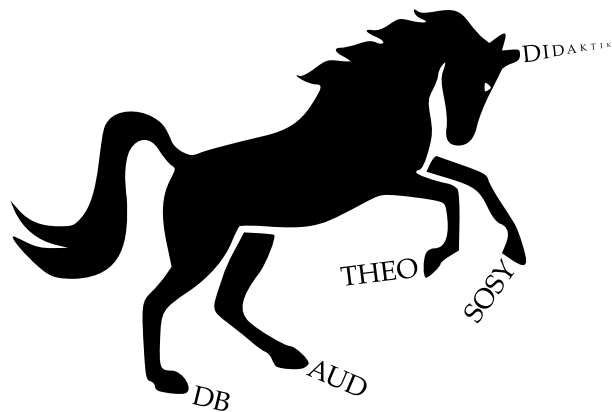


Die komplette Sammlung

Alle Aufgaben



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Inhaltsverzeichnis

46114 (Algorithmen / Datenstrukturen / Programmiermethoden (nicht vertieft))

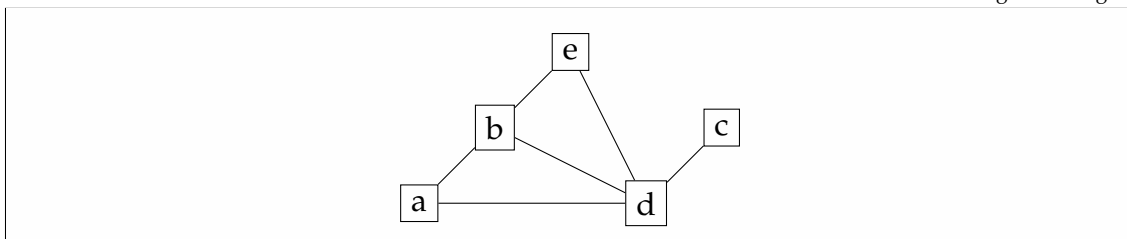
46114 / 2008 / Herbst / Thema 1 / Aufgabe 2

Gegeben sei folgender Graph:

V: {a, b, c, d, e}
 E: a → a, b
 b → b, d, e
 c → c, d
 d → a, e

(a) Stellen Sie den Graphen grafisch dar!

Lösungsvorschlag



(b) Berechnen Sie mit dem Algorithmus von Dijkstra schrittweise die Länge der kürzesten Pfade ab dem Knoten a! Nehmen Sie dazu an, dass alle Kantengewichte 1 sind. Erstellen Sie eine Tabelle gemäß folgendem Muster:

ausgewählt | a | b | c | d | e

Ergebnis:

Hinweis: Nur mit Angabe der jeweiligen Zwischenschritte gibt es Punkte. Es reicht also nicht, nur das Endergebnis hinzuschreiben.

Lösungsvorschlag

Nr.	ausgewählt	a	b	c	d	e
1	a	0	1	∞	1	∞
2	b		1	∞	1	2
3	d			2	1	2
4	c			2		2
5	e					2

(c) Welchen Aufwand hat der Algorithmus von Dijkstra bei Graphen mit $|V|$ Knoten und $|E|$ Kanten,

- wenn die Kantengewichte alle 1 sind? Mit welcher Datenstruktur und welchem Vorgehen lässt sich der Aufwand in diesem Fall reduzieren (mit kurzer Begründung)?
- wenn die Kantengewichte beliebig sind und als Datenstruktur eine Halde verwendet wird (mit kurzer Begründung)?

46114 / 2008 / Herbst / Thema 2 / Aufgabe 3

Quicksort ist ein Sortierungsverfahren, das nach dem Divide-and-Conquer-Prinzip (Teile und Herrsche) arbeitet. Wir betrachten im Folgenden die Anwendung dieses Verfahrens zum Sortieren von Integerzahlen. Die Sortierung soll in aufsteigender Reihenfolge der Werte erfolgen. Wir nehmen dabei an, dass die zu sortierenden Zahlen in einem Feld fester Länge abgelegt sind.

- Beschreiben Sie die Arbeitsweise des Divide-and-Conquer-Prinzips im allgemeinen Fall. Geben Sie dabei die Bedeutung der Schritte divide, conquer und combine an.
- Beschreiben Sie die Arbeitsweise des Algorithmus Quicksort. Geben Sie dabei an, worin die Schritte divide, conquer und combine im konkreten Fall bestehen.
- Geben Sie in C, C++ oder Java eine Implementierung des Algorithmus Quicksort an. Formulieren Sie die Implementierung als rekursive Funktion quicksort() und verwenden Sie das jeweils erste Element des (Teil-)Feldes für die Aufteilung. Verwenden Sie für Ihre Implementierung von quicksort() «lei Parameter:
 - das Feld, in dem die zu sortierenden Zahlen abgelegt sind;
 - den Index des am weitesten links gelegenen Elementes des zu sortierenden Teilfeldes;
 - den Index des am weitesten rechts gelegenen Elementes des zu sortierenden Teilfeldes.

Erläutern Sie die Arbeitsweise Ihrer Implementierung. Kennzeichnen Sie die Schritte divide, conquer und combine des zugrundeliegenden Divide-and-Conquer-Prinzips.

46115 (Theoretische Informatik / Algorithmen / Datenstrukturen (nicht vertieft))

46115 / 2010 / Frühjahr / Thema 1 / Aufgabe 5

5. Datenstrukturen und Algorithmen: Binäre Suchbäume und AVL-Bäume

- Geben Sie jeweils eine Definition für binäre Suchbäume und AVL-Bäume an.

Binärer Suchbaum Für jeden Knoten gilt: Ein Knoten enthält einen Schlüsselwert. Ein Knoten hat zwei Kindknoten: einen linken und einen rechten Kindknoten. Alle Schlüsselwerte des linken Teilbaums sind kleiner, alle Schlüsselwerte des rechten Teilbaums sind größer als der Wert des Knotens.

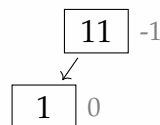
AVL-Baum Alle Eigenschaften des oben definierten binären Suchbaums gelten auch im AVL-Baum. Zusätzlich gilt: Die Differenz der Höhen des rechten und linken Teilbaums darf sich nie mehr als um eins unterscheiden.

- (b) Zeichnen Sie einen AVL-Baum, der die folgenden Schlüssel enthält: 11, 1, 5, 37, 17, 29, 31, 3.

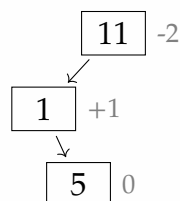
Nach dem Einfügen von „11“:



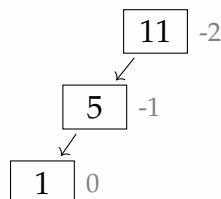
Nach dem Einfügen von „1“:



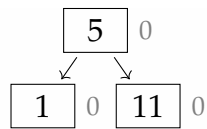
Nach dem Einfügen von „5“:



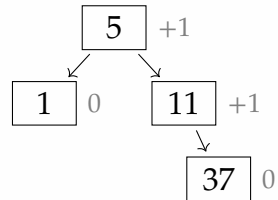
Nach der Linksrotation:



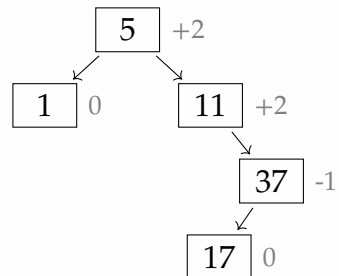
Nach der Rechtsrotation:



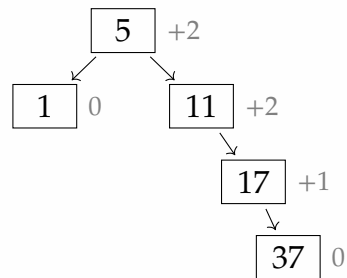
Nach dem Einfügen von „37“:



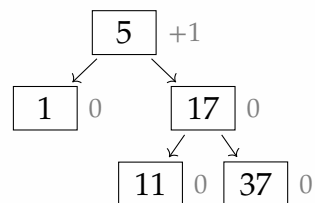
Nach dem Einfügen von „17“:



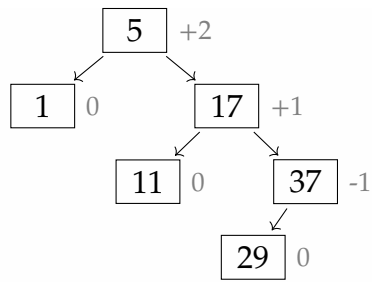
Nach der Rechtsrotation:



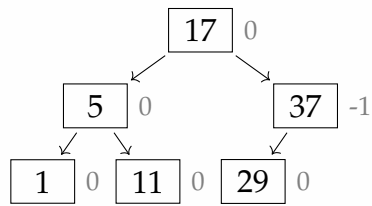
Nach der Linksrotation:



