Technische Universität Darmstadt — Fachbereich Informatik

Abschlussklausur / Fachprüfung

Algorithmen und Datenstrukturen

Sommersemester 2019

03.09.2019 — 09:00–11:00 Uhr

Hinweise:

- Als Schreibmittel ist nur ein **schwarzer oder blauer Schreibstift** erlaubt.
- Als Hilfsmittel können Sie unsere Folien, Übungen, sonstige Notizen, Bücher, Wörterbücher und sonstige 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.
- Zum Bestehen der Klausur sind 50 von insgesamt 100 Punkten hinreichend.

Nachname	
Vorname	
Matrikelnr.	
Studiengang	

Aufgabe	1	2	3	4	5	6	7	8	$\sum$
Max. Punkte	11	15	9	12	9	11	19	14	100
Punkte									
Kürzel									

1 Sortieren (__ / 11 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. (__ / 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. Inwiefern unterscheiden sich die beiden Sortieralgorithmen INSERTION-SORT und MERGE-SORT **konzeptionell**? (__ / 1 P)

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

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

$$A[54 \ldots 59] = [14, 17, 8, 1, 4, 12]$$

aus. Der Algorithmus durchläuft in einer **for**-Schleife nacheinander die Elemente des Arrays. 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. Als Pivotelement wird das letzte Element des Teilarrays gewählt. Geben Sie den Wert von i und j nach jeder Iteration der **for**-Schleife sowie direkt vor dem Ende von PARTITION konkret an. (__ / 5,5 P)

$$[14, 17, 8, 1, 4, 12] \qquad i = 53$$

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

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

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

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

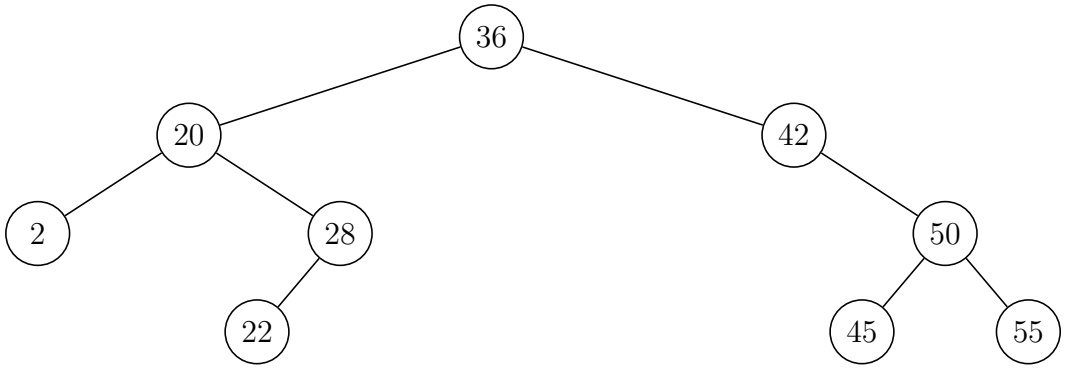
$[\ \ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ ]$ $i =$ ______ , $j =$ ______

$[\ \ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ ]$

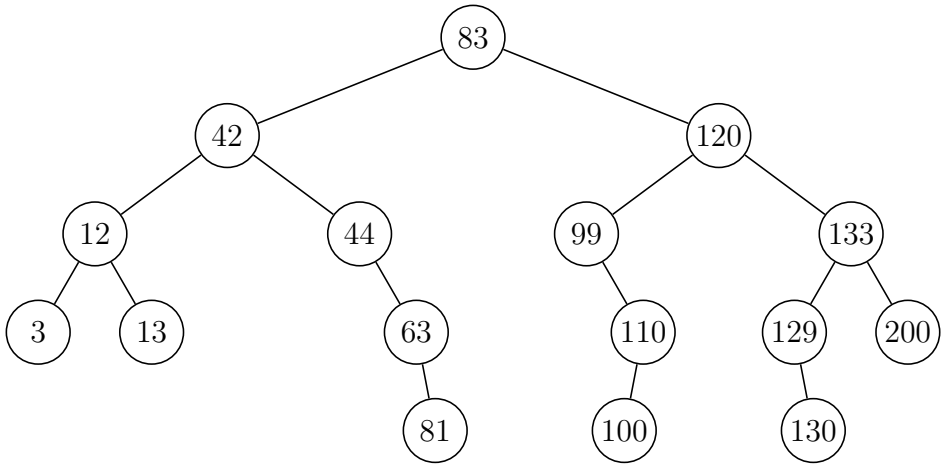
5. Auf welchen beiden Unterarrays wird QUICKSORT als nächstes aufgerufen? (__ / 1 P)

