Technische Universität Darmstadt — Fachbereich Informatik

Abschlussklausur / Fachprüfung

Algorithmen und Datenstrukturen

Wintersemester 2020/2021

09.04.2021 — 09:00–11:00 Uhr

Hinweise:

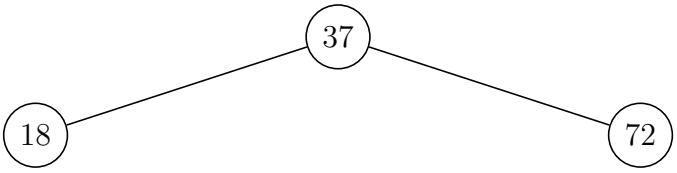
- Als Schreibmittel ist nur ein **schwarzer oder blauer Schreibstift** erlaubt. Verwenden Sie im Besonderen keine Bleistifte, Rot- oder Korrekturstifte, sowie keine Korrekturflüssigkeit oder Tintenlöscher. Streichen Sie deutlich durch, was nicht gewertet werden soll.
- Als Hilfsmittel können Sie unsere Folien, Übungen, sonstige Notizen, Bücher, Wörterbücher und anderweitige Literatur benutzen. **Nicht zugelassen** sind jegliche elektronischen Geräte (Taschenrechner, Laptop, Mobiltelefon etc.); Mobiltelefone und Smartwatches müssen ausgeschaltet sein und dürfen nicht unmittelbar bei sich getragen werden.
- Füllen Sie das Deckblatt **vollständig und gut lesbar** aus.
- Schreiben Sie auf **jedes** Aufgabenblatt Ihren Namen und Ihre Matrikelnummer.
- Schreiben Sie Ihre Lösung in den Zwischenraum unter der jeweiligen Aufgabe. Nutzen Sie gegebenenfalls die leeren Blätter am Ende der Klausur. Bei Bearbeitung auf einer anderen Seite **geben Sie die Seitenzahl deutlich an**.
- Geben Sie zu jeder Aufgabe **nur eine Lösung** ab. Abgabe von mehreren Varianten macht die gesamte Lösung ungültig. Streichen Sie deutlich durch, was nicht gewertet werden soll.
- Falls dies ihr **Drittversuch** ist, dann vermerken Sie dies auf dem Deckblatt, indem Sie die Box unten ankreuzen.
- Zum Bestehen der Klausur sind 50 von insgesamt 100 Punkten hinreichend.

Nachname		
Vorname		
Matrikelnr.		
Studiengang		☐ Drittversuch

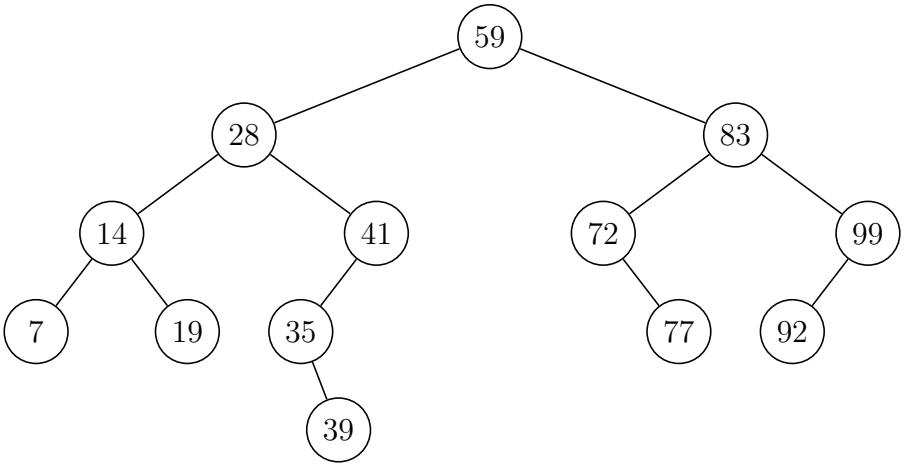
Aufgabe	1	2	3	4	5	6	7	$\sum$
Max. Punkte	10	12	15	23	17	12	11	100
Punkte								
Kürzel								

1 Binäre Suchbäume (__ / 10 P)

1. Fügen Sie **nacheinander** die folgenden Werte in den gegebenen binären Suchbaum ein: 25, 19, 5, 47, 36, 59, 10, 38. Verwenden Sie dabei den Algorithmus aus der Vorlesung zum Einfügen von Knoten in einen binären Suchbaum. Sie dürfen den angegebenen binären Suchbaum vervollständigen. (__ / 2 P)



2. Sei T der folgende binäre Suchbaum. Bestimmen Sie die Ausgabe der Algorithmen $\textsc{Preorder}(T.\texttt{root})$, $\textsc{Postorder}(T.\texttt{root})$ und $\textsc{Inorder}(T.\texttt{root})$, und geben Sie diese unten an. (__ / 3 P)



$\textsc{Preorder}(T.\texttt{root})$:

$\textsc{Postorder}(T.\texttt{root})$:

$\textsc{Inorder}(T.\texttt{root})$:

3. Löschen Sie den Knoten 28 aus dem Baum aus Teilaufgabe 2. Verwenden Sie dabei den Algorithmus aus der Vorlesung zum Löschen von Knoten aus einem binären Suchbaum. Zeichnen Sie den **vollständigen** Baum nach der Löschoperation. (__ / 2 P)

4. Sei T ein beliebiger binärer Suchbaum ohne doppelt vorkommende Werte. Ist die Löschoperation auf T immer kommutativ? Konkret heißt das: Erhält man den gleichen Baum, wenn man entweder zuerst Knoten x und dann Knoten y entfernt, wie wenn man erst y und dann x entfernt? Wenn ja, argumentieren Sie, warum, wenn nein, geben Sie ein Gegenbeispiel an. (__ / 3 P)

2 Datenstrukturen, die keine Bäume sind (__ / 12 P)

2.1 Verständnisfragen (__ / 4 P)

1. Was ist die maximale Höhe einer **Skip-Liste** mit n Elementen und Wahrscheinlichkeit $p > 0$ für übergeordnete Listen? Begründen Sie Ihre Antwort. (__ / 1 P)

2. Sie überprüfen, ob ein gegebener **Bloomfilter** aussagt, dass ein spezifisches Element darin vorhanden sei. Was sagt ein positives bzw. negatives Ergebnis darüber aus, ob dieses Element in den Bloom-Filter eingefügt wurde? (__ / 1 P)

3. Seien BF_1 und BF_2 zwei **Bloomfilter** gleicher Größe und mit den gleichen Hashfunktionen. Sie berechnen nun BF_{AND} als das logische Und der beiden Bloomfilter – $BF_{\text{AND}}[i]$ wird also auf 1 gesetzt, wenn sowohl $BF_1[i]$ als auch $BF_2[i]$ auf 1 gesetzt war. Entspricht BF_{AND} dem **Schnitt** von BF_1 und BF_2? Der Schnitt ist hierbei definiert als der Bloomfilter, dem alle Elemente hinzugefügt werden, die sowohl BF_1 als auch BF_2 hinzugefügt wurden. **Begründen** Sie Ihre Antwort. (__ / 2 P)

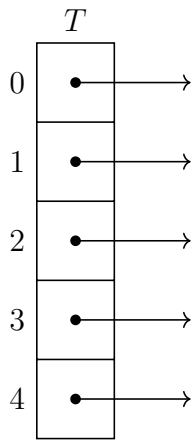
2.2 Hash-Tabellen (__ / 2 P)

Gegeben sei eine Hash-Tabelle mit einer Kapazität von 5, die Zahlen abspeichert. Die verwendete Hashfunktion $h(n)$ ist dabei gegeben als

$$h(n) = 3q(n) + 1 \mod 5,$$

wobei $q(n)$ die Quersumme der Zahl, also die Summe aller Ziffern berechnet (beispielsweise wäre die Quersumme von 76 gleich 13). Die Hash-Tabelle arbeitet mit verketteten Listen für Konflikte, wobei neue Elemente an das **Ende** der Liste gehängt werden.

Fügen Sie die folgenden Zahlen in die Hash-Tabelle nacheinander ein, indem Sie die gegebene Zeichnung ergänzen: 13, 56, 122, 321, 2001, 642, 79, 472.



2.3 Queues (__ / 3 P)

Eine Queue Q ist auf einem **virtuell zyklischen** Array der Größe 5 implementiert. Es wurden bereits die Werte 9, 2, 7 und 3 nacheinander in die zu Beginn leere Queue eingefügt. Führen Sie nacheinander die gegebenen Operationen auf Q durch und geben Sie jeweils das daraus resultierende Array an. Verwenden Sie das Symbol " $-$ ", um alle nicht verwendeten Felder im Array zwischen Q.`front` und Q.`rear` zu kennzeichnen.

	9	2	7	3	$-$
`enqueue`$(Q, 1)$:					
`dequeue`(Q):					
`dequeue`(Q):					
`enqueue`$(Q, 6)$:					
`dequeue`(Q):					
`enqueue`$(Q, 8)$:					

2.4 Bloomfilter mit Löschoperation (__ / 3 P)

Implementieren Sie nun einen Bloomfilter der Länge n mit k Hashfunktionen $H_1, \ldots, H_k$, der auch das **Löschen** von Elementen erlaubt. Dazu dürfen Sie in BF an jeder Position statt eines Bits eine beliebige nichtnegative Zahl speichern. Ihre Implementierung sollte alle anderen Eigenschaften des Bloomfilters erhalten und keine schlechtere Fehlerwahrscheinlichkeit haben. Die Initfunktion BloomInit wurde Ihnen bereits vorgegeben, implementierten Sie nun die Funktionen BloomAdd, BloomSearch und BloomDelete.

BloomInit(BF, n)

```
1 : for i = 0 to n − 1 do
2 :     BF[i] = 0
```

BloomSearch($BF, value$)

```
1 :
2 :
3 :
4 :
5 :
```

BloomAdd($BF, value$)

```
1 :
2 :
3 :
4 :
5 :
```

BloomDelete($BF, value$)

```
1 :
2 :
3 :
4 :
5 :
```

3 Sortieren (__ / 15 P)

1. Im Folgenden ist der Sortieralgorithmus INSERTION-SORT lückenhaft dargestellt. Füllen Sie die fehlenden Stellen aus, sodass Sie einen funktionierenden INSERTION-SORT Algorithmus haben, der in **aufsteigender** Reihenfolge sortiert. (__ / 2 P)

INSERTION-SORT(A)

```
1 : for j = 1 to A.length − 1 do
2 :     key = A[j]
3 :     i = j − 1
4 :     while i ≥ 0 and _____ do
5 :         A[i + 1] = A[i]
6 :         _____
7 :     A[i + 1] = key
```

2. Erklären Sie in maximal **drei** Sätzen das Prinzip Divide-and-Conquer. (__ / 1,5 P)

3. Gegeben sei ein beliebiges Array A der Länge 5. Sie führen SELECTION-SORT in Form von MIN-SORT auf diesem Array aus. Geben Sie jeweils für $i = 0, \ldots, |A| - 2$ die Anzahl der jeweiligen Vergleiche an und leiten Sie daraus eine allgemeine Formel für die Anzahl der Vergleiche her. (__ / 2,5 P)

4. Führen Sie die Unterroutine PARTITION in Quicksort auf dem Array

$$A[37 \ldots 42] = [12, 18, 2, 16, 8, 13]$$

aus.
Verwenden Sie dabei den Algorithmus aus der Vorlesung, und wählen Sie als Pivotelement das letzte Element des Arrays. Der Algorithmus durchläuft in einer **for**-Schleife nacheinander alle Elemente in A. Bestimmen Sie den Zustand des Arrays A nach jedem Durchlauf der **for**-Schleife, sowie direkt vor dem Ende von PARTITION. Geben Sie zuerst den Wert von i an vor der Ausführung der **for**-Schleife sowie den Wert von j in der jeweiligen Iteration der **for**-Schleife, den Wert von i am Ende der j-ten Iteration der Schleife, sowie den von PARTITION ausgegebenen Index $i+1$ an. (__ / 6 P)

$[12, 18, 2, 16, 8, 13]$ $i =$ ______

$[\ \ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ ]$ $i =$ ______ , $j =$ ______

$[\ \ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ ]$ $i =$ ______ , $j =$ ______

$[\ \ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ ]$ $i =$ ______ , $j =$ ______

$[\ \ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ ]$ $i =$ ______ , $j =$ ______

$[\ \ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ ]$ $i =$ ______ , $j =$ ______

$[\ \ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ ]$ $i + 1 =$ ______

5. Nachfolgend sehen Sie drei Folgen gewisser Momentaufnahmen von jeweils einem der vier Algorithmen (a) INSERTION-SORT, (b) MERGE-SORT, (c) BUBBLE-SORT und (d) SELECTION-SORT, wie Sie in der Vorlesung vorgestellt wurden. Geben Sie unter der jeweiligen Folge jeweils den Namen des zugehörigen Algorithmus an. (__ / 3 P)

5	3	2	1	4
5	3	1	2	4
1	5	3	2	4
1	2	5	3	4
1	2	3	5	4
1	2	3	4	5

_____

5	3	2	1	4
1	3	2	5	4
1	2	3	5	4
1	2	3	4	5

_____

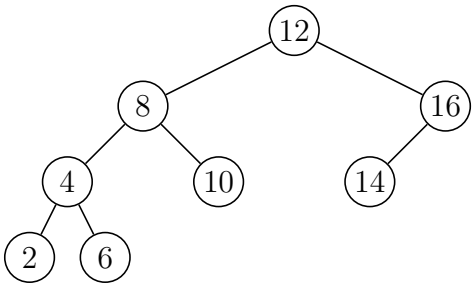
5	3	2	1	4
3	5	2	1	4
2	3	5	1	4
1	2	3	5	4
1	2	3	4	5

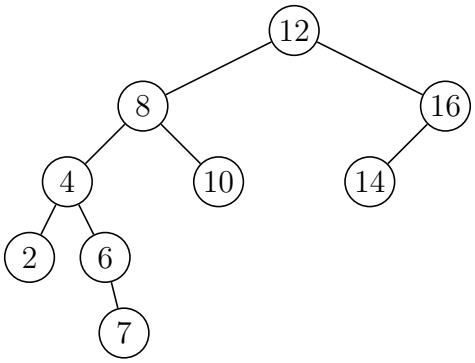
_____

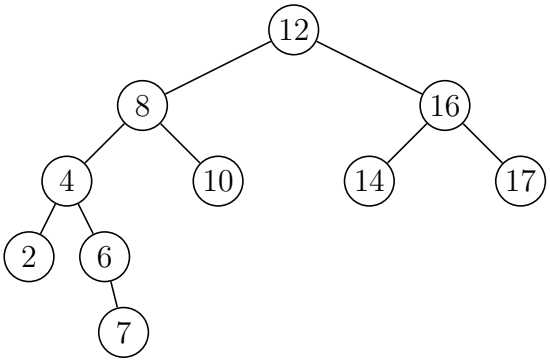
4 Bäume (__ / 23 P)

4.1 Bäume erkennen (__ / 4,5 P)

Bestimmen Sie für jeden der folgenden Binärbäume, ob es sich um einen binären Suchbaum oder um einen AVL-Baum handelt, oder ob es für ihn eine Rot-Schwarz-Färbung gibt, oder ob keine dieser Möglichkeiten zutrifft. Schreiben Sie dazu "BST", "AVL", "RBT", oder "Keine" in das Feld unter der jeweiligen Abbildung.
Machen Sie immer die genauste Angabe: Wenn also beispielsweise ein Baum ein binärer Suchbaum und ein AVL-Baum ist, dann schreiben Sie "AVL" und nicht "BST". Es ist immer nur eine Antwort korrekt.







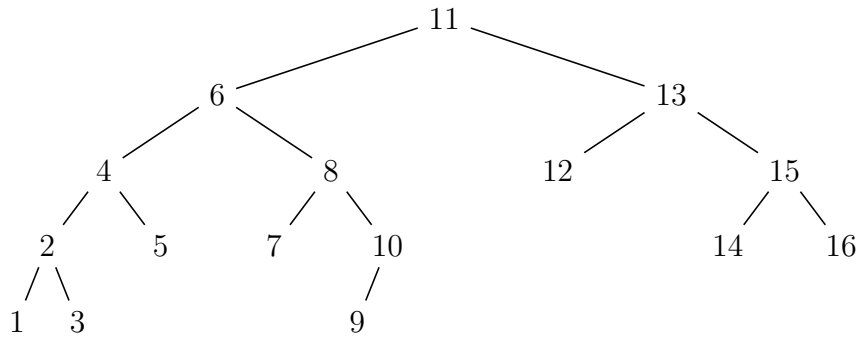
4.2 Rot-Schwarz-Bäume (__ / 3 P)

In dieser Teilaufgabe wird die folgende Notation für rot und schwarz gefärbte Knoten verwendet: Rote Knoten werden durch ein Rechteck gekennzeichnet, und schwarze Knoten durch einen Kreis markiert. Definieren Sie diese Symbole **nicht** um.

Roter Knoten

Schwarzer Knoten

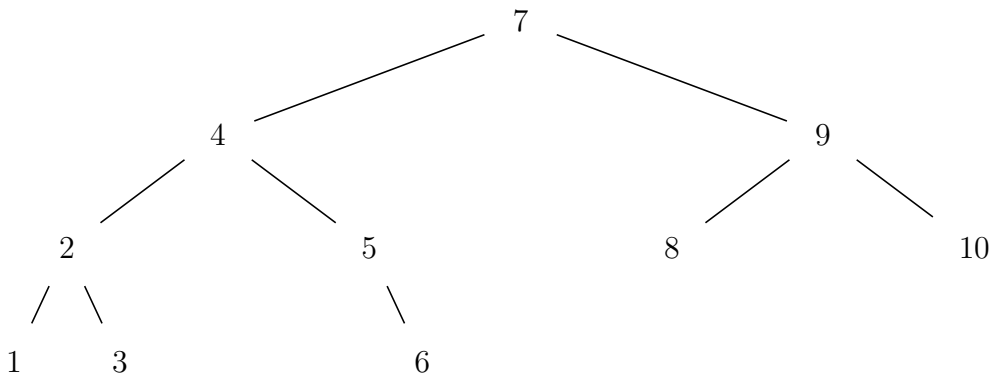
1. Bestimmen Sie die **Anzahl** F der möglichen Rot-Schwarz-Färbungen für den folgenden binären Suchbaum. (__ / 2 P)



$F =$ _____

(Platz für Skizzen)

2. Gegeben sei der folgende binäre Suchbaum. Bestimmen Sie eine Färbung der Knoten, sodass der Baum zu einem Rot-Schwarz-Baum mit Schwarzhöhe 2 wird. Vervollständigen Sie jeden Knoten entsprechend durch ein Rechteck oder einen Kreis. (__ / 1 P)



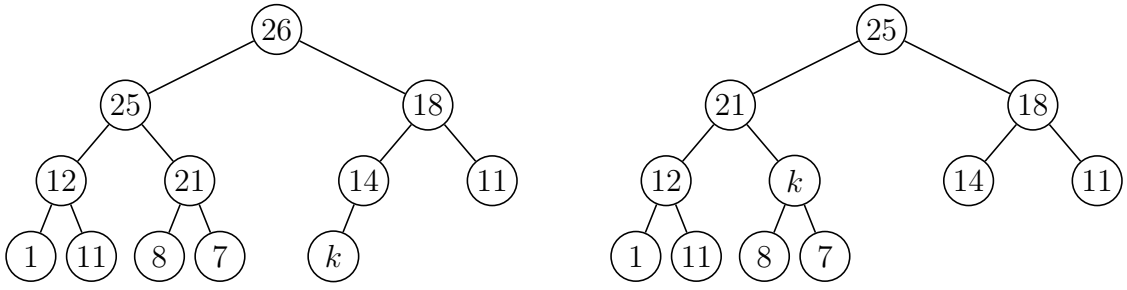
4.3 Heaps (__ / 3 P)

1. Bestimmen Sie die **Anzahl** N_5 der unterschiedlichen binären Max-Heaps, welche genau die Schlüssel in der Menge $\{1, 2, 3, 4, 5\}$ ohne Wiederholungen verwalten. (__ / 1,5 P)

$$N_5 = \_\_\_\_\_$$

(Platz für Skizzen)

2. Es sei H das binäre Max-Heap unten links. Bestimmen Sie alle möglichen Schlüsselwerte $k \in \mathbb{N}$, sodass die Operation $\textsc{Extract-Max}(H)$ das binäre Max-Heap unten rechts liefert. (__ / 1,5 P)



$$k \in \{\_\_\_\_\_\}$$

4.4 AVL-Bäume (__ / 4,5 P)

1. Wir nennen einen Knoten v in einem Binärbaum T ein "Einzelkind", wenn v einen Elternknoten, aber keinen Geschwisterknoten hat. Es bezeichne $EK(T)$ die Anzahl der Einzelkinder in T. Beweisen Sie die folgende Aussage: (__ / 2,5 P)

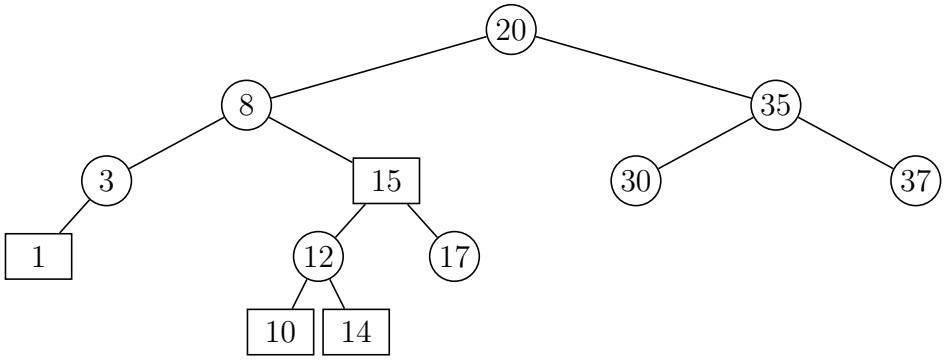
   Sei T ein AVL-Baum mit n Knoten. Dann gilt $EK(T)/n \leq 1/2$.

2. Zeichnen Sie einen AVL-Baum T mit den folgenden Eigenschaften: (__ / 2 P)

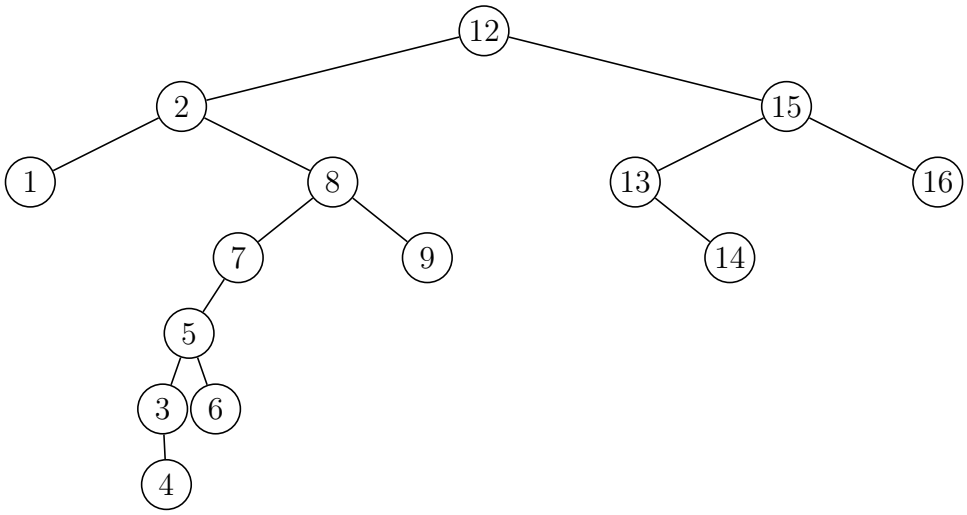
   - Die Höhe von T ist 4;
   - $B(v) = -1$ für jeden inneren Knoten v von T;
   - Die Schlüssel in T sind die Zahlen $1, \ldots, n$ für ein gewisses $n \in \mathbb{N}$.

4.5 Einfügen und Löschen (__ / 8 P)

1. Fügen Sie einen Knoten mit dem Schlüssel 9 in den folgenden Rot-Schwarz-Baum ein. Verwenden Sie dabei den Algorithmus aus der Vorlesung zum Einfügen von Knoten in einen Rot-Schwarz-Baum. Benutzen Sie die Notation aus Teilaufgabe 4.2.
   Zeichnen Sie Ihr Zwischenergebnis **vollständig** und **jeweils** nach dem Einfügen, sowie nach jeder Iteration der **while**-Schleife in FixColorsAfterInsertion und gegebenenfalls nach dem letzten Umfärben der Wurzel. (__ / 3 P)



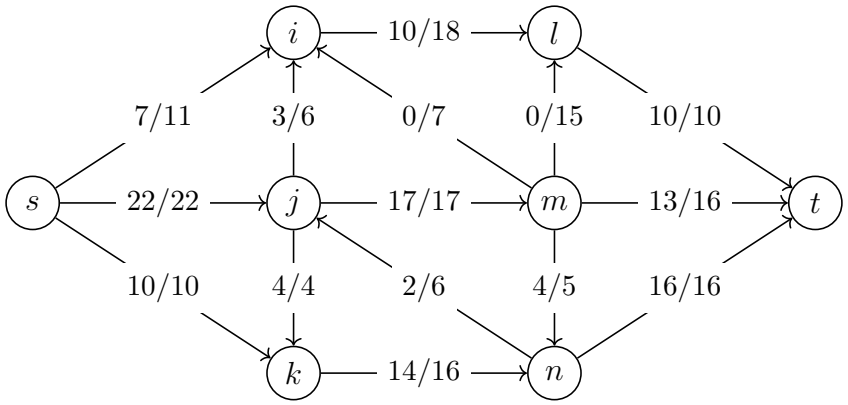
2. Löschen Sie den Knoten mit dem Schlüssel 7 aus dem folgenden Splay-Baum. Verwenden Sie dabei den Algorithmus aus der Vorlesung zum Löschen von Knoten aus einem Splay-Baum.
   Zeichnen Sie Ihr Zwischenergebnis **vollständig** und **jeweils** nach jeder erfolgten Zig-Zig-, Zig-Zag-, und Zig-Operation, sowie nach dem Löschen des Knotens und Aufteilen des Baumes, und nach dem Zusammenführen der zwei Teilbäume. (__ / 5 P)



5 Graphenalgorithmen (__ / 17 P)

5.1 Maximale Flüsse (__ / 4 P)

Betrachten Sie das folgende Flussnetzwerk N mit Quelle s und Senke t, sowie den angegebenen Fluss f für N.



1. Bestimmen Sie den Wert $|f|$ von f. (__ / 1 P)

$$|f| = \_\_\_\_\_$$

2. Führen Sie eine Iteration des Algorithmus von Ford-Fulkerson auf N aus. Beginnen Sie mit dem Fluss f und geben Sie die Knoten eines (zyklenfreien) hinzugefügten Pfades in einem Array an. (__ / 2 P)

Hinzugefügter Pfad $= [\_\_\_\_\_]$

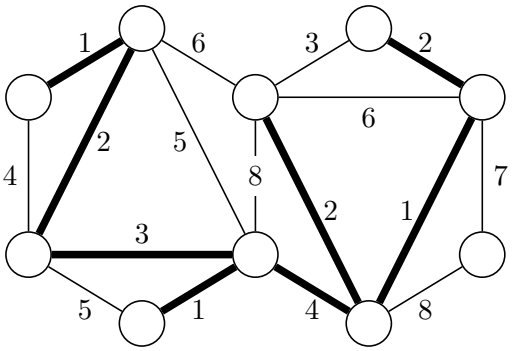
3. Bestimmen Sie den Wert eines maximalen Flusses $f_{\max}$ für N. (__ / 1 P)

$$|f_{\max}| = \_\_\_\_\_$$

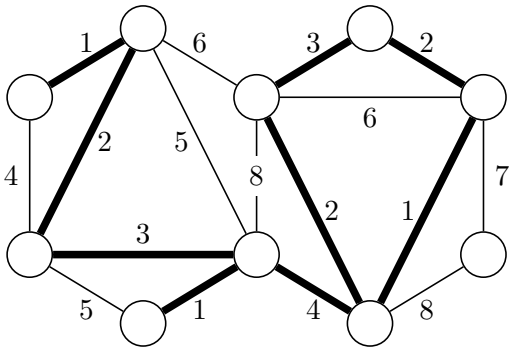
(Platz für Skizzen)

5.2 Minimale Spannbäume (__ / 6 P)

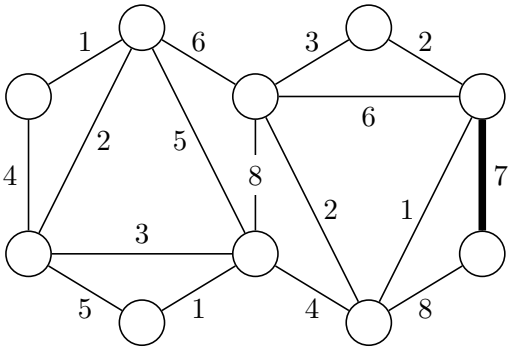
Es folgen vier Darstellungen eines Graphen, in denen jeweils einige Kanten fett gedruckt hervorgehoben sind. Bestimmen Sie jeweils, ob die hervorgehobene Kantenmenge durch eine (eventuell frühzeitig abgebrochene) Ausführung des Algorithmus von Kruskal oder von Prim entsteht, oder ob beide oder keine dieser Möglichkeiten zutrifft. Schreiben Sie je nachdem "Kruskal", "Prim", "Beide", oder "Keine" in das Feld unter der jeweiligen Abbildung.
Machen Sie immer die genauste Angabe: Wenn also beispielsweise eine Kantenmenge durch beide Algorithmen entstehen kann, dann schreiben Sie "Beide" und nicht "Kruskal" oder "Prim". Es ist immer nur eine Antwort korrekt.



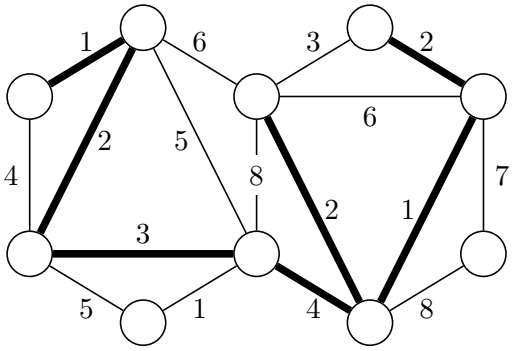
_____



_____



_____



_____

(Platz für Skizzen)

5.3 Algorithmus von Dijkstra (__ / 7 P)

Betrachten Sie den gewichteten Graphen $G = (V, E, w)$ mit $V = \{a, b, c, d, e, f, g, h\}$, dessen gewichtete Kanten durch folgende Matrix beschrieben sind:

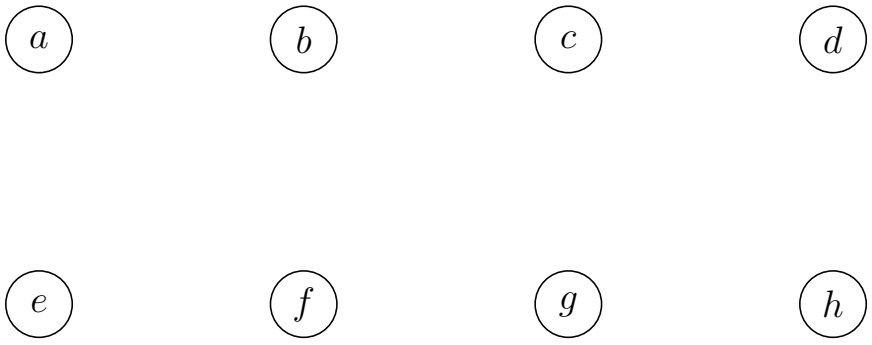
	a	b	c	d	e	f	g	h
a	□	1	17	21	3	10	7	15
b	13	□	19	□	2	5	33	□
c	32	7	□	1	4	20	33	5
d	73	42	□	□	38	54	□	3
e	6	9	2	15	□	17	20	2
f	□	21	17	4	60	□	46	□
g	26	38	42	2	20	40	□	1
h	65	37	55	1	36	45	5	□

Hier bedeutet eine Zahl k in Zeile i und Spalte j, dass $(i, j) \in E$ und $w(i, j) = k$. Ein Kästchen " □ " an der entsprechenden Stelle bedeutet, dass $(i, j) \notin E$.

1. Führen Sie den Algorithmus von Dijkstra auf diesem Graphen aus. Beginnen Sie im Knoten d. Füllen Sie dazu die Tabelle auf der **nächsten Seite** aus. Diese enthält eine separate Zeile für jede Iteration der **while**-Schleife im Algorithmus. Geben Sie in jeder Zeile an, welcher Knoten r im gegebenen Schritt aus der Menge Q extrahiert wird, sowie die Menge Q am Ende des betrachteten Schrittes. Bestimmen Sie außerdem die Schätzung des kürzesten Pfades $s.\mathrm{d}$ und den Vorgängerknoten $s.\mathrm{p}$ für jeden Knoten s im Graphen nach jeder Iteration.

   Bitte beachten Sie: Die Tabelle muss **vollständig** ausgefüllt werden. Insbesondere werden unvollständige Zeilen als Fehler gewertet. Benutzen Sie das Symbol " = ", wenn Sie den Inhalt der Zelle oberhalb der aktuellen Zelle in die aktuelle Zelle kopieren möchten. (Natürlich dürfen Sie den Inhalt auch nochmal abschreiben.) (__ / 6 P)

2. Verwenden Sie nun das obige Ergebnis, um den kürzesten Pfad von d nach f zu ermitteln. Zeichnen Sie dazu die entsprechenden Kanten in die Abbildung unten ein. (__ / 1 P)



								∞	$a.\mathtt{d}$
								∞	$b.\mathtt{d}$
								∞	$c.\mathtt{d}$
								0	$d.\mathtt{d}$
								∞	$e.\mathtt{d}$
								∞	$f.\mathtt{d}$
								∞	$g.\mathtt{d}$
								∞	$h.\mathtt{d}$
								nil	$a.\mathtt{p}$
								nil	$b.\mathtt{p}$
								nil	$c.\mathtt{p}$
								nil	$d.\mathtt{p}$
								nil	$e.\mathtt{p}$
								nil	$f.\mathtt{p}$
								nil	$g.\mathtt{p}$
								nil	$h.\mathtt{p}$
								–	r
								$\{a, b, c, d, e, f, g, h\}$	Q

6 Advanced Designs (__ / 12 P)

6.1 Allgemeine Fragen (__ / 6 P)

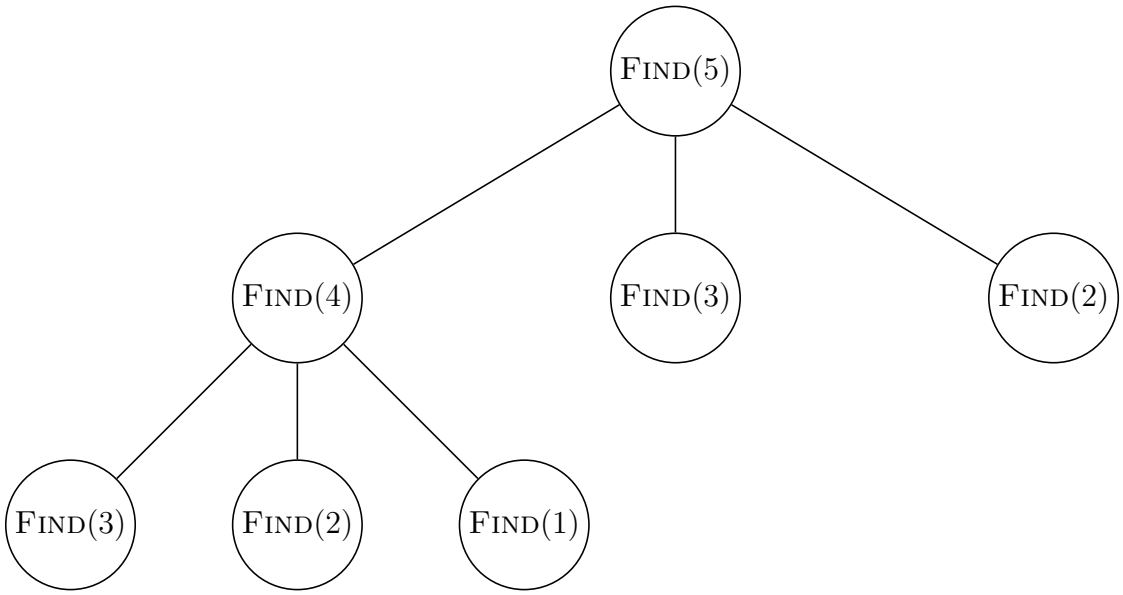
1. Wann ist es sinnvoll Metaheuristiken zur Lösung eines Problems anzuwenden?(__ / 1 P)

2. Warum führt man eine amortisierte Analyse durch? (__ / 1 P)

3. Auf welcher grundlegenden Idee beruht Backtracking? (__ / 1 P)

4. In der dynamischen Programmierung gibt es die Ansätze **Top-down** und **Bottom-up**. Erklären Sie kurz den Unterschied zwischen den Ansätzen. (__ / 2 P)

5. Gegeben ist ein vollständiger Rekursionsbaum für die Ausführung eines rekursiven Algorithmus FIND für $n = 5$. Leiten Sie anhand des Rekursionsbaums die Laufzeit des zugrundeliegenden Algorithmus ab. (__ / 1 P)



6.2 Greedy Algorithmus (__ / 6 P)

1. Auf welcher grundlegenden Idee beruht das Greedy Paradigma? (__ / 1 P)

2. Der Fachbereich Informatik der Technischen Universität Darmstadt bittet Sie bei der Erstellung eines Belegungsplans von Vorlesungen für das Audimax zu helfen. Gegeben sind n Vorlesungen, welche durch ihre jeweilige Start- und Endzeit gegeben sind, also konkret $[s_i, e_i)$ für $1 \leq i \leq n$. Schreiben Sie nun selbstständig einen Greedy-Algorithmus in Pseudocode, welcher die Belegung des Hörsaals mit möglichst vielen Ereignissen mit disjunkten Zeitspannen stattfinden lässt. Nutzen Sie dazu die folgende Vorlage und beachten Sie, dass Sie nicht alle Zeilen der Vorlage nutzen müssen. (__ / 5 P)

GREEDY( )

```
1 :
2 :
3 :
4 :
5 :
6 :
7 :
8 :
9 :
10 :
11 :
12 :
```

7 Asymptotik (__ / 11 P)

7.1 Asymptotische Laufzeiten (__ / 7 P)

1. Geben Sie für die folgenden Abschätzungen der Laufzeit $f(n)$ die asymptotische Laufzeit in O-, Ω-, oder Θ-Notation an. Wählen Sie jeweils die **restriktivste** Notation, und entfernen Sie alle überflüssigen Terme.

   Beispiel: $f(n) = 2n^2+3n+4$ sollte durch $\Theta(n^2)$, jedoch nicht durch $O(n^3)$ oder $\Theta(2n^2+3n+4)$ angegeben werden. (__ / 5 P)

   i) $f(n) = 22n^2 + 5n^4 + 1024n$

   ii) $f(n) \leq 2^{5n} + n^{10} - 3$

   iii) $f(n) \geq 1027 + 2^{2^2 8}$

   iv) $f(n) = g(27g(8g(n) + 1024))$, wobei $g(n) = n^3$

   v) $f(n) \leq 15g(g(n) - 8)$, wobei $g(n) = \log_2(n)$

2. Sei f eine Funktion in $O(n)$. **Argumentieren Sie anhand der Definition** von asymptotischer Komplexität: Es gibt einen Wert $n_0 \in \mathbb{N}$, ab dem die Funktion $g(n) = n^2$ immer größer ist als $f(n)$. (__ / 2 P)

   Hinweis: Die Argumentation, n^2 sei in $\Omega(n^2)$ und $O(n) < \Omega(n^2)$ ist hier **nicht ausreichend**!

7.2 Mastertheorem und NP (__ / 4 P)

1. Begründen Sie für die Rekursionsgleichung

$$T(n) = 16T(n/2) + n^{4.1} \log n,$$

ob Sie das Mastertheorem anwenden können oder nicht. Benutzen Sie gegebenenfalls das Mastertheorem, um eine asymptotische Schranke für $T(n)$ zu bestimmen. (__ / 2,5 P)

2. Welche Bedingungen muss ein Problem erfüllen, damit es in NP enthalten ist? (__ / 1 P)

3. Nennen Sie zwei NP-vollständige Probleme. (__ / 0,5 P)