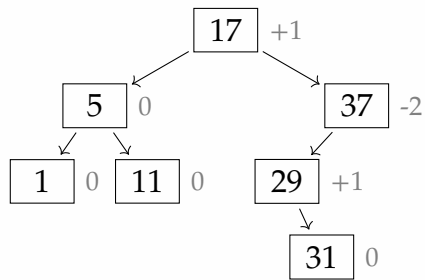
Nach dem Einfügen von „29“:



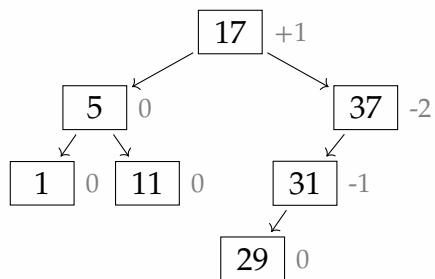
Nach der Linksrotation:



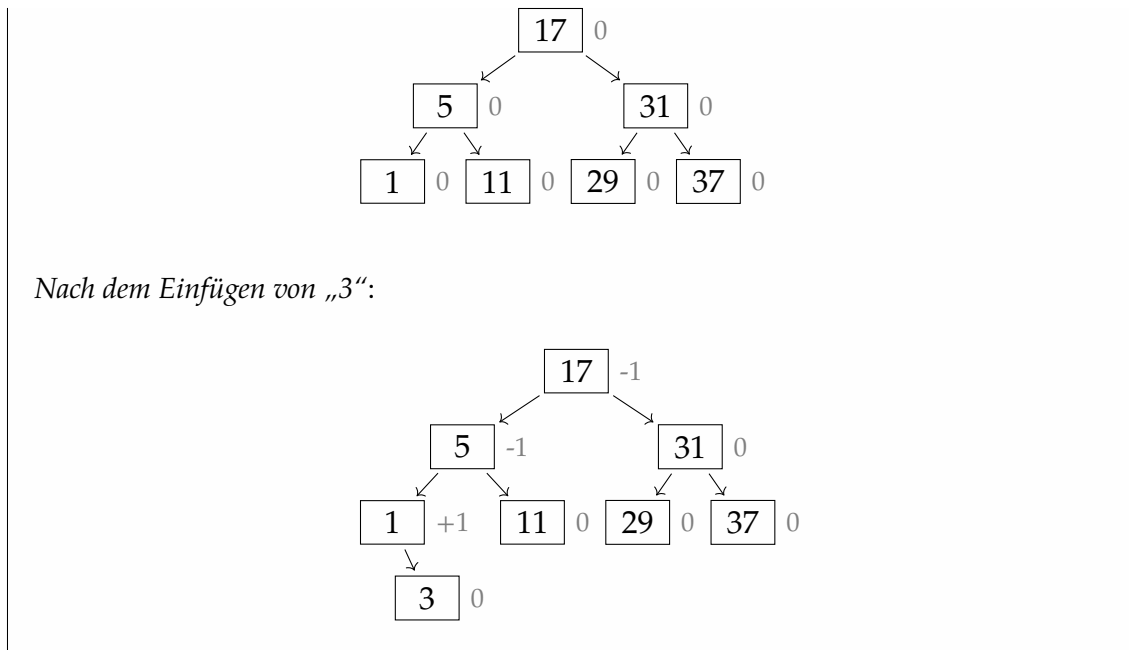
Nach dem Einfügen von „31“:



Nach der Linksrotation:



Nach der Rechtsrotation:



- (c) Welche Zeitkomplexität haben die Operationen *Einfügen*, *Löschen* und *Suchen* auf AVL-Bäumen? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Für alle Operationen wird jeweils nur ein Pfad von der Wurzel bis zum gewünschten Knoten beschriftet. Für alle Operation ist deshalb die maximale Höhe des Baums entscheidend. Da AVL-Bäume höhenbalanciert sind.

- (d) Implementieren Sie die Datenstruktur AVL-Baum mit Schlüsseln vom Typ `int` (für Java oder entsprechende Datentypen für die anderen Programmiersprachen) in entweder Java, C++, Smalltalk oder Eiffel. Ihre Implementierung muss als einzige Operation die Methode `suche` bereitstellen, die einen Schlüssel als Parameter bekommt und
- `true` zurückliefert, wenn der gesuchte Schlüssel im Baum enthalten ist,
 - ansonsten `false`.

Ihre Implementierung muss keine Konstruktoren oder andere Methoden zur Initialisierung bereitstellen. Desweiteren soll die Implementierung nur Schlüsselknoten berücksichtigen.

Kommentieren Sie Ihre Implementierung ausführlich. Geben Sie an, welche Programmiersprache Sie gewählt haben.

```
public class AVLBaum {
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2010/fruehjahr/AVLBaum.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2010/fruehjahr/AVLBaum.java)

46115 / 2010 / Frühjahr / Thema 2 / Aufgabe 1

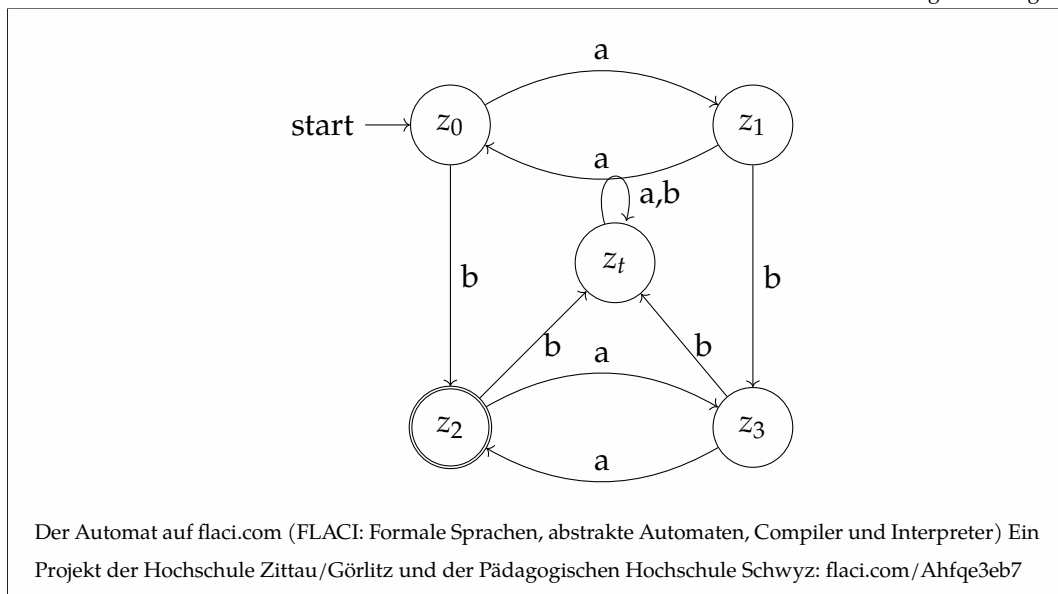
Aufgabe 1

(a) Gegeben ist die folgende Sprache $L1$ über dem Alphabet $\Sigma = \{a, b\}$:

$L1 = \{w \in \Sigma^* \mid \text{die Anzahl der } a \text{ in } w \text{ ist gerade und } b \text{ kommt in } w \text{ genau einmal vor}\}.$

(i) Geben Sie einen deterministischen endlichen Automaten an, der die Sprache $L1$ akzeptiert.

Lösungsvorschlag



(ii) Geben Sie einen regulären Ausdruck an, der die Sprache $L1$ beschreibt.

Lösungsvorschlag

$(aa)^*(b|aba)(aa)^*$

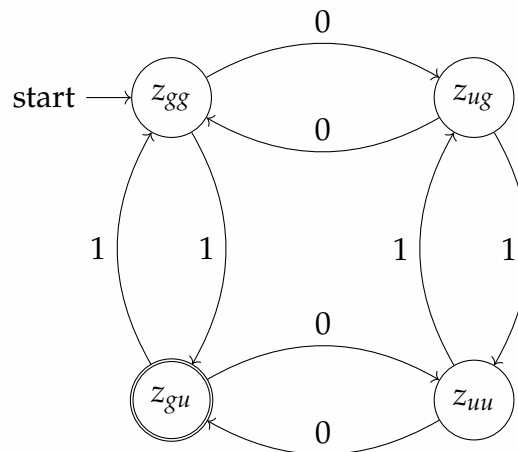
(b) Die folgende Sprache $L2$ ist eine Erweiterung von $L1$:

$L1 = \{w \in \Sigma^* \mid \text{die Anzahl der } a \text{ in } w \text{ ist gerade und die Anzahl der } b \text{ in } w \text{ ist ungerade}\}.$

(i) Geben Sie einen deterministischen endlichen Automaten an, der die Sprache $L2$ akzeptiert.

Lösungsvorschlag

a
 $gu = \text{gerade Anzahl } a\text{'s, ungerade Anzahl } b\text{'s}$
 $ug = \text{ungerade Anzahl } a\text{'s, gerade Anzahl } b\text{'s}$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Af0vcjys9

^ahttps://www.informatik.uni-hamburg.de/TGI/lehre/v1/SS14/FGI1/Folien/fgi1_v2_handout.pdf

(ii) Geben Sie eine rechtslineare Grammatik an, die die Sprache L_2 erzeugt.

Lösungsvorschlag

$$P = \left\{ \begin{array}{l} A \rightarrow aB \mid bD \mid b \\ B \rightarrow bC \mid aA \\ C \rightarrow aD \mid a \mid bB \\ D \rightarrow bA \mid aC \end{array} \right\}$$

<https://flaci.com/Ahfqe3eb7>

46115 / 2013 / Frühjahr / Thema 1 / Aufgabe 4

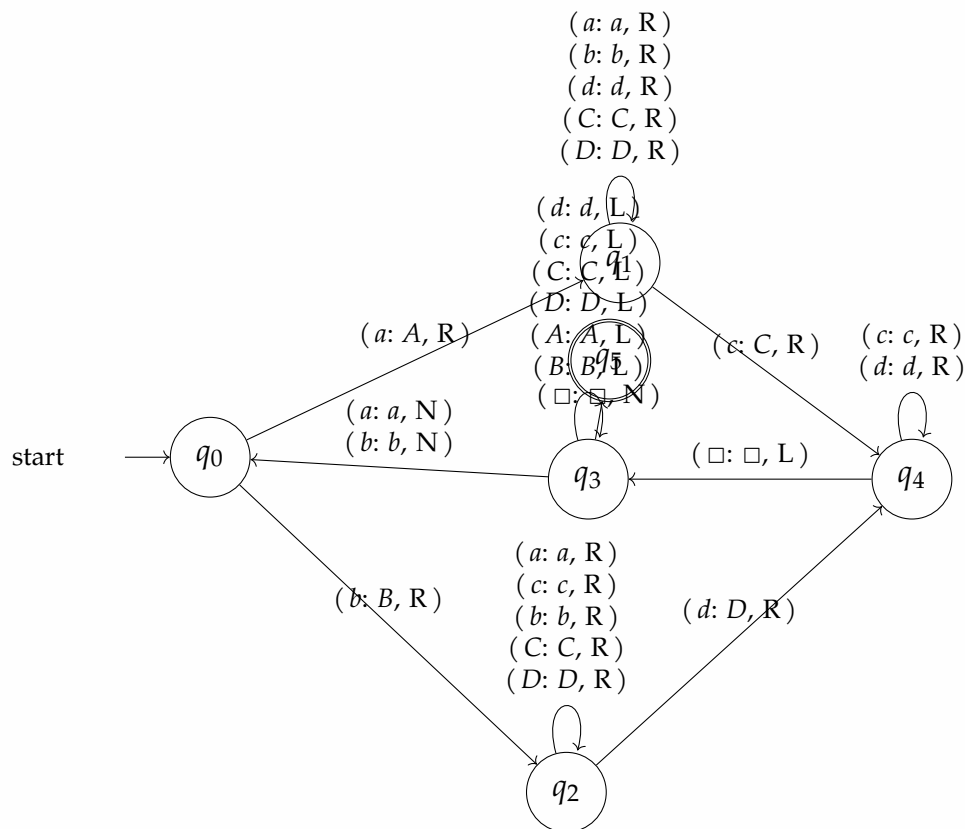
Aufgabe 4

Sei $L = \{ uv \mid u \in \{a, b\}^*, v \in \{c, d\}^*, \#_a(u) = \#_c(v) \text{ und } \#_b(u) = \#_d(v) \}$ wobei $\#_a(u)$ die Anzahl der in u vorkommenden a 's ist.

(a) Geben Sie eine Turingmaschine M an, die L erkennt. Beschreiben Sie in Worten, wie Ihre Turingmaschine arbeitet.

Lösungsvorschlag

noch nicht fertig ... akzeptiert auch abcd!d



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajhi5w0ha

- (b) Welche Laufzeit (Zeitkomplexität) hat Ihre Turingmaschine (in O-Notation). Begründen Sie Ihre Angabe auf der Grundlage der Beschreibung.

46115 / 2013 / Frühjahr / Thema 2 / Aufgabe 4

„Streuspeicherung“

Die Werte 7, 0, 9, 11, 18, 4, 5, 3, 13, 24, 2 sollen in eine Hashtabelle der Größe 11 (Fächer 0 bis 10) eingetragen werden. Die zur Hashfunktion $h(x) = (7 \cdot x) \% 11$ gehörenden Schlüssel sind in der folgenden Tabelle bereits ausgerechnet:

x	7	0	9	11	18	4	5	3	13	24	2
$h(x)$	5	0	8	0	5	6	2	10	3	3	3

- (a) Fügen Sie die oben genannten Schlüssel in der vorgegebenen Reihenfolge in einen Streuspeicher ein, welcher zur Kollisionsauflösung verkettete Listen verwendet, und stellen Sie die endgültige Streutabelle dar.

Index	0	1	2	3	4	5	6	7	8	9	10
Schlüssel	0		5	13		7	4		8		3
	11			24		18					
				2							

- (b) Fügen Sie die gleichen Schlüssel mit linearem Sondieren bei Schrittweite +1 zur Kollisionsauflösung in eine neue Hash-Tabelle ein. Geben Sie für jeden Schlüssel an, auf welche Felder beim Einfügen zugegriffen wird und ob Kollisionen auftreten. Geben Sie die gefüllte Streutabelle an.

Index	0	1	2	3	4	5	6	7	8	9	10
Schlüssel	0	11 ₁	5	13	24 ₁	7	18 ₁	4 ₁	9	2 ₆	3

- (c) Wie hoch ist der „Load“-Faktor (die Belegung) der Hashtabelle aus a) bzw. b) in Prozent? Können Sie weitere Schlüssel einfügen?

Teilaufgabe a)

$\frac{11}{11} = 100\%$: Es können allerdings weitere Elemente eingefügt werden. Die Verkettung lässt einen Loadfaktor über 100% zu. Der Suchaufwand wird dann jedoch größer.

Teilaufgabe b)

$\frac{11}{11} = 100\%$: Es können keine weiteren Elemente eingefügt werden, da alle Buckets belegt sind.

- (d) Würden Sie sich bei dieser Zahlensequenz für das Hashing-Verfahren nach a) oder nach b) entscheiden? Begründen Sie kurz Ihre Entscheidung.

Das Verfahren a) scheint hier sinnvoller, da noch nicht zu viele Suchoperationen notwendig sind (max. 2), während bei Verfahren b) einmal bereits 6-mal sondiert werden muss.

Aufgabe 10: Graphen I

Gegeben seien folgende ungerichtete Graphen in textueller Notation, wobei die erste Menge die Menge der Knoten und die zweite Menge die Menge der Kanten ist:

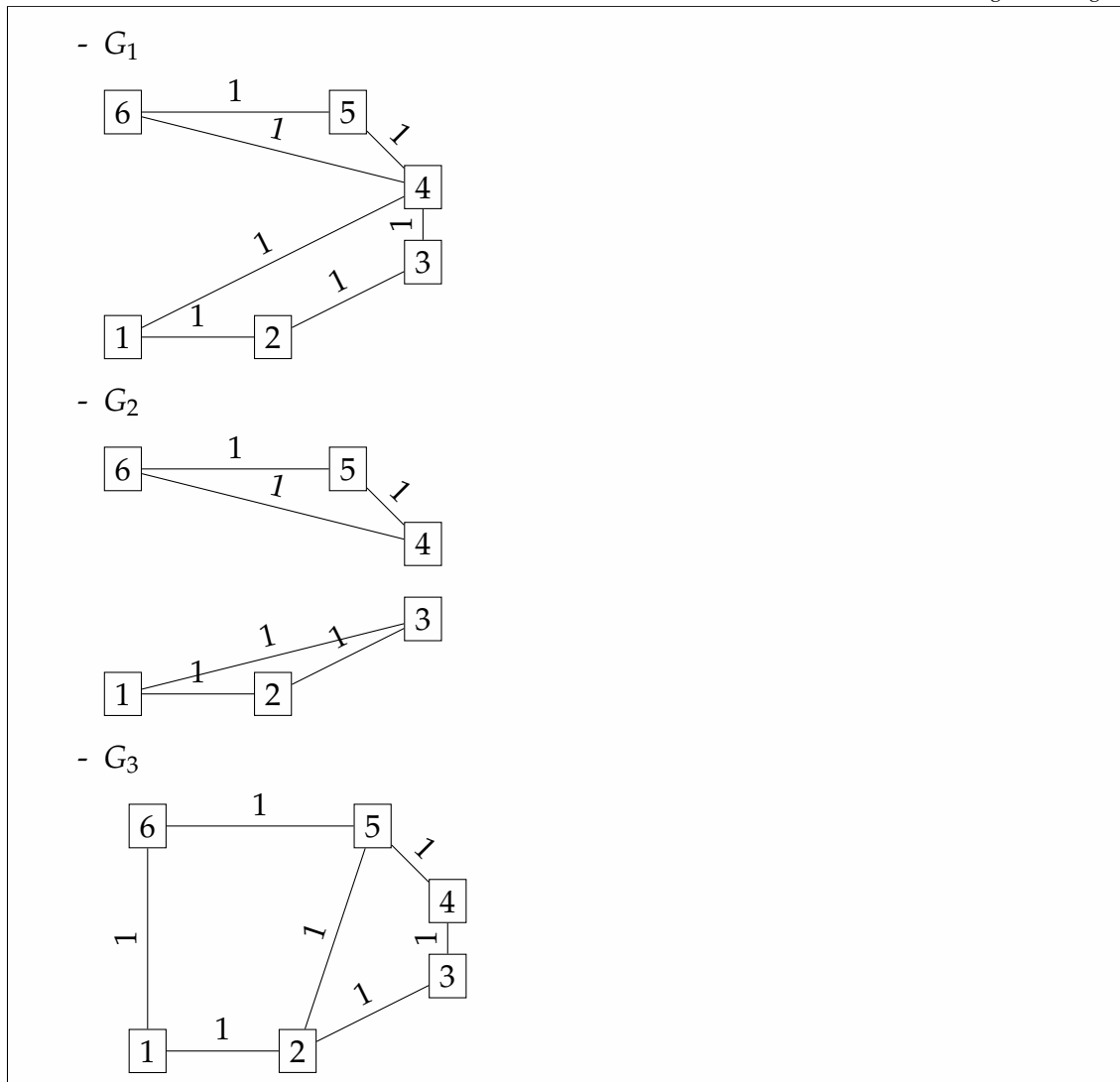
$$G_1 = (\{1, 2, 3, 4, 5, 6\}, \{[1, 2], [1, 4], [2, 3], [3, 4], [4, 5], [4, 6], [5, 6]\})$$

$$G_2 = (\{1, 2, 3, 4, 5, 6\}, \{[1, 2], [1, 3], [2, 3], [4, 5], [4, 6], [5, 6]\})$$

$$G_3 = (\{1, 2, 3, 4, 5, 6\}, \{[1, 2], [1, 6], [2, 3], [2, 5], [3, 4], [4, 5], [5, 6]\})$$

(a) Zeichnen Sie die obigen Graphen.

Lösungsvorschlag



(b) Erstellen Sie zu jedem Graphen die zugehörige Adjazenzmatrix mit X als Symbol für eine eingetragene Kante.

Lösungsvorschlag



$$\begin{array}{c}
 \begin{array}{cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
 1 & \left(\begin{array}{cccccc}
 - & X & & X & & \\
 X & - & X & & & \\
 & X & - & X & & \\
 X & & X & - & X & X \\
 & & & X & - & X \\
 & & & X & X & -
 \end{array} \right)
 \end{array}
 \end{array}$$

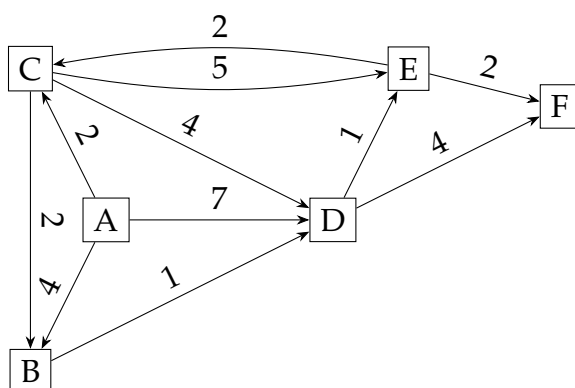
- G_2

$$\begin{array}{c}
 \begin{array}{cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
 1 & \left(\begin{array}{cccccc}
 - & X & X & & & \\
 X & - & X & & & \\
 X & X & - & & & \\
 & & & - & X & X \\
 & & & X & - & X \\
 & & & X & X & -
 \end{array} \right)
 \end{array}
 \end{array}$$

- G_3

$$\begin{array}{c}
 \begin{array}{cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
 1 & \left(\begin{array}{cccccc}
 - & X & & & X & X \\
 X & - & X & & & \\
 & X & - & X & & \\
 & & X & - & X & \\
 & X & & X & - & X \\
 X & & & & X & -
 \end{array} \right)
 \end{array}
 \end{array}$$

(c) Betrachten Sie nun folgenden gerichteten Graphen G_4 :



Bestimmen Sie die kürzeste Entfernung von Knoten A zu jedem anderen Knoten des Graphen. Verwenden Sie dazu den Algorithmus von Dijkstra und tragen Sie Ihre einzelnen Rechenschritte in eine Tabelle folgender Form ein (schreiben Sie neben jede Zeile die Prioritätswarteschlange der noch zu bearbeitenden Knoten, priorisiert nach ihren Wegkosten):

Hinweis: Mit den „Wegkosten“ eines Knotens ist die gegenwärtige Entfernung dieses Knotens vom Startknoten gemeint.

Lösungsvorschlag

A	B	C	D	E	F	Warteschlange
0	∞	∞	∞	∞	∞	A
	4 (A)	2 (A)	7 (A)	∞	∞	C, B, D
			6 (C)	7 (C)	∞	B, D, E
			5 (B)	7 (C)	∞	D, E
				6 (D)	9 (D)	E, F
					8 (E)	F

46115 / 2013 / Frühjahr / Thema 2 / Aufgabe 6

Aufgabe 6

- (a) Vervollständigen Sie die folgende Sortierung mit MergeSort (Sortieren durch Mischen) — beginnen Sie dabei Ihren „rekursiven Abstieg“ immer im linken Teilfeld:

D | 40 5 89 95 85 84 || 14 25 20 52 7 71 |

Notation: Markieren Sie Zeilen mit D(ivide), in denen das Array zerlegt wird, und mit M(erge), in denen Teilarrays zusammengeführt werden. Beispiel:

D | 82 || 89 44 |

D 82 | 89 || 44 |

M 82 | 44 89 |

M | 44 82 89 |

Lösungsvorschlag

D		40	5		89	95	85	84		14	25	20	52	7	71					
D		40	5		89		95	85	84											
D		40	5		89															
D		40		5																
M		5		40																
M		5		40		89														
D							95	85	84											
D							95	85		84										
D							95		85											
M							85	95												
M							84	85	95											
M		5		40	84	85	89	95												
D											14	25	20		52	7	71			
D											14	25		20						
D											14		25							
M											14	25								
M											14	20	25							
D																52	7		71	
D																52		7		
M																7	52			
M																7	52	71		
M											7	14	20			25	52	71		
M											7	14	20			25	52	71		
M		5		7	14	20	25	40	52	71	84	85	89	95						

- (b) Sortieren Sie mittels HeapSort (Haldensortierung) die folgende Liste weiter: Notation: Markieren Sie die Zeilen wie folgt:

I: Initiale Heap-Eigenschaft hergestellt (größtes Element am Anfang der Liste).

R: Erstes und letztes Element getauscht und letztes „gedanklich entfernt“.

S: Erstes Element nach unten „versickert“ (Heap-Eigenschaft wiederhergestellt).

Lösungsvorschlag

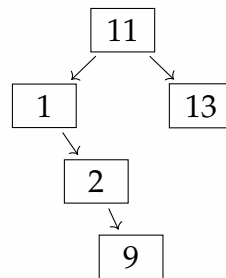
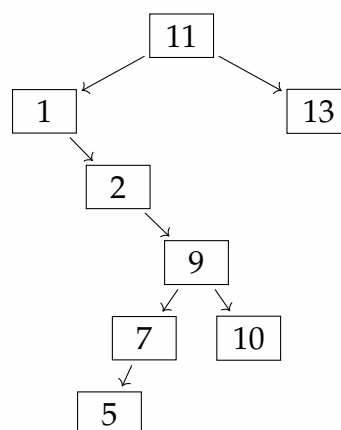
I		99	63	91	4	36	81	76																	
R		76	63	91	4	36	81		99																
S		91	63	81	4	36	76		99																
R		76	63	81	4	36		91	99																
S		81	76	63	4	36		91	99																
R		36	76	63	4		81	91	99																
S		76	36	63	4		81	91	99																
R		4	36	63		76	81	91	99																
S		63	4	36		76	81	91	99																
R		4	36		63	76	81	91	99																
S		36	4		63	76	81	91	99																
R		4		36	63	76	81	91	99																
S		4		36	63	76	81	91	99																
R		4	36	63	76	81	91	99																	

46115 / 2014 / Frühjahr / Thema 1 / Aufgabe 7

Fügen Sie nacheinander die Zahlen 11, 1, 2, 13, 9, 10, 7, 5

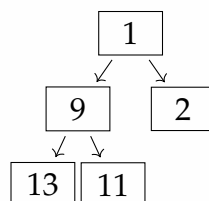
- (a) in einen leeren binären Suchbaum und zeichnen Sie den Suchbaum jeweils nach dem Einfügen von „9“ und „5“

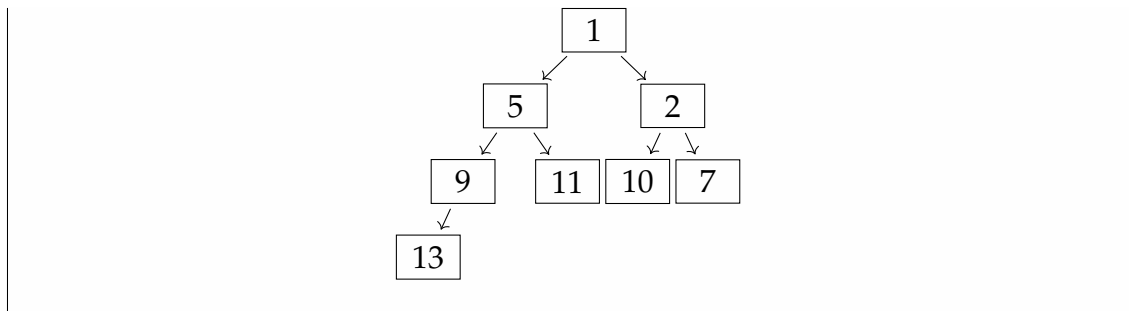
Lösungsvorschlag

Nach dem Einfügen von „9“:*Nach dem Einfügen von „5“:*

- (b) in einen leeren Min-Heap ein, der bzgl. „ \leq “ angeordnet ist und geben Sie den Heap nach „9“ und nach „5“ an

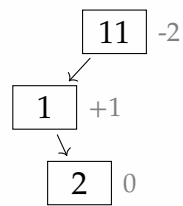
Lösungsvorschlag

Nach dem Einfügen von „9“:*Nach dem Einfügen von „5“:*

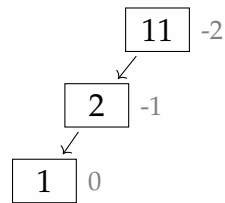


- (c) in einen leeren AVL-Baum ein! Geben Sie den AVL Baum nach „2“ und „5“ an und beschreiben Sie die ggf. notwendigen Rotationen beim Einfügen dieser beiden Elemente!

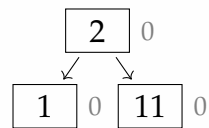
Nach dem Einfügen von „2“:



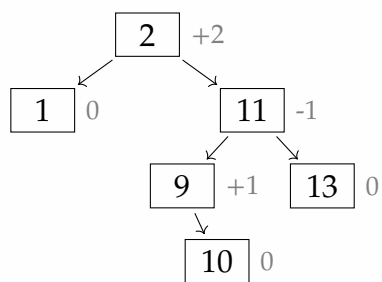
Nach der Linksrotation:



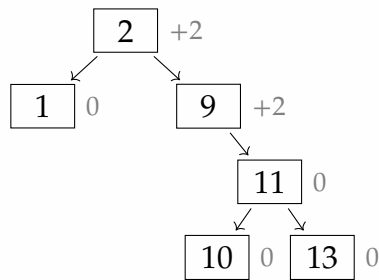
Nach der Rechtsrotation:



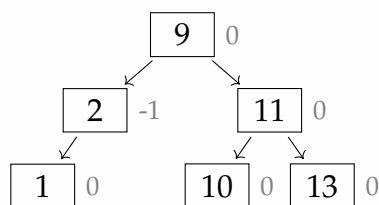
Nach dem Einfügen von „10“:



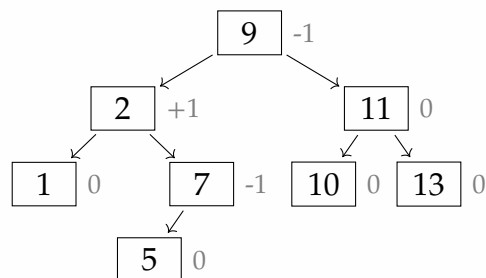
Nach der Rechtsrotation:



Nach der Linksrotation:



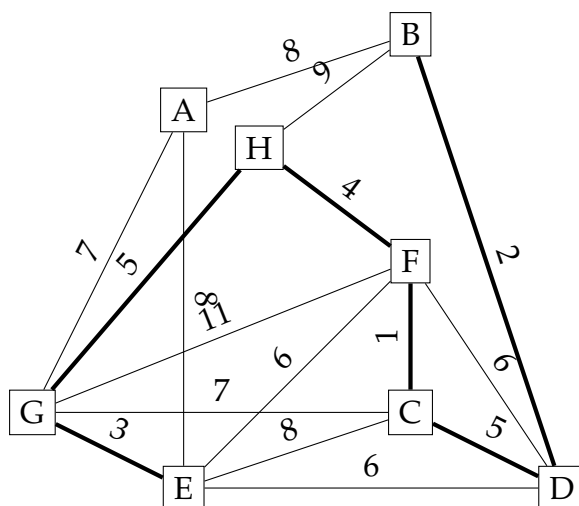
Nach dem Einfügen von „5“:



Frühjahr 2014 (46115) - Thema 1 Aufgabe 8

Bestimmen Sie einen minimalen Spannbaum für einen ungerichteten Graphen, der durch die nachfolgende Entfernungsmatrix gegeben ist! Die Matrix ist symmetrisch und ∞ bedeutet, dass es keine Kante gibt. Zeichnen Sie den Graphen und geben Sie die Spannbaumkanten ein !

	A	B	C	D	E	F	G	H
A	0	8	-1	∞	8	∞	7	∞
B	8	0	∞	2	∞	∞	∞	9
C	-1	∞	0	5	8	1	7	∞
D	∞	2	5	0	6	6	∞	∞
E	8	∞	8	6	0	6	3	∞
F	∞	∞	1	6	6	0	11	4
G	7	∞	7	∞	3	11	0	5
H	∞	9	∞	∞	∞	4	5	0



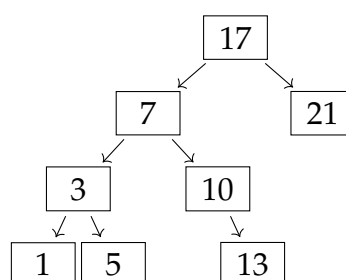
Kante	Gewicht
AC	-1
BD	2
CF	1
EG	3
FH	4
GH	5
CD	5
	19

Nach dem Algorithmus von Kruskal wählt man aus den noch nicht gewählten Kanten immer die kürzeste, die keinen Kreis mit den bisher gewählten Kanten bildet.

46115 / 2014 / Frühjahr / Thema 2 / Aufgabe 3

Frühjahr 2014 (46115) - Thema 2 Aufgabe 3

- (a) Fügen Sie die Zahlen 17, 7, 21, 3, 10, 13, 1, 5 nacheinander in der vorgegebenen Reihenfolge in einen binären Suchbaum ein und zeichnen Sie das Ergebnis!



- (b) Implementieren Sie in einer objektorientierten Programmiersprache eine rekursiv festgelegte Datenstruktur, deren Gestaltung sich an folgender Definition eines binären Baumes orientiert!

Ein binärer Baum ist entweder ein leerer Baum oder besteht aus einem Wurzelement, das einen binären Baum als linken und einen als rechten Teilbaum besitzt. Bei dieser Teilaufgabe können Sie auf die Implementierung von Methoden (außer ggf. notwendigen Konstruktoren) verzichten!

Klasse Knoten

```
class Knoten {
    public int wert;
    public Knoten links;
    public Knoten rechts;
    public Knoten elternKnoten;

    Knoten(int wert) {
        this.wert = wert;
        links = null;
        rechts = null;
        elternKnoten = null;
    }

    public Knoten findeMiniumRechterTeilbaum() {
    }

    public void anhängen (Knoten knoten) {
    }
}

public class BinärerSuchbaum {
    public Knoten wurzel;

    BinärerSuchbaum(Knoten wurzel) {
        this.wurzel = wurzel;
    }

    BinärerSuchbaum() {
        this.wurzel = null;
    }

    public void einfügen(Knoten knoten) {
    }

    public void einfügen(Knoten knoten, Knoten elternKnoten) {
    }

    public Knoten suchen(int wert) {
    }

    public Knoten suchen(int wert, Knoten knoten) {
    }
}
```

```
}  
  
}
```

- (c) Beschreiben Sie durch Implementierung in einer gängigen objektorientierten Programmiersprache, wie bei Verwendung der obigen Datenstruktur die Methode `loescheKnoten(w)` gestaltet sein muss, mit der der Knoten mit dem Eintrag `w` aus dem Baum entfernt werden kann, ohne die Suchbaumeigenschaft zu verletzen!

Lösungsvorschlag

```
public void loescheKnoten(int w) {  
    Knoten knoten = suchen(w);  
    if (knoten == null) return;  
    // Der Knoten hat keine Teilbäume.  
    if (knoten.links == null && knoten.rechts == null) {  
        if (w < knoten.elternKnoten.wert) {  
            knoten.elternKnoten.links = null;  
        } else {  
            knoten.elternKnoten.rechts = null;  
        }  
    }  
  
    // Der Knoten besitzt einen Teilbaum.  
    // links  
    else if (knoten.links != null && knoten.rechts == null) {  
        knoten.elternKnoten.anhängen(knoten.links);  
    }  
    // rechts  
    else if (knoten.links == null) {  
        knoten.elternKnoten.anhängen(knoten.rechts);  
    }  
  
    // Der Knoten besitzt zwei Teilbäume.  
    else {  
        Knoten minimumKnoten = knoten.findeMinimumRechterTeilbaum();  
        minimumKnoten.links = knoten.links;  
        minimumKnoten.rechts = knoten.rechts;  
        knoten.elternKnoten.anhängen(minimumKnoten);  
    }  
}
```

Code-Beispiel auf Github ansehen:
[src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/suchbaum/BinaererSuchbaum.java](https://github.com/bschlangaul/examen_46115_jahr_2014/fruehjahr/suchbaum/BinaererSuchbaum.java)

Klasse „BinärerSuchbaum“

```
public class BinaererSuchbaum {  
    public Knoten wurzel;  
  
    BinaererSuchbaum(Knoten wurzel) {  
        this.wurzel = wurzel;  
    }  
}
```



```
BinaererSuchbaum() {
    this.wurzel = null;
}

public void einfuegen(Knoten knoten) {
    if (wurzel != null) {
        einfuegen(knoten, wurzel);
    } else {
        wurzel = knoten;
        knoten.elternKnoten = wurzel;
    }
}

public void einfuegen(int wert) {
    einfuegen(new Knoten(wert));
}

public void einfuegen(Knoten knoten, Knoten elternKnoten) {
    if (knoten.wert <= elternKnoten.wert) {
        if (elternKnoten.links != null) {
            einfuegen(knoten, elternKnoten.links);
        } else {
            elternKnoten.links = knoten;
            knoten.elternKnoten = elternKnoten;
        }
    } else {
        if (elternKnoten.rechts != null) {
            einfuegen(knoten, elternKnoten.rechts);
        } else {
            elternKnoten.rechts = knoten;
            knoten.elternKnoten = elternKnoten;
        }
    }
}

public Knoten suchen(int wert) {
    if (wurzel == null || wurzel.wert == wert) {
        return wurzel;
    } else {
        return suchen(wert, wurzel);
    }
}

public Knoten suchen(int wert, Knoten knoten) {
    if (knoten.wert == wert) {
        return knoten;
    } else if (wert < knoten.wert && knoten.links != null) {
        return suchen(wert, knoten.links);
    } else if (wert > knoten.wert && knoten.rechts != null) {
        return suchen(wert, knoten.rechts);
    }
    return null;
}
```

```
public void loescheKnoten(int w) {
    Knoten knoten = suchen(w);
    if (knoten == null) return;
    // Der Knoten hat keine Teilbäume.
    if (knoten.links == null && knoten.rechts == null) {
        if (w < knoten.elternKnoten.wert) {
            knoten.elternKnoten.links = null;
        } else {
            knoten.elternKnoten.rechts = null;
        }
    }

    // Der Knoten besitzt einen Teilbaum.
    // links
    else if (knoten.links != null && knoten.rechts == null) {
        knoten.elternKnoten.anhängen(knoten.links);
    }
    // rechts
    else if (knoten.links == null) {
        knoten.elternKnoten.anhängen(knoten.rechts);
    }

    // Der Knoten besitzt zwei Teilbäume.
    else {
        Knoten minimumKnoten = knoten.findeMiniumRechterTeilbaum();
        minimumKnoten.links = knoten.links;
        minimumKnoten.rechts = knoten.rechts;
        knoten.elternKnoten.anhängen(minimumKnoten);
    }
}

// Der Baum aus dem Foliensatz
public BinaererSuchbaum erzeugeTestBaum() {
    BinaererSuchbaum binärerSuchbaum = new BinaererSuchbaum();
    binärerSuchbaum.einfügen(new Knoten(7));
    binärerSuchbaum.einfügen(new Knoten(3));
    binärerSuchbaum.einfügen(new Knoten(11));
    binärerSuchbaum.einfügen(new Knoten(2));
    binärerSuchbaum.einfügen(new Knoten(6));
    binärerSuchbaum.einfügen(new Knoten(9));
    binärerSuchbaum.einfügen(new Knoten(1));
    binärerSuchbaum.einfügen(new Knoten(5));
    return binärerSuchbaum;
}

public void ausgebenInOrder() {
}

public static void main(String[] args) {
    BinaererSuchbaum binärerSuchbaum = new BinaererSuchbaum();
    BinaererSuchbaum testBaum = binärerSuchbaum.erzeugeTestBaum();
}
```

```

// Teste das Einfügen

System.out.println(testBaum.wurzel.wert); // 7
System.out.println(testBaum.wurzel.links.wert); // 3
System.out.println(testBaum.wurzel.links.links.wert); // 2
System.out.println(testBaum.wurzel.links.rechts.wert); // 6
System.out.println(testBaum.wurzel.rechts.wert); // 11

// Teste das Suchen

System.out.println("Gesucht nach 5 und gefunden: " + testBaum.suchen(5).wert);
System.out.println("Gesucht nach 9 und gefunden: " + testBaum.suchen(9).wert);
System.out.println("Gesucht nach 7 und gefunden: " + testBaum.suchen(7).wert);
System.out.println("Gesucht nach 10 und gefunden: " + testBaum.suchen(10));

// Teste das Löschen

// Der Knoten hat keine Teilbäume.
System.out.println("Noch nicht gelöschter Knoten 9: " +
    ↪ testBaum.suchen(9).wert);
testBaum.loescheKnoten(9);
System.out.println("Gelöschter Knoten 9: " + testBaum.suchen(9));

// Der Knoten hat einen Teilbaum.
// fristen Testbaum erzeugen.
testBaum = binärerSuchbaum.erzeugeTestBaum();
Knoten elternKnoten = testBaum.suchen(3);
System.out.println("Rechts Kind von 3 vor dem Löschen: " +
    ↪ elternKnoten.rechts.wert);
testBaum.loescheKnoten(6);
System.out.println("Rechts Kind von 3Nach dem Löschen: " +
    ↪ elternKnoten.rechts.wert);

// Der Knoten hat zwei Teilbäume.
// fristen Testbaum erzeugen.
testBaum = binärerSuchbaum.erzeugeTestBaum();
Knoten wurzel = testBaum.wurzel;
System.out.println("Linkes Kind der Wurzel vor dem Löschen: " +
    ↪ wurzel.links.wert); // 5
testBaum.loescheKnoten(3);
System.out.println("Linkes Kind der WurzelNach dem Löschen: " +
    ↪ wurzel.links.wert); // 3
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/suchbaum/BinaererSuchbaum.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/suchbaum/BinaererSuchbaum.java)

Klasse „Knoten“

```

class Knoten {
    public int wert;
    public Knoten links;
    public Knoten rechts;
    public Knoten elternKnoten;
}

```

```

Knoten(int wert) {
    this.wert = wert;
    links = null;
    rechts = null;
    elternKnoten = null;
}

public Knoten findeMinimumRechterTeilbaum() {
    if (rechts != null) {
        Knoten minimumKnoten = rechts;
        while (minimumKnoten.links != null) {
            minimumKnoten = minimumKnoten.links;
        }
        return minimumKnoten;
    }
    return null;
}

public void anhängen (Knoten knoten) {
    if (knoten.wert < wert) {
        links = knoten;
    } else {
        rechts = knoten;
    }
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/suchbaum/Knoten.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/suchbaum/Knoten.java)

46115 / 2014 / Frühjahr / Thema 2 / Aufgabe 4

Aufgabe 4

Für Binomialkoeffizienten $\binom{n}{k}$ gelten neben den grundlegenden Beziehungen $\binom{n}{0} = 1$ und $\binom{n}{n} = 1$ auch folgende Formeln:

Exkurs: Binomialkoeffizient

Der Binomialkoeffizient ist eine mathematische Funktion, mit der sich eine der Grundaufgaben der Kombinatorik lösen lässt. Er gibt an, auf wie viele verschiedene Arten man k bestimmte Objekte aus einer Menge von n verschiedenen Objekten auswählen kann (ohne Zurücklegen, ohne Beachtung der Reihenfolge). Der Binomialkoeffizient ist also die Anzahl der k -elementigen Teilmengen einer n -elementigen Menge.^a

^a<https://de.wikipedia.org/wiki/Binomialkoeffizient>

A $\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$

B $\binom{n}{k} = \binom{n-1}{k-1} \cdot \frac{n}{k}$

- (a) Implementieren Sie unter Verwendung von Beziehung (A) eine rekursive Methode `binRek(n, k)` zur Berechnung des Binomialkoeffizienten in einer objektorientierten Programmiersprache oder entsprechendem Pseudocode!

Lösungsvorschlag

Zuerst verwandeln wir die Beziehung (A) geringfügig um, indem wir n durch $n - 1$ ersetzen:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

```
public static int binRek(int n, int k) {
    if (k == 0 || k == n) {
        return 1;
    } else {
        return binRek(n - 1, k - 1) + binRek(n - 1, k);
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/Binomialkoeffizient.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/Binomialkoeffizient.java)

- (b) Implementieren Sie unter Verwendung von Beziehung (B) eine iterative Methode `binIt(n, k)` zur Berechnung des Binomialkoeffizienten in einer objektorientierten Programmiersprache oder entsprechendem Pseudocode!

Lösungsvorschlag

```
public static int binIt(int n, int k) {
    // Das Ergebnis wird als Kommazahl deklariert, da nicht alle
    // Zwischenergebnisse ganze Zahlen sind.
    double ergebnis = 1;
    while (k > 0) {
        ergebnis = ergebnis * n / k;
        n--;
        k--;
    }
    // Vor dem Zurückgeben kann das Ergebnis nun in eine ganze Zahl
    // umgewandelt werden.
    return (int) ergebnis;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/Binomialkoeffizient.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/Binomialkoeffizient.java)

- (c) Geben Sie die Laufzeitkomplexität der Methoden `binRek(n, k)` und `binIt(n, k)` aus den vorhergehenden beiden Teilaufgaben in O-Notation an!

Komplette Java-Klasse

```
/**
 * <a href="https://www.studon.fau.de/file2889270_download.html">Angabe:
 * ↪ PUE_AUD_WH.pdf</a>
 * <a href="https://www.studon.fau.de/file3081306_download.html">Lösung:
 * ↪ PUE_AUD_WH_Lsg.pdf</a>
```

```
*/
public class Binomialkoeffizient {

    /**
     * Berechnet rekursiv den Binominalkoeffizienten „n über k“. Dabei muss gelten:
     * n &#x3E;= 0, k &#x3E;= 0 und n &#x3E;= k.
     *
     * @param n Ganzzahl n
     * @param k Ganzzahl k
     *
     * @return Eine Ganzzahl.
     */
    public static int binRek(int n, int k) {
        if (k == 0 || k == n) {
            return 1;
        } else {
            return binRek(n - 1, k - 1) + binRek(n - 1, k);
        }
    }

    /**
     * Berechnet iterativ den Binominalkoeffizienten „n über k“. Dabei muss gelten:
     * n &#x3E;= 0, k &#x3E;= 0 und n &#x3E;= k.
     *
     * @param n Ganzzahl n
     * @param k Ganzzahl k
     *
     * @return Eine Ganzzahl.
     */
    public static int binIt(int n, int k) {
        // Das Ergebnis wird als Kommazahl deklariert, da nicht alle
        // Zwischenergebnisse ganze Zahlen sind.
        double ergebnis = 1;
        while (k > 0) {
            ergebnis = ergebnis * n / k;
            n--;
            k--;
        }
        // Vor dem Zurückgeben kann das Ergebnis nun in eine ganze Zahl
        // umgewandelt werden.
        return (int) ergebnis;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/Binomialkoeffizient.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/Binomialkoeffizient.java)

Test

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class BinomialkoeffizientTest {

    public void testeRek(int n, int k, int ergebnis) {
```

```

    assertEquals(ergebnis, Binomialkoeffizient.binIt(n, k));
}

public void testeIt(int n, int k, int ergebnis) {
    assertEquals(ergebnis, Binomialkoeffizient.binIt(n, k));
}

public void teste(int n, int k, int ergebnis) {
    testeRek(n, k, ergebnis);
    testeIt(n, k, ergebnis);
}

@Test
public void teste() {
    teste(0, 0, 1);

    teste(1, 0, 1);
    teste(1, 1, 1);

    teste(2, 0, 1);
    teste(2, 1, 2);
    teste(2, 2, 1);

    teste(3, 0, 1);
    teste(3, 1, 3);
    teste(3, 2, 3);
    teste(3, 3, 1);

    teste(4, 0, 1);
    teste(4, 1, 4);
    teste(4, 2, 6);
    teste(4, 3, 4);
    teste(4, 4, 1);
}
}

```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/BinomialkoeffizientTest.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/BinomialkoeffizientTest.java)

46115 / 2015 / Herbst / Thema 1 / Aufgabe 3

Aufgabe 3

Eine Folge von Zahlen a_1, \dots, a_n heie unimodal, wenn sie bis zu einem bestimmten Punkt echt ansteigt und dann echt fllt. Zum Beispiel ist die Folge 1, 3, 5, 6, 5, 2, 1 unimodal, die Folgen 1, 3, 5, 4, 7, 2, 1 und 1, 2, 3, 3, 4, 3, 2, 1 aber nicht.

Exkurs: Unimodale Abbildung

Eine unimodale Abbildung oder unimodale Funktion ist in der Mathematik eine Funktion mit einem eindeutigen (lokalen und globalen) Maximum wie zum Beispiel $f(x) = -x^2$.^a

^ahttps://de.wikipedia.org/wiki/Unimodale_Abbildung

- (a) Entwerfen Sie einen Algorithmus, der zu (als Array) gegebener unimodaler Folge a_1, \dots, a_n in Zeit $\mathcal{O}(\log n)$ das Maximum $\max a_i$ berechnet. Ist die Folge nicht unimodal, so kann Ihr Algorithmus ein beliebiges Ergebnis liefern. Größenvergleiche, arithmetische Operationen und Arrayzugriffe können wie üblich in konstanter Zeit ($\mathcal{O}(1)$) getätigt werden. Hinweise: binäre Suche, divide-and-conquer.

Lösungsvorschlag

Wir wählen einen Wert in der Mitte der Folge aus. Ist der direkte linke und der direkte rechte Nachbar dieses Wertes kleiner, dann ist das Maximum gefunden. Ist nur linke Nachbar größer, setzen wir die Suche wie oben beschrieben in der linken Hälfte, sonst in der rechten Hälfte fort.

- (b) Begründen Sie, dass Ihr Algorithmus tatsächlich in Zeit $\mathcal{O}(\log n)$ läuft.

Lösungsvorschlag

Da der beschriebene Algorithmus nach jedem Bearbeitungsschritt nur auf der Hälfte der Feld-Elemente zu arbeiten hat, muss im schlechtesten Fall nicht die gesamte Folge durchsucht werden. Nach dem ersten Teilen der Folge bleiben nur noch $\frac{n}{2}$ Elemente, nach dem zweiten Schritt $\frac{n}{4}$, nach dem dritten $\frac{n}{8}$ usw. Allgemein bedeutet dies, dass im i -ten Durchlauf maximal $\frac{n}{2^i}$ Elemente zu durchsuchen sind. Entsprechend werden $\log_2 n$ Schritte benötigt. Somit hat der Algorithmus zum Finden des Maximums in einer unimodalen Folge in der Landau-Notation ausgedrückt die Zeitkomplexität $\mathcal{O}(\log n)$.

- (c) Schreiben Sie Ihren Algorithmus in Pseudocode oder in einer Programmiersprache Ihrer Wahl, z. B. Java, auf. Sie dürfen voraussetzen, dass die Eingabe in Form eines Arrays der Größe n vorliegt.

Lösungsvorschlag

Rekursiver Ansatz

```
public static int findeMaxRekursiv(int feld[], int links, int rechts) {
    if (links == rechts - 1) {
        return feld[links];
    }
    // bedeutet aufrunden
    // https://stackoverflow.com/a/17149572
    int mitte = (int) Math.ceil((double) (links + rechts) / 2);
    if (feld[mitte - 1] < feld[mitte]) {
        return findeMaxRekursiv(feld, mitte, rechts);
    } else {
        return findeMaxRekursiv(feld, links, mitte);
    }
}

public static int findeMaxRekursiv(int feld[]) {
    return findeMaxRekursiv(feld, 0, feld.length - 1);
}
```


Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java](#)

Iterativer Ansatz

```
public static int findeMaxIterativ(int[] feld) {
    int links = 0;
    int rechts = feld.length - 1;
    int mitte;

    while (links < rechts) {
        mitte = links + (rechts - links) / 2;
        if (feld[mitte] > feld[mitte - 1] && feld[mitte] > feld[mitte + 1]) {
            return feld[mitte];
        } else if (feld[mitte] > feld[mitte - 1]) {
            links = mitte + 1;
        } else {
            rechts = mitte - 1;
        }
    }
    return KEIN_MAX;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java](#)

- (d) Beschreiben Sie in Worten ein Verfahren, welches in Zeit $\mathcal{O}(n)$ feststellt, ob eine vorgelegte Folge unimodal ist oder nicht.

Lösungsvorschlag

```
public static boolean testeUnimodalität(int[] feld) {
    if (feld.length < 2) {
        // Die Reihe braucht mindestens 3 Einträge
        return false;
    }

    if (feld[0] > feld[1]) {
        // Die Reihe muss zuerst ansteigen
        return false;
    }

    boolean maxErreicht = false;
    for (int i = 0; i < feld.length - 1; i++) {
        if (feld[i] > feld[i + 1] && !maxErreicht) {
            maxErreicht = true;
        }

        if (maxErreicht && feld[i] < feld[i + 1]) {
            // Das Maximum wurde bereits erreicht und die nächste Zahl ist
            // → größer
            return false;
        }
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java)

- (e) Begründen Sie, dass es kein solches Verfahren (Test auf Unimodalität) geben kann, welches in Zeit $\mathcal{O}(\log n)$ läuft.

Lösungsvorschlag

Da die Unimodalität nur durch einen Werte an einer beliebigen Stelle der Folge verletzt werden kann, müssen alle Elemente durchsucht und überprüft werden.

Lösungsvorschlag

Komplette Klasse

```
public static int findeMaxRekursiv(int feld[], int links, int rechts) {
    if (links == rechts - 1) {
        return feld[links];
    }
    // bedeutet aufrunden
    // https://stackoverflow.com/a/17149572
    int mitte = (int) Math.ceil((double) (links + rechts) / 2);
    if (feld[mitte - 1] < feld[mitte]) {
        return findeMaxRekursiv(feld, mitte, rechts);
    } else {
        return findeMaxRekursiv(feld, links, mitte);
    }
}

public static int findeMaxRekursiv(int feld[]) {
    return findeMaxRekursiv(feld, 0, feld.length - 1);
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java)

Test

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class UnimodalFinderTest {

    private void testeMaxIterativ(int[] feld, int max) {
        assertEquals(max, UnimodalFinder.findeMaxIterativ(feld));
    }

    private void testeMaxRekursiv(int[] feld, int max) {
        assertEquals(max, UnimodalFinder.findeMaxRekursiv(feld));
    }

    private void testeMax(int[] feld, int max) {
        testeMaxIterativ(feld, max);
        testeMaxRekursiv(feld, max);
    }
}
```

```

    }

    @Test
    public void findeMax() {
        testeMax(new int[] { 1, 2, 3, 1 }, 3);
    }

    @Test
    public void findeMaxLaengeresFeld() {
        testeMax(new int[] { 1, 3, 4, 6, 7, 8, 9, 11, 6, 5, 4, 3, 2 }, 11);
    }

    @Test
    public void keinMaxAufsteigend() {
        testeMaxIterativ(new int[] { 1, 2, 3 }, UnimodalFinder.KEIN_MAX);
    }

    @Test
    public void keinMaxAbsteigend() {
        testeMaxIterativ(new int[] { 3, 2, 1 }, UnimodalFinder.KEIN_MAX);
    }

    @Test
    public void maxNegativeZahlen() {
        testeMax(new int[] { -2, -1, 3, 1 }, 3);
    }

    private void testeUnimodalität(int[] feld, boolean wahr) {
        assertEquals(wahr, UnimodalFinder.testeUnimodalität(feld));
    }

    @Test
    public void unimodalität() {
        testeUnimodalität(new int[] { 1, 2, 3, 1 }, true);
    }

    @Test
    public void unimodalitätFalsch() {
        testeUnimodalität(new int[] { 1, -2, 3, 1, 2 }, false);
        testeUnimodalität(new int[] { 1, 2, 3, 1, 2 }, false);
        testeUnimodalität(new int[] { 3, 2, 1 }, false);
    }
}

```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinderTest.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinderTest.java)

46115 / 2015 / Herbst / Thema 2 / Aufgabe 1

Fügen Sie die folgenden Werte in der gegebenen Reihenfolge in eine Streutabelle der Größe 8 (mit den Indizes 0 bis 7) und der Streufunktion $h(x) = x \bmod 8$ ein. Verwenden Sie die jeweils angegebene Hash-Variante bzw. Kollisionsauflösung: 15, 3, 9, 23, 1, 8, 17, 4

(a) Offenes Hashing

Zur Kollisionsauflösung wird Verkettung verwendet.

Beispiel

Für die beiden Werte 8 und 16 würde die Lösung wie folgt aussehen:

Bucket	0	1	2	...
Inhalt	8			
	16			

Lösungsvorschlag

$$h(15) = 15 \bmod 8 = 7$$

$$h(3) = 3 \pmod{8} = 3$$

$$h(9) = 9 \bmod 8 = 1$$

$$h(23) = 23 \bmod 8 = 7$$

$$h(1) = 1 \pmod{8} = 1$$

$$h(8) = 8 \mod 8 = 0$$

$$h(17) = 17 \bmod 8 = 1$$

$$h(4) = 4 \pmod 8 = 4$$

Bucket	0	1	2	3	4	5	6	7
Inhalt	8	9		3	4			15
		1						23
		17						

(b) Geschlossenes Hashing

Zur Kollisionsauflösung wird lineares Sondieren (nur hochzählend) mit Schrittweite +5 verwendet.

Treten beim Einfügen Kollisionen auf, dann notieren Sie die Anzahl der Versuche zum Ablegen des Wertes im Subskript (z. B. das Einfügen des Wertes 8 gelingt im 5. Versuch: 8_5).

Beispiel

Für die beiden Werte 8 und 16 würde die Lösung wie folgt aussehen:

Bucket	0	1	2	3	4	5	...
Inhalt	8					16 ₁	

$$h'(x) = x \bmod 8$$

$$h(x, i) = (h'(x) + i \cdot 5) \bmod 8$$

17 einfügen

Bucket	0	1	2	3	4	5	6	7
Inhalt	8	9		3	23 ₂		1 ₂	15

1. Versuch: $h(17, 0) = (h'(17) + 0 \cdot 5) \bmod 8 = (1 + 0) \bmod 8 = 1 \bmod 8 = 1$ (belegt von 9)
2. Versuch: $h(17, 1) = (h'(17) + 1 \cdot 5) \bmod 8 = (1 + 5) \bmod 8 = 6 \bmod 8 = 6$ (belegt von 1)
3. Versuch: $h(17, 2) = (h'(17) + 2 \cdot 5) \bmod 8 = (1 + 10) \bmod 8 = 11 \bmod 8 = 3$ (belegt von 3)
4. Versuch: $h(17, 3) = (h'(17) + 3 \cdot 5) \bmod 8 = (1 + 15) \bmod 8 = 16 \bmod 8 = 0$ (belegt von 8)
5. Versuch: $h(17, 4) = (h'(17) + 4 \cdot 5) \bmod 8 = (1 + 20) \bmod 8 = 21 \bmod 8 = 5$

4 einfügen

Bucket	0	1	2	3	4	5	6	7
Inhalt	8	9		3	23 ₂	17 ₅	1 ₂	15

1. Versuch: $h(4, 0) = (h'(4) + 0 \cdot 5) \bmod 8 = (4 + 0) \bmod 8 = 4$ (belegt von 23)
2. Versuch: $h(4, 1) = (h'(4) + 1 \cdot 5) \bmod 8 = (4 + 5) \bmod 8 = 1$ (belegt von 9)
3. Versuch: $h(4, 2) = (h'(4) + 2 \cdot 5) \bmod 8 = (4 + 10) \bmod 8 = 6$ (belegt von 1)
4. Versuch: $h(4, 3) = (h'(4) + 3 \cdot 5) \bmod 8 = (4 + 15) \bmod 8 = 3$ (belegt von 3)
5. Versuch: $h(4, 4) = (h'(4) + 4 \cdot 5) \bmod 8 = (4 + 20) \bmod 8 = 0$ (belegt von 8)
6. Versuch: $h(4, 5) = (h'(4) + 5 \cdot 5) \bmod 8 = (4 + 25) \bmod 8 = 5$ (belegt von 17)
7. