
87	_	_

Einfügen von Schlüssel 52. Belegung nach Einfügen 0.67, nach Behandlung 0.67

78	52	35
87	_	_

Einfügen von Schlüssel 22. Belegung nach Einfügen 0.83, nach Behandlung 0.62

52	_	78	87
_	_	22	35

Einfügen von Schlüssel 83. Belegung nach Einfügen 0.67, nach Behandlung 0.67

52	_	78	87
_	_	22	35
			↓
			83

Einfügen von Schlüssel 40. Belegung nach Einfügen 0.78, nach Behandlung 0.58

78	_	_	87	52
_	_	_	35	40
			↓	↓
			83	22

Einfügen von Schlüssel 20. Belegung nach Einfügen 0.67, nach Behandlung 0.67

78	_	20	87	52
_	_	_	35	40
			↓	↓
			83	22

Einfügen von Schlüssel 27. Belegung nach Einfügen 0.69, nach Behandlung 0.69

78	_	20	87	52
_	_	_	35	40
			↓	↓
			83	22
			↓	
			27	

Einfügen von Schlüssel 88. Belegung nach Einfügen 0.71, nach Behandlung 0.71

78	_	20	87	52	35
_	_	_	27	40	83
				↓	
				22	
				↓	
				88	

Einfügen von Schlüssel 32. Belegung nach Einfügen 0.79, nach Behandlung 0.73

32	_	_	87	20	35	78
40	_	_	27	52	83	22
↓						
88						

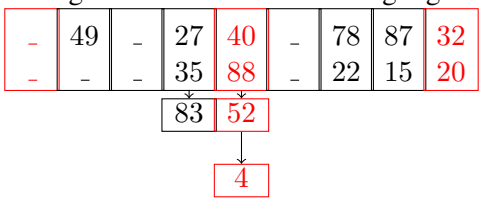
Einfügen von Schlüssel 49. Belegung nach Einfügen 0.80, nach Behandlung 0.67

32	49	_	27	20	_	78	87
40	_	_	35	52	_	22	_
↓			↓				
88			83				

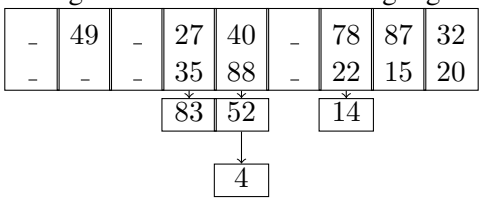
Einfügen von Schlüssel 4. Belegung nach Einfügen 0.68, nach Behandlung 0.68

32	49	_	27	20	_	78	87
40	_	_	35	52	_	22	_
↓			↓	↓			
88			83	4			

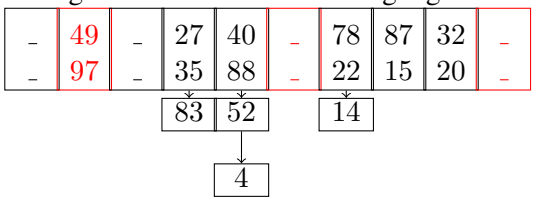
Einfügen von Schlüssel 15. Belegung nach Einfügen 0.74, nach Behandlung 0.67



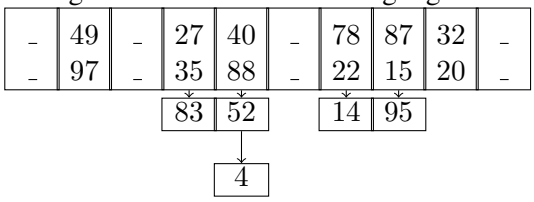
Einfügen von Schlüssel 14. Belegung nach Einfügen 0.68, nach Behandlung 0.68



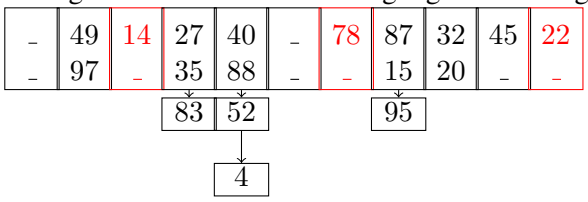
Einfügen von Schlüssel 97. Belegung nach Einfügen 0.73, nach Behandlung 0.67