2 Binäre Suchbäume (__ / 15 P)

1. Fügen Sie **nacheinander** die folgenden Werte in den gegebenen binären Suchbaum ein: $10, 8, 39, 49, 23, 41, 34, 37$. Verwenden Sie dabei den Algorithmus aus der Vorlesung zum Einfügen von Knoten in einen binären Suchbaum. Sie dürfen den angegebenen binären Suchbaum vervollständigen. (__ / 4 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. (__ / 6 P)



PREORDER(T.root):

POSTORDER(T.root):

INORDER(T.root):

3. Löschen Sie den Knoten 120 aus dem Baum aus Teilaufgabe 2. Verwenden Sie dabei den Algorithmus aus der Vorlesung zum Löschen von Knoten aus einem binären Suchbaum. Geben Sie den Baum an, der daraus entsteht. (__ / 2 P)

4. Für eine der folgenden drei Zahlenfolgen existiert kein binärer Suchbaum, für den die Folge ein gültiger Suchpfad ist. Nennen Sie diese Folge und begründen Sie, warum sie keinen gültigen Suchpfad darstellt. (__ / 3 P)

   Folge 1: $232, 809, 760, 320, 512, 603, 759, 758$

   Folge 2: $111, 723, 412, 698, 564, 398, 487, 460$

   Folge 3: $1, 2, 3, 300, 128, 76, 54, 60$

3 Verkettete Listen, Stacks und Queues (__ / 9 P)

1. Im folgenden sind einige Szenarien genannt, die jeweils eine Datenstruktur zum Speichern benötigen. Entscheiden Sie, in welchen der folgenden Situationen Stacks und in welchen Queues verwendet werden sollten. **Begründen** Sie Ihre Auswahl.

   a) Ein Mailserver speichert einkommende Mails zwischen, um sie dann abarbeiten zu können. (__ / 1 P)

   b) Ein Texteditor speichert Änderungen, um eine "Rückgängig"-Funktion zu implementieren. (__ / 1 P)

   c) Ein Backtracking-Algorithmus speichert in jedem Schritt, welche möglichen nächsten Schritte schon ausprobiert wurden. (__ / 1 P)

   d) Eine Breitensuche speichert die Kinder des aktuellen Knotens als Knoten, die noch abgearbeitet werden müssen. (__ / 1 P)

2. Eine Queue Q ist auf einem **virtuell zyklischen** Array der Größe 5 implementiert. Es wurden bereits die Werte $5, 8, 9$ und 7 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.
(__ / 3 P)

	5	8	9	7	$-$
`dequeue`(Q):					
`enqueue`$(Q, 3)$:					
`enqueue`$(Q, 0)$:					
`dequeue`(Q):					
`dequeue`(Q):					
`enqueue`$(Q, 7)$:					

3. Nennen Sie einen Vorteil und einen Nachteil, Queues mit verketteten Listen statt mit Arrays zu implementieren. (__ / 2 P)

4 Rot-Schwarz-Bäume (__ / 12 P)

In 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.

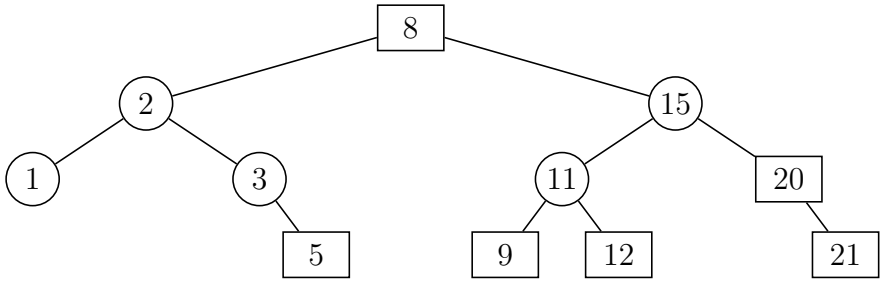
Roter Knoten

Schwarzer Knoten

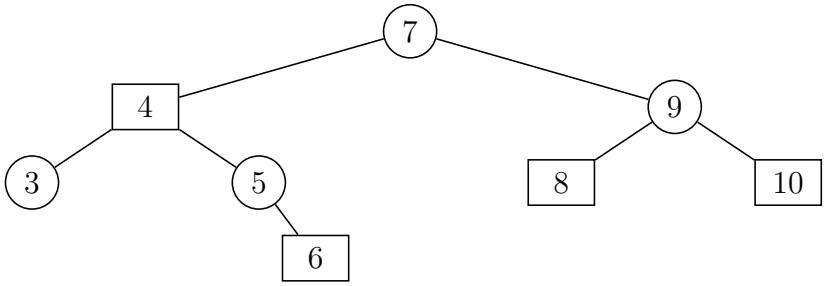
4.1 Verständnisfragen (__ / 8 P)

1. Welche der folgenden Bäume sind Rot-Schwarz-Bäume? **Begründen** Sie Ihre Angabe.

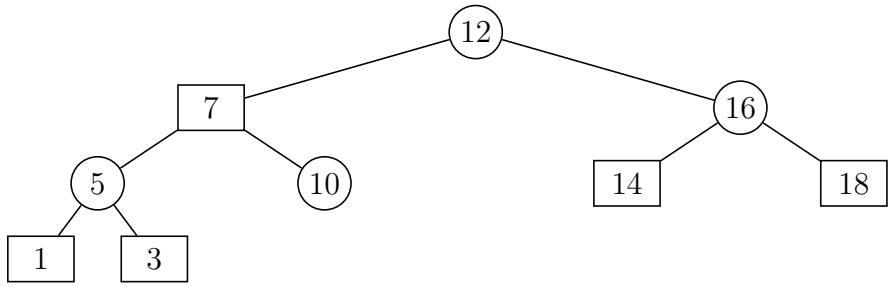
a) (__ / 1 P)



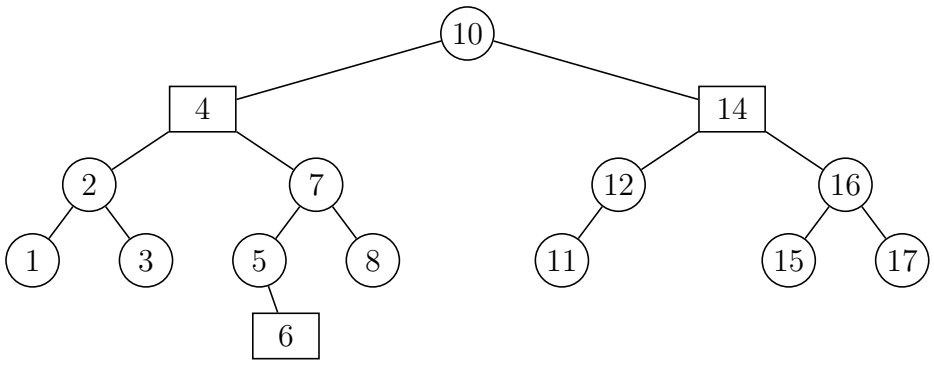
b) (__ / 1 P)



c) (__ / 1 P)

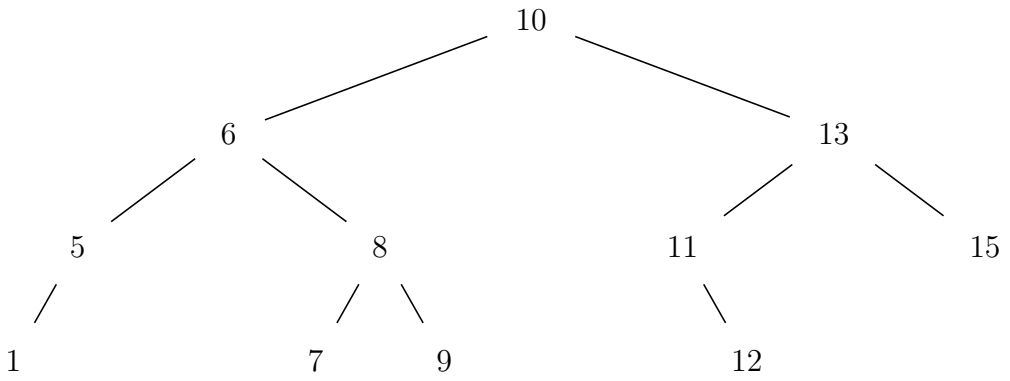


d) (__ / 1 P)



2. Nennen Sie einen Vorteil von Rot-Schwarz-Bäumen gegenüber herkömmlichen binären Suchbäumen. Wie wird dieser erreicht? (__ / 2 P)

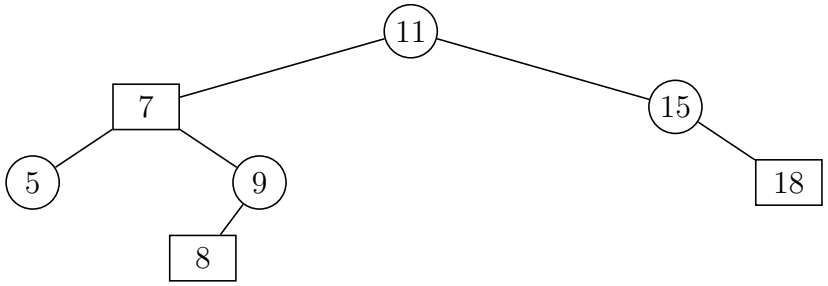
3. Gegeben sei folgender **binärer Suchbaum**. Bestimmen Sie eine mögliche Rot-Schwarz-Färbungen für diesen Baum. Vervollständigen Sie jeden Knoten entsprechend durch ein Rechteck oder einen Kreis. (__ / 2 P)



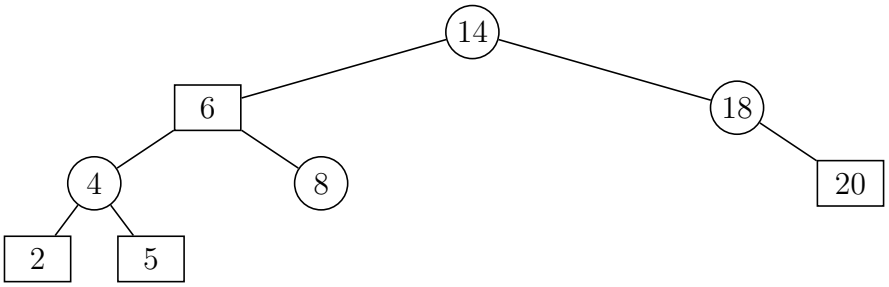
4.2 Einfügen (__ / 4 P)

In dieser Aufgabe sollen Sie bestimmte Knoten in vorgegebene Rot-Schwarz-Bäume einfügen. Verwenden Sie dabei stets den Algorithmus aus der Vorlesung zum Einfügen von Knoten in einen Rot-Schwarz-Baum. Zeichnen Sie Ihr Zwischenergebnis **jeweils** nach dem Einfügen, sowie nach jeder Iteration der **while**-Schleife in FixColorsAfterInsertion und gegebenenfalls nach dem letzten Umfärben der Wurzel.

1. Fügen Sie den Knoten 10 in den folgenden Rot-Schwarz-Baum ein: (__ / 1 P)



2. Fügen Sie den Knoten 3 in den folgenden Rot-Schwarz-Baum ein: (__ / 3 P)

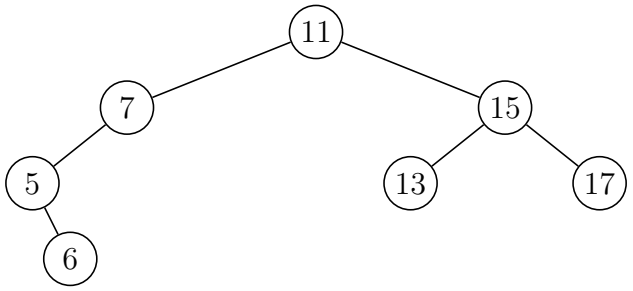


5 Splay-Bäume (__ / 9 P)

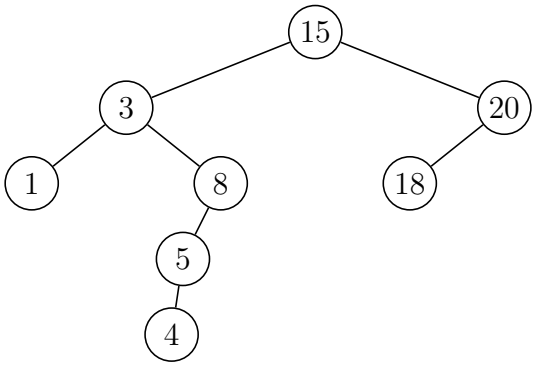
5.1 Einfügen (__ / 5 P)

In dieser Aufgabe sollen Sie bestimmte Knoten in vorgegebene Splay-Bäume einfügen. Verwenden Sie dabei stets den Algorithmus aus der Vorlesung zum Einfügen von Knoten in einen Splay-Baum. Zeichnen Sie Ihr Zwischenergebnis **jeweils** nach jeder erfolgten Zig-Zig-, Zig-Zag-, und Zig-Operation, sowie nach dem Einfügen des Knotens.

1. Fügen Sie den Knoten 9 in den folgenden Splay-Baum ein: (__ / 2 P)

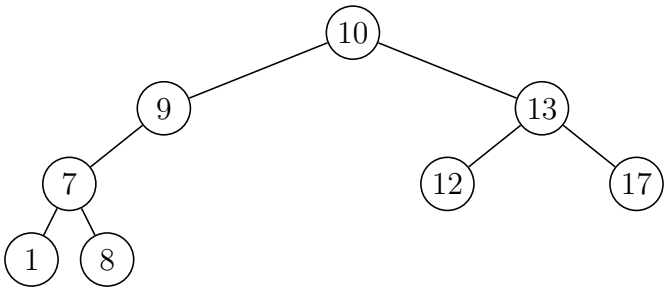


2. Fügen Sie den Knoten 6 in den folgenden Splay-Baum ein: (__ / 3 P)



5.2 Löschen (__ / 4 P)

Löschen Sie den Knoten 9 aus dem folgenden Splay-Baum. Verwenden Sie dabei stets den Algorithmus aus der Vorlesung zum Löschen von Knoten aus einem Splay-Baum. Zeichnen Sie Ihr Zwischenergebnis **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.



6 Advanced Designs (__ / 11 P)

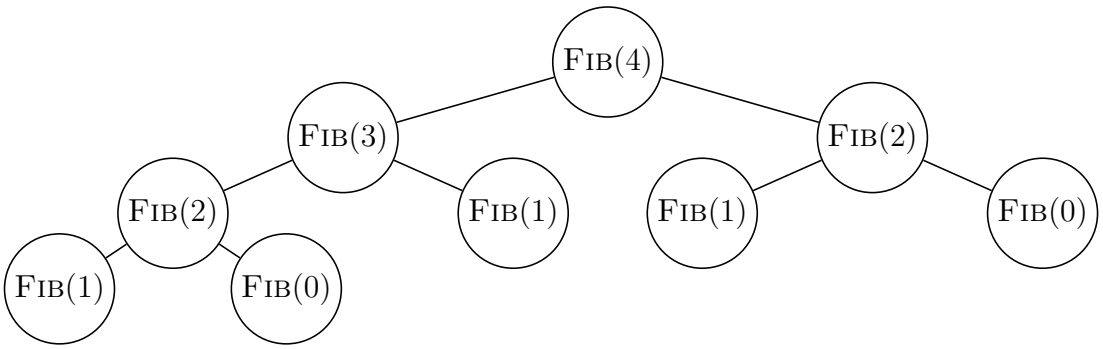
6.1 Dynamic Programming (__ / 4 P)

Im Folgenden ist der naive rekursive Algorithmus zum Berechnen der n-ten Fibonacci Zahl dargestellt.

```
FIB(n)
1 : if n ≤ 2 then
2 :     f = 1
3 : else
4 :     f = FIB(n − 1) + FIB(n − 2)
5 : return f
```

1. Sie haben in der Vorlesung die Top-Down mit Memoization Variante kennengelernt, welche die Laufzeit des Algorithmus verbessert hat. Benennen und erklären Sie die Eigenschaft, welche dafür ausgenutzt wird. (__ / 2 P)

2. Betrachten Sie im Folgenden den Rekursionsbaum zur Berechnung der vierten Fibonacci Zahl. Geben Sie an, in welchen Rekursionsaufrufen das Ergebnis der vierten Fibonacci Zahl tatsächlich berechnet wurde, wenn der Algorithmus in der Version Top-Down mit Memoization ausgeführt wurde. Umkreisen Sie hierzu den entsprechenden Teilbaum. (__ / 2 P)



6.2 Greedy-Algorithmus (__ / 7 P)

1. Auf welcher grundlegenden Idee beruht der Greedy-Algorithmus? (__ / 1 P)

2. Sie wurden von der Wissenschaftsstadt Darmstadt beauftragt für ein Parkhaus einen Parkautomaten zu programmieren, welcher nach erfolgreicher Zahlung der Parkgebühr etwaiges Rückgeld ausgibt. Die Menge an Münzen, die der Automat zurückgibt, ist gegeben durch die Menge $M = \{1, 2, 5, 10, 20, 50\}$ (Münzen haben die Einheit Eurocent). Schreiben Sie selbstständig einen Greedy-Algorithmus in Pseudocode, welcher das obige Rückgeld entsprechend zurückgibt. 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 :
```

3. Führen Sie den Greedy-Algorithmus jeweils für das Wechselgeld 36 und 74 aus. (__ / 1 P)

7 Graphenalgorithmen (__ / 19 P)

7.1 Verständnisfragen (__ / 4 P)

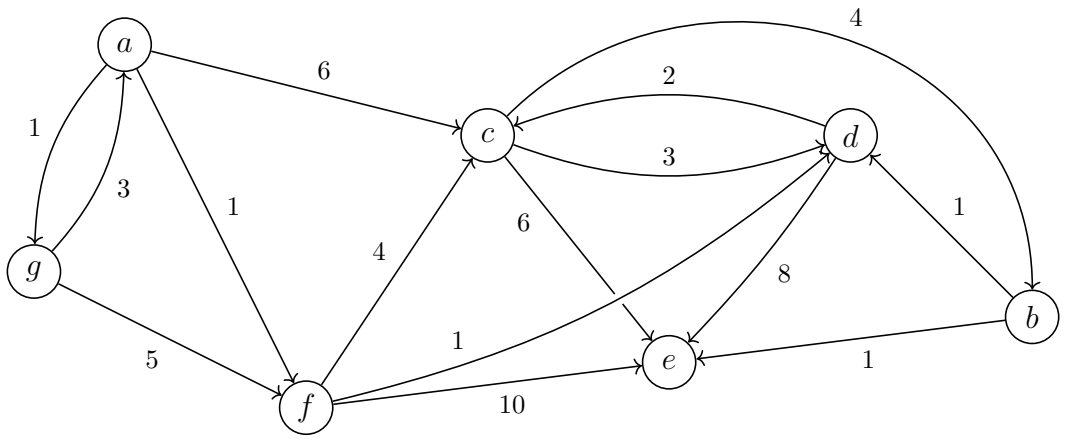
1. **Nennen** und **definieren** Sie das Problem, welches von Dijkstras Algorithmus gelöst wird. (__ / 2 P)

2. Gibt der Algorithmus von Dijkstra immer ein *falsches* Ergebnis aus, wenn er auf einem gewichteten Graphen ausgeführt wird, dessen Gewichtsfunktion auch negative ganzzahlige Werte annimmt? **Begründen** Sie (durch einen Beweis oder Angabe eines Gegenbeispiels) Ihre Antwort. (__ / 1 P)

3. Sei $G = (V, E, w)$ ein gerichteter, gewichteter Graph mit positiver, injektiver Gewichtsfunktion w. Sind die kürzesten Wege in G immer eindeutig? **Begründen** Sie (durch einen Beweis oder Angabe eines Gegenbeispiels) Ihre Antwort. (__ / 1 P)

7.2 Algorithmus von Dijkstra (__ / 9 P)

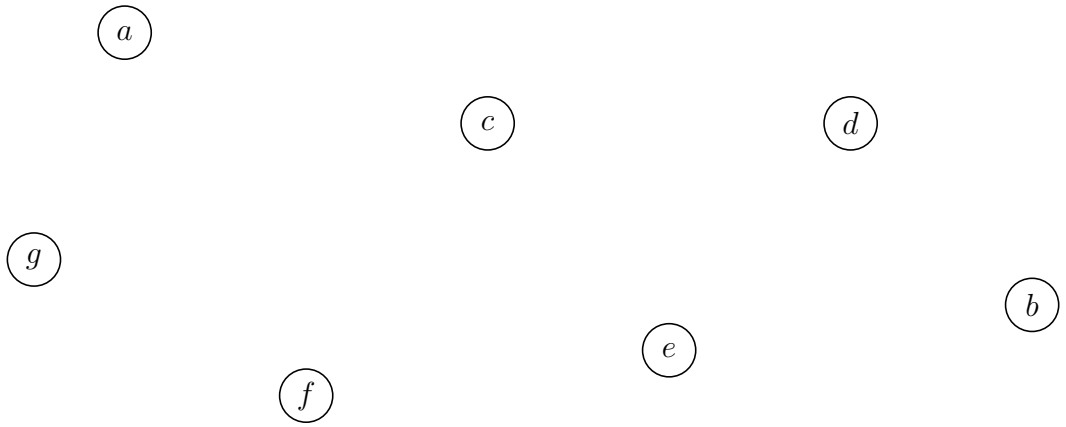
1. Führen Sie den Algorithmus von Dijkstra auf folgendem Graphen aus. Beginnen Sie im Knoten g.



Füllen Sie dazu die Tabellen auf der **nächsten Seite** aus. Diese enthalten jeweils eine separate Zeile für jede Iteration der 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 $s.\mathtt{d}$ und den Vorgängerknoten $s.\mathtt{p}$ für jeden Knoten s des Graphen nach jeder Iteration.

Bitte beachten Sie: Die Tabellen müssen **vollständig** ausgefüllt werden. Insbesondere werden leere Zellen 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). (__ / 8 P)

2. Geben Sie den kürzesten Pfad von g nach e an. Zeichnen Sie dazu die entsprechenden Kanten im Bild unten ein. (__ / 1 P)

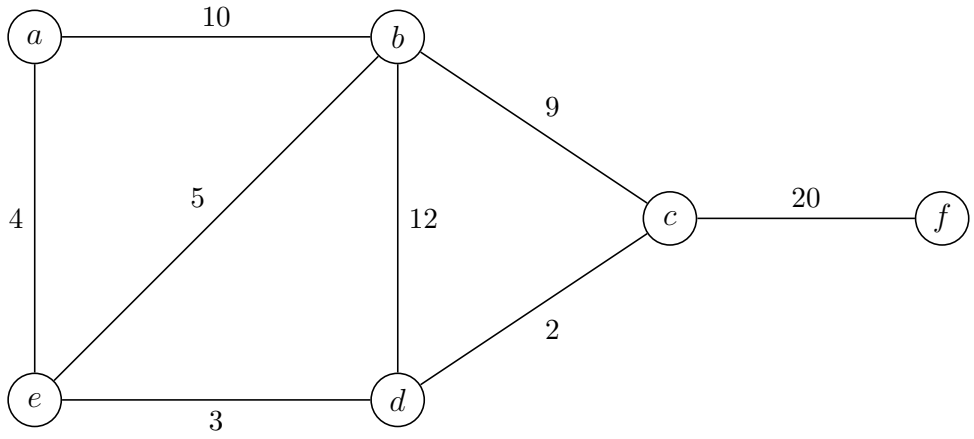


a.d	b.d	c.d	d.d	e.d	f.d	g.d	r
∞	∞	∞	∞	∞	∞	0	$-$

a.p	b.p	c.p	d.p	e.p	f.p	g.p	Q
nil	nil	nil	nil	nil	nil	nil	$\{a, b, c, d, e, f, g\}$

7.3 Algorithmus von Kruskal (__ / 6 P)

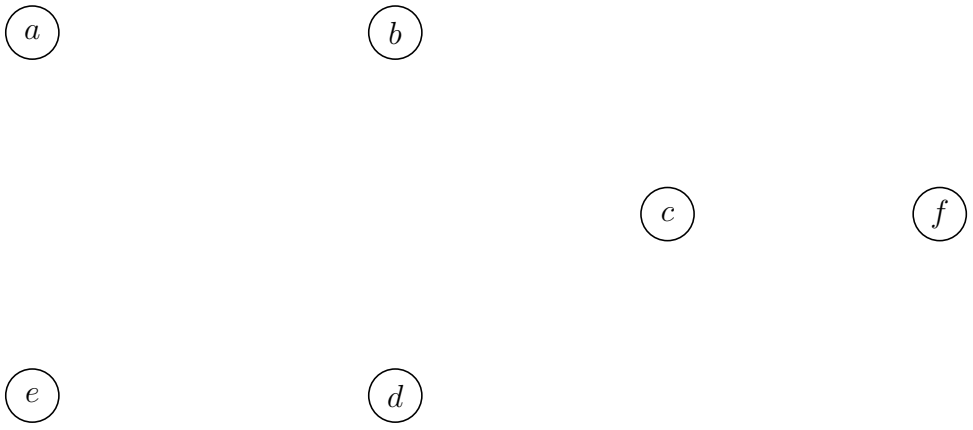
1. Führen Sie den Algorithmus von Kruskal auf folgendem Graphen aus.



Füllen Sie dazu die Tabelle auf der **nächsten Seite** aus. Diese enthält eine separate Zeile für jede Iteration der Schleife im Algorithmus. Sie sollen in jeder Zeile angeben, welche Kante $\{u, v\}$ im gegebenen Schritt betrachtet wird, das Gewicht $w(\{u, v\})$ von $\{u, v\}$, ob die Kante dem minimalen Spannbaum hinzugefügt wird oder nicht (j/n), sowie die Menge $\mathrm{set}(s)$ für jeden Knoten s des Graphen am Ende des betrachteten Schrittes.

Bitte beachten Sie: Die Tabelle muss **vollständig** ausgefüllt werden. Insbesondere werden leere Zellen als Fehler gewertet. Sie dürfen das Symbol " $=$ " verwenden, wenn Sie den Inhalt der Zelle oberhalb der aktuellen Zelle in die aktuelle Zelle kopieren möchten (natürlich dürfen Sie den Inhalt auch nochmal abschreiben). (__ / 5 P)

2. Geben Sie den minimalen Spannbaum für obigen Graphen an. Zeichnen Sie dazu die entsprechenden Kanten im Bild unten ein. (__ / 1 P)



$\{u, v\}$	–								
$w(\{u, v\})$	–								
Dazu?	–								
$\mathrm{set}(a)$	$\{a\}$								
$\mathrm{set}(b)$	$\{b\}$								
$\mathrm{set}(c)$	$\{c\}$								
$\mathrm{set}(d)$	$\{d\}$								
$\mathrm{set}(e)$	$\{e\}$								
$\mathrm{set}(f)$	$\{f\}$								

8 Asymptotik (__ / 14 P)

8.1 Asymptotische Laufzeiten (__ / 9 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. (__ / 6 P)

   i) $f(n) \leq n^3 - 25n^2 + 123n$

   ii) $f(n) = 125n^5 + 3n^3 + n^2 + 5^n$

   iii) $f(n) \geq (n-25)^3 + 2^{\log_2(n)}$

   iv) $f(n) \leq 25n \log_4(n) + 36 \log_2(\log_2(n))$

   v) $f(n) = \begin{cases} 5^n, & n < 10000 \\ 13n^2 - n + 18, & n \geq 10000 \end{cases}$

   vi) $g(n) = n^5, f(n) \leq g(2g(n))$

2. Sie sollen eine Berechnung auf den Daten aller ca. 26000 Studierenden der TU Darmstadt durchführen. Sie haben zwei Algorithmen dazu zur Verfügung: Algorithmus A hat eine asymptotische Laufzeit von $\Theta(n^2 \log n)$, Algorithmus B hat eine asymptotische Laufzeit von $\Theta(n^3)$. Können Sie anhand dieser Informationen entscheiden, welcher Algorithmus schneller für Ihre Anwendung ist, und wenn ja, welcher? Begründen Sie Ihre Antwort.
(__ / 3 P)

8.2 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. $T(n) = 27T(n/3) + n^4$

2. $T(n) = 2T(n/4) + \log_2(n) + \sqrt{n}$