Technische Universität Darmstadt — Fachbereich Informatik

Abschlussklausur / Fachprüfung

Algorithmen und Datenstrukturen

Sommersemester 2020

01.09.2020 — 09:00–11:00 Uhr

Hinweise:

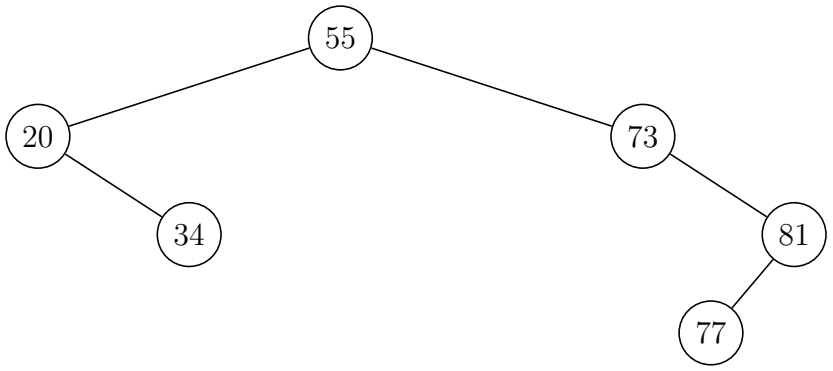
- Als Schreibmittel ist nur ein **schwarzer oder blauer Schreibstift** erlaubt. Verwenden Sie im Besonderen keine Bleistifte, Rot- oder Korrekturstifte, sowie keine Korrekturflüssigkeit oder Tintenlöscher. Streichen Sie deutlich durch, was nicht gewertet werden soll.
- Als Hilfsmittel können Sie unsere 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

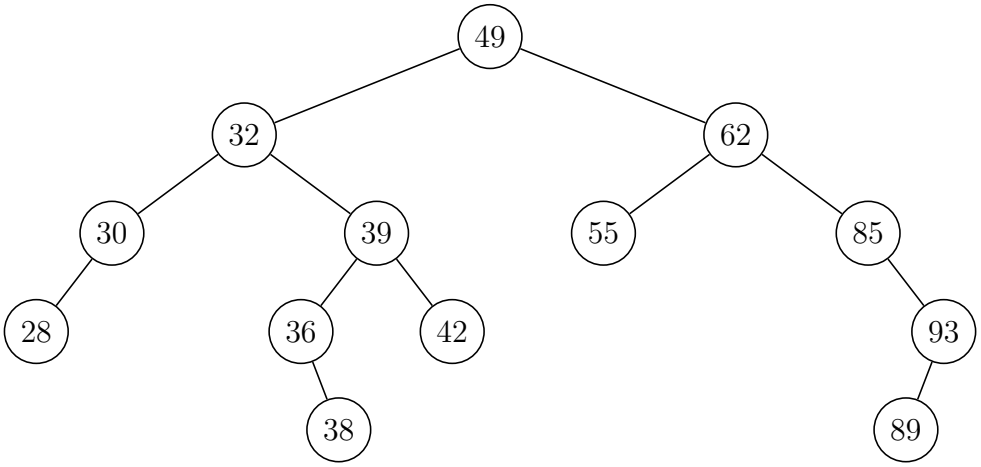
Aufgabe	1	2	3	4	5	6	7	$\sum$
Max. Punkte	10	9	23	18	15	15	10	100
Punkte								
Kürzel								

1 Binäre Suchbäume (__ / 10 P)

1. Fügen Sie **nacheinander** die folgenden Werte in den gegebenen binären Suchbaum ein: 80, 15, 70, 22, 56, 54, 101, 72. 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 PREORDER(T.root), POSTORDER(T.root) und INORDER(T.root), und geben Sie diese unten an. (__ / 3 P)



PREORDER(T.root):

POSTORDER(T.root):

INORDER(T.root):

3. Löschen Sie den Knoten 32 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. Geben Sie ein **Gegenbeispiel** für die folgende Aussage an: Sei B ein Suchpfad in einem binären Suchbaum, der bei einem Blatt endet, und seien A die Knoten links von dem Suchpfad sowie C die Knoten rechts von dem Suchpfad. Dann gilt für alle $a \in A, b \in B$ und $c \in C$: $a \leq b \leq c$.

   Zeichnen Sie den Suchpfad in das Gegenbeispiel durch Umkreisen ein und begründen Sie, warum dieser Baum die Bedingung verletzt. (__ / 3 P)

2 Bloomfilter (__ / 9 P)

1. 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)

2. Wir betrachten jetzt einen Bloomfilter, der natürliche Zahlen enthält. Gegeben seien dafür die beiden Hashfunktionen H_0 und H_1 und ein Bloomfilter BF, der diese beiden Hashfuntionen nutzt, in folgendem initialen Zustand:

$$H_0(n) = 3q(n) + 1 \mod 7 \qquad H_1(n) = q(n)^2 + 1 \mod 7$$

$$BF = \begin{array}{|c|c|c|c|c|c|c|} \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

Dabei berechnet $q(n)$ die Quersumme der Zahl, also die Summe aller Ziffern (beispielsweise wäre die Quersumme von 76 gleich 13).

Fügen Sie nacheinander die folgenden Elemente in den Bloomfilter ein und geben Sie jeweils den resultierenden Zustand des Bloomfilters an: (__ / 4,5 P)

a) 15

b) 202

c) 2012

3. Ergänzen Sie die folgende Funktion, die die **Vereinigung** von zwei Bloomfiltern BF_1 und BF_2 berechnet: Ein Element soll also im neuen Bloomfilter enthalten sein, wenn es in BF_1 oder BF_2 enthalten war. BF_1 und BF_2 sind dabei von der gleichen Länge n und benutzen die gleichen Hashfunktionen H_0 und H_1. (__ / 3,5 P)

BloomUnion(BF_1, BF_2)

```
1 :  BF = new Array(n)
2 :  for i = 0 to n − 1 do
3 :      BF[i] = 0
4 :
5 :
6 :
7 :
8 :
9 :
10 :
```

3 Bäume (__ / 23 P)

In den Teilen zu Rot-Schwarz-Bäumen innerhalb dieser Aufgabe wird folgende Notation für rot und schwarz gefärbte Knoten verwendet: Rote Knoten werden durch ein Rechteck gekennzeichnet, und schwarze Knoten durch einen Kreis markiert.

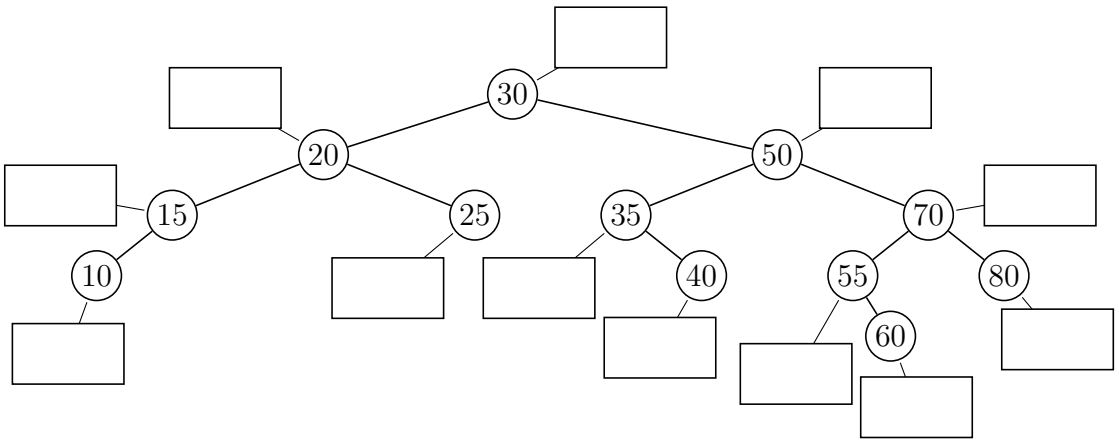
Roter Knoten

Schwarzer Knoten

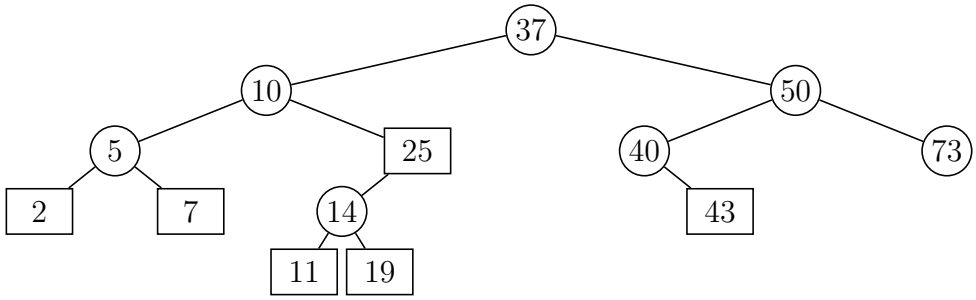
Bei den Aufgaben zu anderen Arten von Bäumen wird die übliche Notation aus der Vorlesung verwendet. Definieren Sie diese Symbole **nicht** um.

3.1 Bäume erkennen (__ / 8 P)

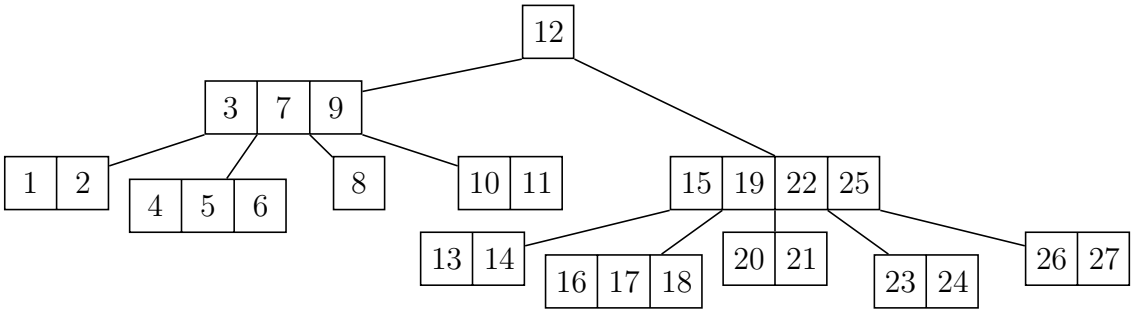
1. Ist der folgende Baum ein AVL-Baum? Bestimmen Sie zunächst die Balance in jedem Knoten des Baumes, und schreiben Sie diese in das jeweils vorgegebene Feld.
   Geben Sie anschließend an, ob es sich bei dem Baum um einen AVL-Baum handelt. **Begründen** Sie Ihre Antwort. Geben Sie bei einer positiven Antwort an, welche Eigenschaften gelten, und nennen Sie bei einer negativen Antwort mindestens eine Eigenschaft, die verletzt wird, und wo. (__ / 3 P)



2. Ist der folgende Baum ein Rot-Schwarz-Baum? **Begründen** Sie Ihre Antwort. Geben Sie bei einer positiven Antwort an, welche Eigenschaften gelten, und nennen Sie bei einer negativen Antwort mindestens eine Eigenschaft, die verletzt wird, und wo. (__ / 2 P)

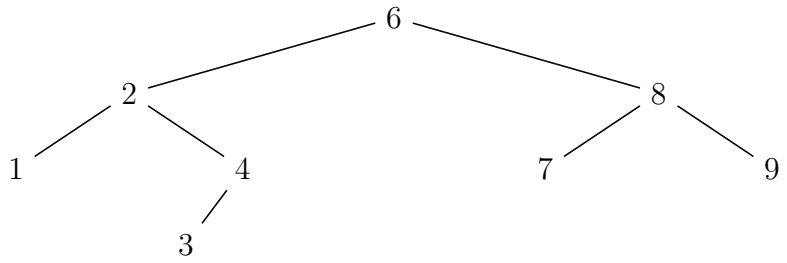


3. Existiert ein $t \in \mathbb{N}_{\geq 2}$, sodass der folgende Baum ein B-Baum vom Grad t ist? **Begründen** Sie Ihre Antwort. Geben Sie bei einer positiven Antwort an, welche Eigenschaften gelten und bestimmen Sie einen möglichen Kandidaten für t, und nennen Sie bei einer negativen Antwort mindestens eine Eigenschaft, die verletzt wird, und wo. (__ / 3 P)



3.2 Verständnisfragen (__ / 7 P)

1. Gegeben sei der folgende **binäre Suchbaum**. Bestimmen Sie jeweils **eine** mögliche Rot-Schwarz-Färbungen für diesen Baum mit Schwarzhöhe 2 und 3, und geben Sie diese im entsprechenden Abschnitt an. (__ / 2 P)



a) Färbung mit Schwarzhöhe 2:

b) Färbung mit Schwarzhöhe 3:

2. In dieser Aufgabe soll die Anzahl $RB(1)$ der **verschiedenen** Rot-Schwarz-Bäume mit Schwarzhöhe 1 bestimmt werden. Dabei bedeutet "verschieden", dass lediglich die Farbe der Knoten und deren Position im Baum von Bedeutung sind. Die konkreten Schlüsselwerte sollen dabei ignoriert werden.

   Bestimmen Sie den Wert $RB(1)$. **Begründen** Sie Ihre Angabe. (__ / 2 P)

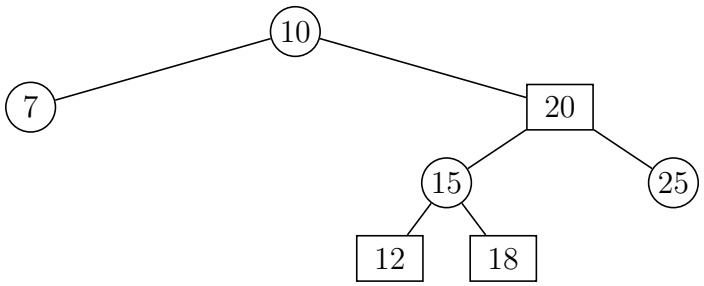
   $$RB(1) = \_\_\_\_\_$$

3. Zeichnen Sie einen AVL-Baum, der (genau) die Schlüssel in $\{n \in \mathbb{N} : 1 \leq n \leq 12\}$ verwaltet, und in dem jeder Knoten, der kein Blatt ist, rechts- oder linkslastig ist. Dabei soll für jeden Knoten v, der kein Blatt ist, folgende Eigenschaft gelten:

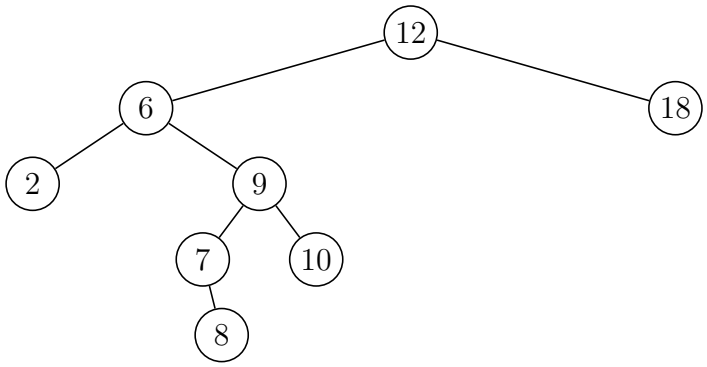
   - Wenn die Tiefe von v gerade ist, dann ist $B(v) = +1$;
   - Wenn die Tiefe von v ungerade ist, dann ist $B(v) = -1$. (__ / 3 P)

3.3 Einfügen und Löschen (__ / 8 P)

1. Fügen Sie einen Knoten mit dem Schlüssel 17 in den folgenden Rot-Schwarz-Baum ein. Verwenden Sie dabei den Algorithmus aus der Vorlesung zum Einfügen von Knoten in einen Rot-Schwarz-Baum.
   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. (__ / 4 P)



2. Löschen Sie den Knoten mit dem Schlüssel 9 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. (__ / 4 P)

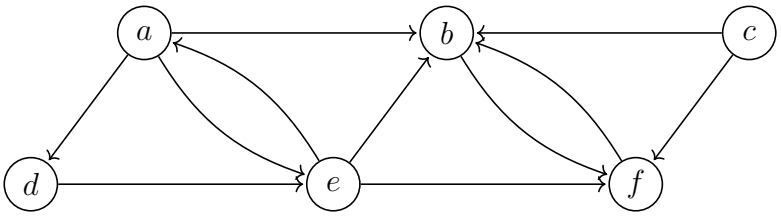


4 Graphenalgorithmen (__ / 18 P)

4.1 Verständnisfragen (__ / 11 P)

1. Nennen und **definieren** Sie das Problem, welches vom Algorithmus von Bellman-Ford gelöst wird. Welche asymptotische Laufzeit hat der Algorithmus? (__ / 2 P)

2. Gegeben sei der folgende Graph $G = (V, E)$. Bestimmen Sie die starken Zusammnhangskomponenten von G, indem Sie diese in der Abbildung unten einkreisen. (__ / 1 P)



3. a) Seien $n \in \mathbb{N}_{\geq 1}$ und $G = (V, E)$ ein Graph mit $|V| = n$ Knoten. Bestimmen Sie die maximale Anzahl von Kanten die G haben darf, sodass G topologisch sortiert werden kann. (__ / 2 P)

b) Beweisen Sie Ihre Schranke aus a). (__ / 3 P)

4. Beweisen oder widerlegen (mithilfe eines Gegenbeispiels) Sie die folgende Behauptung:

Es seien $G = (V, E, w)$ ein zusammenhängender, ungerichteter, gewichteter Graph mit $w(e) \geq 0$ für alle $e \in E$, und $s \in V$ ein ausgezeichneter Knoten. Man führe den Algorithmus von Dijkstra auf dem Graphen G mit Startknoten s aus, und es sei $G_p = (V, E_p)$ mit $E_p = \{\{v, v.p\} : v \in V, v \neq s\}$ der daraus entstehende (ungerichtete) Vorgängergraph. Dann ist G_p ein minimaler Spannbaum von G.

(__ / 3 P)

4.2 Algorithmus von Prim (__ / 7 P)

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

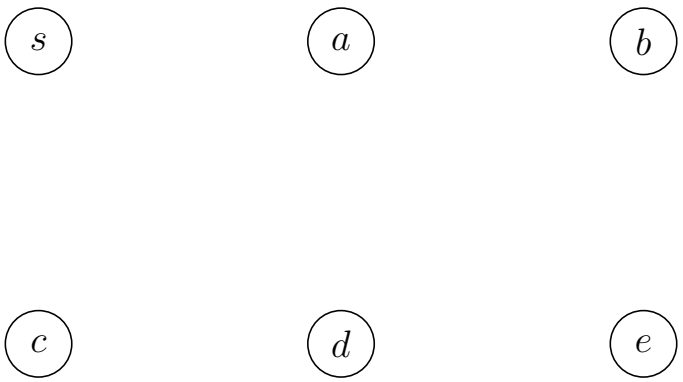
$$\begin{array}{c} \\ s \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccccc} s & a & b & c & d & e \end{array} \\ \begin{pmatrix} \square & 5 & -2 & 73 & -7 & 13 \\ 5 & \square & -1 & 2 & 3 & 7 \\ -2 & -1 & \square & 14 & 2 & \square \\ 73 & 2 & 14 & \square & 37 & 8 \\ -7 & 3 & 2 & 37 & \square & 10 \\ 13 & 7 & \square & 8 & 10 & \square \end{pmatrix} \end{array}$$

Hier bedeutet eine Zahl k in Zeile i und Spalte j, dass $(i, j) \in E$ und $w((i, j)) = k$. Ein Kästchen " $\square$ " an der entsprechenden Stelle bedeutet, dass $(i, j) \notin E$. Da der Graph G ungerichtet ist, ist die obige Matrix symmetrisch.

1. Führen Sie den Algorithmus von Prim auf diesem Graphen aus. Beginnen Sie im Knoten s. 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 u im gegebenen Schritt aus der Menge Q extrahiert wird, die Werte der Key- und Vorgänger-Parameter $v.k$ und $v.p$ für jeden Knoten v des Graphen, sowie die Menge Q, am Ende des betrachteten Schrittes.

   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 einen minimalen Spannbaum von G anzugeben. Zeichnen Sie dazu die entsprechenden Kanten in die Vorlage unten ein. (__ / 1 P)



						$-\infty$	$s.k$
						∞	$a.k$
						∞	$b.k$
						∞	$c.k$
						∞	$d.k$
						∞	$e.k$
						nil	$s.p$
						nil	$a.p$
						nil	$b.p$
						nil	$c.p$
						nil	$d.p$
						nil	$e.p$
						–	u
						$\{s, a, b, c, d, e\}$	Q

5 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 **abfallender** 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. Gegeben sei ein Array A der Länge n. Analysieren Sie die Laufzeit von SELECTION-SORT für Array A , wenn Sie diesen in Form von MIN-SORT (wie in der Vorlesung besprochen) ausführen, wie folgt: Geben Sie zuerst die Anzahl der notwendigen Vergleiche an und schließen Sie dann auf die worst-case Laufzeit. (__ / 2 P)

   Geben Sie im Vergleich noch die best-case Laufzeit des Algorithmus an und begründen Sie Ihre Antwort. (__ / 1 P)

3. Führen Sie die Unterroutine MERGE in MERGE-SORT auf dem Array

$$A[19..24] = [3, 6, 11, 5, 7, 9]$$

aus.

a) Bestimmen Sie im ersten Schritt die Arrays L und R. (__ / 1 P)

$$L = [\qquad\qquad\qquad\qquad], \quad R = [\qquad\qquad\qquad\qquad]$$

b) Anschließend durchläuft der Algorithmus eine **for**-Schleife und überschreibt nach und nach die Elemente in A. Geben Sie den Zustand des Arrays A nach jedem Durchlauf der **for**-Schleife sowie direkt vor dem Ende an. Verwenden Sie den Algorithmus, der in der Vorlesung gezeigt wurde. Geben Sie die Werte von i, j und k zu Beginn des Schleifenkopfs der **for**-Schleife sowie direkt vor dem Ende von MERGE konkret an. (__ / 6 P)

$[3, 6, 11, 5, 7, 9]$

$i =$ _____, $j =$ _____, $k =$ _____

[, , , , ,]

$i =$ _____, $j =$ _____, $k =$ _____

[, , , , ,]

$i =$ _____, $j =$ _____, $k =$ _____

[, , , , ,]

$i =$ _____, $j =$ _____, $k =$ _____

[, , , , ,]

$i =$ _____, $j =$ _____, $k =$ _____

[, , , , ,]

$i =$ _____, $j =$ _____, $k =$ _____

[, , , , ,]

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

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

_____

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

_____

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

_____

6 Advanced Designs (__ / 15 P)

6.1 Allgemeine Fragen (__ / 4 P)

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

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

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

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

6.2 Dynamic Programming (__ / 11 P)

Sie finden nach einer langen Nacht in der Universitäts- und Landesbibliothek Darmstadt auf einem verlassenen Tisch ein Blatt Papier mit der Überschrift *Münzenproblem*. Auf dem Blatt Papier finden Sie die folgende Problembeschreibung sowie einen unvollständigen Algorithmus in Pseudocode zum Lösen des Problems.

Problembeschreibung

Gegeben ist eine Währung, welche aus drei Münzen mit den Werten 1, 2 und 3 besteht. Eine Person möchte jetzt gerne das Rückgeld mit dem Wert n zurückgeben und ist daran interessiert wie viele verschiedene Möglichkeiten es gibt den Betrag aus den gegebenen Münzen zu formen um genau den Betrag n zu geben. Beachten Sie dabei, dass eine Ausgabe von $1, 2$ unterschiedlich ist zu $2, 1$.

Beispielsweise kann der Betrag von $n = 3$ in vier verschiedenen Weisen erstattet werden.

1. Geben Sie die vier Möglichkeiten an um den Betrag von $n = 3$ zu erstatten. (__ / 1 P)

2. Implementieren Sie einen rekursiven Algorithmus, der die Anzahl aller möglichen Ausgaben berechnet um den Betrag des Wertes n auszugeben. Vervollständigen Sie die fehlenden Lücken im Pseudocode. (__ / 3 P)

$\textsc{Coins}(n)$

```
1 : if n = 1 then
2 :     return _____
3 : elseif n = 2 then
4 :     return _____
5 : elseif n = 3 then
6 :     return _____
7 : else
8 :     s = _____
9 : return s
```

3. Geben Sie den vollständigen Rekursionsbaum für die Ausführung von $\textsc{Coins}(5)$ an und nutzen Sie diesen um zu berechnen wie viele Möglichkeiten es gibt um $n = 5$ zurückzugeben. (__ / 2 P)

4. Welche Laufzeit hat der Algorithmus $\textsc{Coins}$? (__ / 1 P)

5. Entwerfen Sie unter Beachtung des **Bottom-up** Ansatzes einen Algorithmus zur Lösung des Münzproblems. Verwenden Sie dazu die gegebene Vorlage und beachten Sie, dass Sie nicht alle Zeilen der Vorlage nutzen müssen. (__ / 4 P)

BOTTOM-UP-COINS( )

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

7 Asymptotik (__ / 10 P)

7.1 Asymptotische Laufzeiten (__ / 5 P)

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.

i) $f(n) = 3n^2 + 5 \cdot 2^n + 1024n^{125}$

ii) $f(n) \geq 5n^3 + 16n^2 - 3n + 1024$

iii) $f(n) \leq 156 \cdot 5^{1024}$

iv) $f(n) = \begin{cases} 2^n, & n < 10000 \\ n^2 + 15n + 2048, & n \geq 10000 \end{cases}$

v) $f(n) \leq g(g(n) + 5)$, wobei $g(n) = \log_2(n)$,

7.2 Mastertheorem (__ / 5 P)

Begründen Sie für jede der folgenden Rekursionsgleichungen $T(n)$, ob Sie das Mastertheorem anwenden können oder nicht. Benutzen Sie gegebenenfalls das Mastertheorem, um eine asymptotische Schranke für $T(n)$ zu bestimmen.

1. $T(n) = 25T(n/5) + n^4 \log_3(n) + 15$

2. $T(n) = 8T(n/2) + n^3 \log_4(n + 16)$