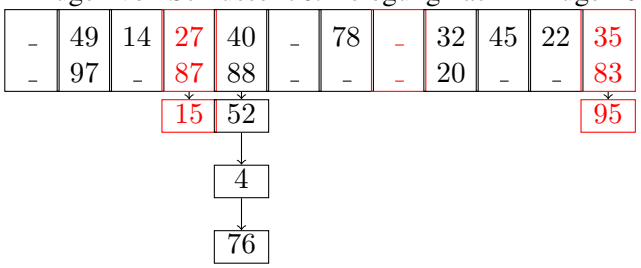
Einfügen von Schlüssel 95. Belegung nach Einfügen 0.68, nach Behandlung 0.68



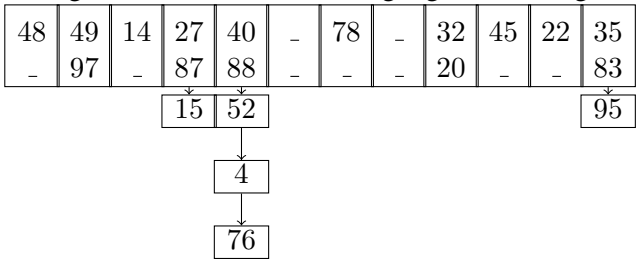
Einfügen von Schlüssel 45. Belegung nach Einfügen 0.72, nach Behandlung 0.69



Einfügen von Schlüssel 76. Belegung nach Einfügen 0.70, nach Behandlung 0.66

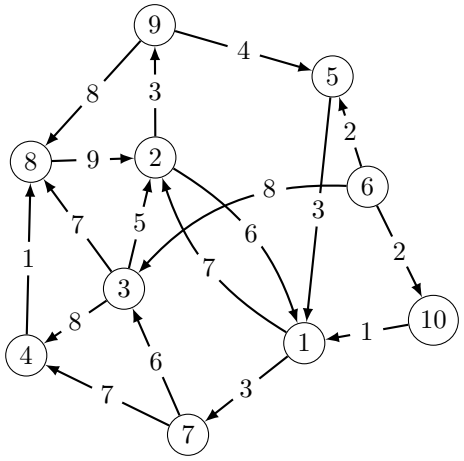


Einfügen von Schlüssel 48. Belegung nach Einfügen 0.69, nach Behandlung 0.69



Aufgabe 12-5 Graphalgorithmen

Gegeben ist folgender Graph:



(a) Erstellen Sie die Adjazenzmatrix und Adjazenzlisten des Graphen.
Auf der Diagonalen und statt ∞ dürfen Sie auch $-$ schreiben.

(b) Führen Sie, ausgehend vom Knoten **6** einen Breitendurchlauf und einen Tiefendurchlauf durch den Graphen durch. Zeichnen Sie jeweils den Ergebnisspannbaum, der sich aus Breiten- und Tiefendurchlauf ergibt. Erstellen Sie den Spannbaum so, dass bei Mehrdeutigkeiten zuerst der Knoten mit der kleinsten Knoten-ID besucht wird.

(c) Berechnen Sie die kürzesten Wege ausgehend von Knoten **6** zu allen anderen Knoten. Verwenden Sie dazu den Algorithmus von Dijkstra. Zeichnen Sie für jeden Schritt einen Baum, der nur die kürzesten Wege enthält. Kennzeichnen Sie besuchte Knoten, um sie von "offenen" Knoten zu unterscheiden.

(d) Wenden Sie den Algorithmus von Floyd auf den Graphen an. Füllen Sie entsprechend dem Algorithmus die Kostenmatrix A. Die Einträge von A sollen dabei nicht nur die Kosten, sondern zusätzlich den Nachfolgerknoten (= den nächsten Schritt) des kostenminimalen Pfades in Klammern beinhalten. Die Kostenmatrix und die *pathCost*-Matrix werden somit in *einer* Tabelle dargestellt.

(e) Betrachten Sie den Graphen nun als ungerichteten Graphen. Konstruieren Sie mit Hilfe des Algorithmus von Kruskal den minimalen Spannbaum. Entscheiden Sie, ob das Ergebnis des Algorithmus eindeutig ist und begründen Sie Ihre Antwort.

(a)

Adjazenzmatrix:

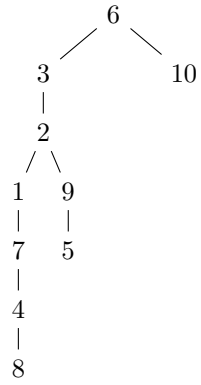
	1	2	3	4	5	6	7	8	9	10
1	–	7	–	–	–	–	3	–	–	–
2	6	–	–	–	–	–	–	–	3	–
3	–	5	–	8	–	–	–	7	–	–
4	–	–	–	–	–	–	–	1	–	–
5	3	–	–	–	–	–	–	–	–	–
6	–	–	8	–	2	–	–	–	–	2
7	–	–	6	7	–	–	–	–	–	–
8	–	9	–	–	–	–	–	–	–	–
9	–	–	–	–	4	–	–	8	–	–
10	1	–	–	–	–	–	–	–	–	–

Adjazenzlisten:

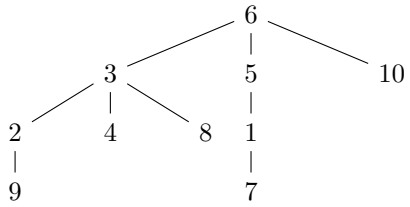
	Gerichtet	Ungerichtet (Min)
1	2 (7), 7 (3)	2 (6), 5 (3), 7 (3), 10 (1)
2	1 (6), 9 (3)	1 (6), 3 (5), 8 (9), 9 (3)
3	2 (5), 4 (8), 8 (7)	2 (5), 4 (8), 6 (8), 7 (6), 8 (7)
4	8 (1)	3 (8), 7 (7), 8 (1)
5	1 (3)	1 (3), 6 (2), 9 (4)
6	3 (8), 5 (2), 10 (2)	3 (8), 5 (2), 10 (2)
7	3 (6), 4 (7)	1 (3), 3 (6), 4 (7)
8	2 (9)	2 (9), 3 (7), 4 (1), 9 (8)
9	5 (4), 8 (8)	2 (3), 5 (4), 8 (8)
10	1 (1)	1 (1), 6 (2)

(b) Spannbäume, beginnend bei 6:

Tiefensuche:

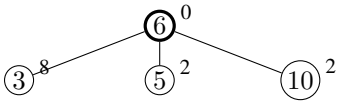


Breitensuche:

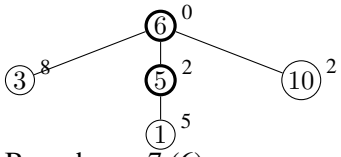


(c) Dijkstra, beginnend bei 6:

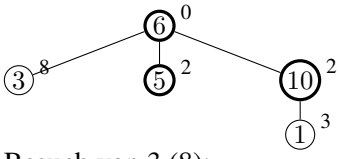
Besuch von 6 (0):



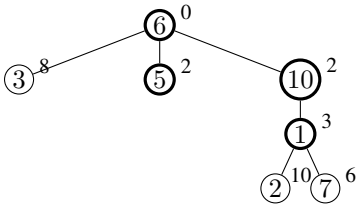
Besuch von 5 (2):



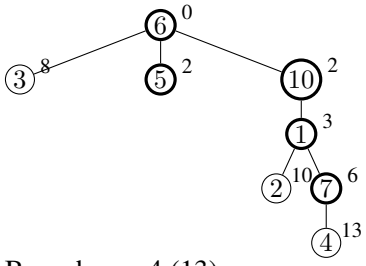
Besuch von 10 (2):



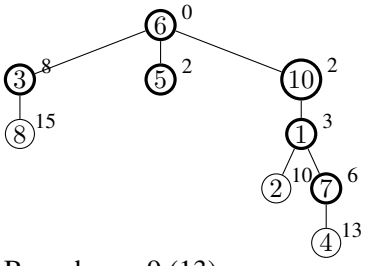
Besuch von 1 (3):



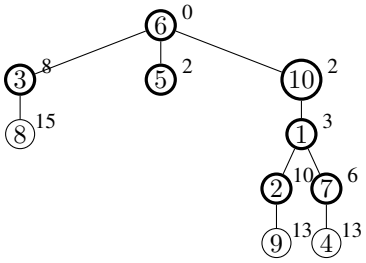
Besuch von 7 (6):



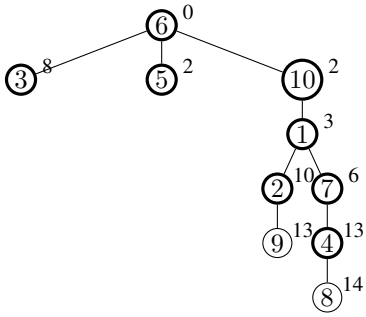
Besuch von 3 (8):



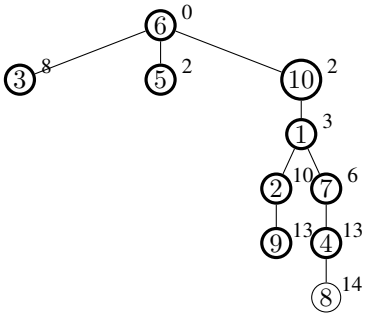
Besuch von 2 (10):



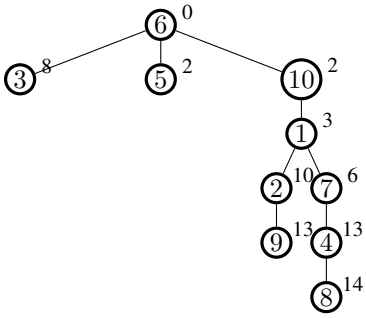
Besuch von 4 (13):



Besuch von 9 (13):



Besuch von 8 (14): trivial



(d) Floyd:

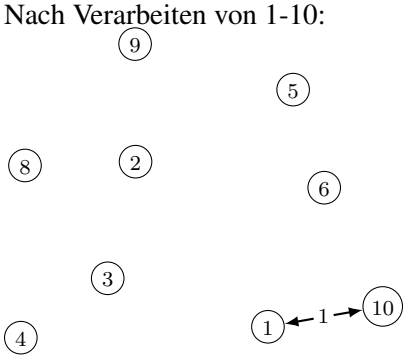
Knoten	Neue Kanten	Geänderte Kanten
1	(2,7,9) (5,2,10) (5,7,6) (10,2,8) (10,7,4)	
2	(1,9,10) (3,1,11) (3,7,14) (3,9,8) (5,9,13) (8,1,15) (8,7,18) (8,9,12) (10,9,11)	
3	(6,1,19) (6,2,13) (6,4,16) (6,7,22) (6,8,15) (6,9,16) (7,1,17) (7,2,11) (7,8,13) (7,9,14)	
4		(7,8,8)
5	(9,1,7) (9,2,14) (9,7,10)	(6,1,5) (6,2,12) (6,7,8) (6,9,15)
6		
7	(1,3,9) (1,4,10) (1,8,11) (2,3,15) (2,4,16) (2,8,17) (5,3,12) (5,4,13) (5,8,14) (8,3,24) (8,4,25) (9,3,16) (9,4,17) (10,3,10) (10,4,11) (10,8,12)	(6,4,15)
8	(4,1,16) (4,2,10) (4,3,25) (4,7,19) (4,9,13)	
9	(1,5,14) (2,5,7) (3,5,12) (4,5,17) (7,5,18) (8,5,16) (10,5,15)	(2,8,11)
10		(6,1,3) (6,2,10) (6,4,13) (6,7,6) (6,8,14) (6,9,13)

Resultierende Adjazenzmatrix:

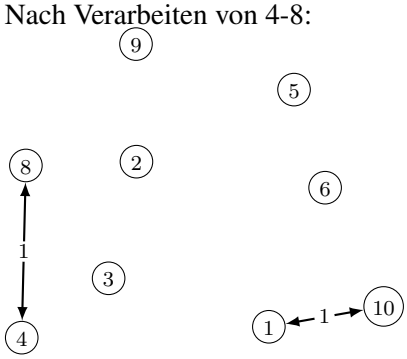
	1	2	3	4	5	6	7	8	9	10
1	–	7 (-)	9 (7)	10 (7)	14 (9)	–	3 (-)	11 (7)	10 (2)	–
2	6 (-)	–	15 (7)	16 (7)	7 (9)	–	9 (1)	11 (9)	3 (-)	–
3	11 (2)	5 (-)	–	8 (-)	12 (9)	–	14 (2)	7 (-)	8 (2)	–
4	16 (8)	10 (8)	25 (8)	–	17 (9)	–	19 (8)	1 (-)	13 (8)	–
5	3 (-)	10 (1)	12 (7)	13 (7)	–	–	6 (1)	14 (7)	13 (2)	–
6	3 (10)	10 (10)	8 (-)	13 (10)	2 (-)	–	6 (10)	14 (10)	13 (10)	2 (-)
7	17 (3)	11 (3)	6 (-)	7 (-)	18 (9)	–	–	8 (4)	14 (3)	–
8	15 (2)	9 (-)	24 (7)	25 (7)	16 (9)	–	18 (2)	–	12 (2)	–
9	7 (5)	14 (5)	16 (7)	17 (7)	4 (-)	–	10 (5)	8 (-)	–	–
10	1 (-)	8 (1)	10 (7)	11 (7)	15 (9)	–	4 (1)	12 (7)	11 (2)	–

(e) Kruskal, auf nicht gerichtetem Graphen (Minimum, falls zwei Kanten):

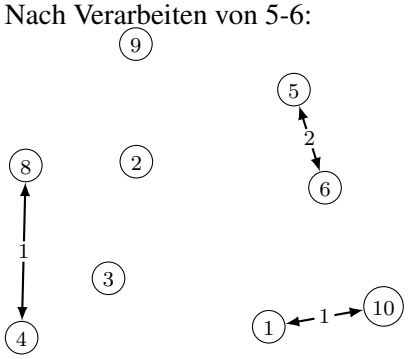
Nach Verarbeiten von 1-10:



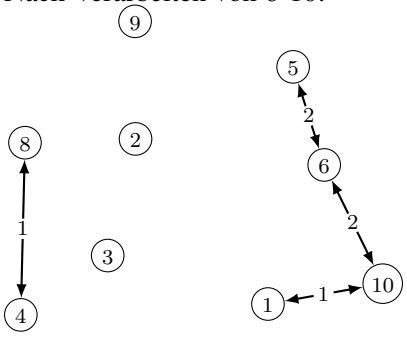
Nach Verarbeiten von 4-8:



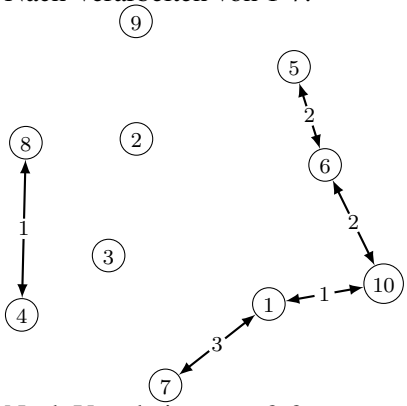
Nach Verarbeiten von 5-6:



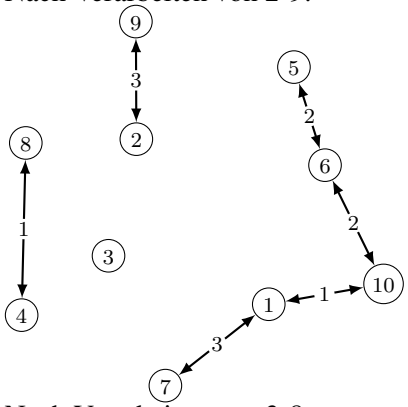
Nach Verarbeiten von 6-10:



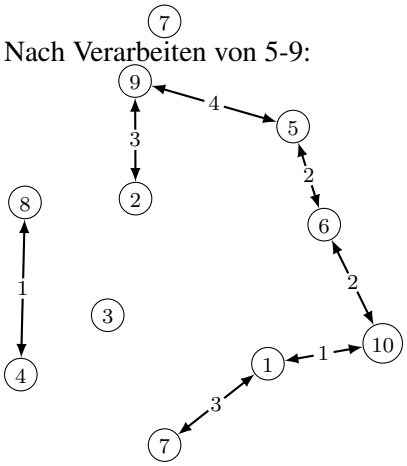
Nach Verarbeiten von 1-7:



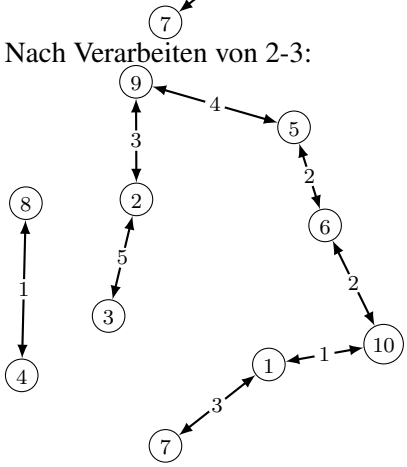
Nach Verarbeiten von 2-9:



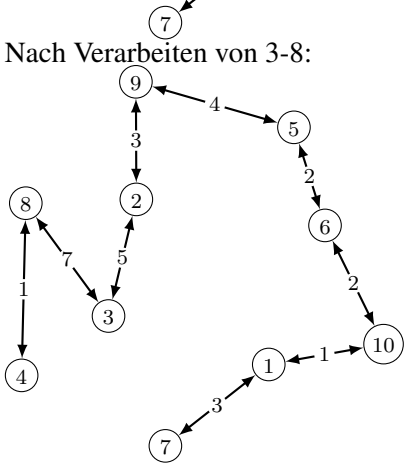
Nach Verarbeiten von 5-9:



Nach Verarbeiten von 2-3:



Nach Verarbeiten von 3-8:



ODER: Nach Verarbeiten von 4-7: Insbesondere ist das Ergebnis also hier nicht eindeutig!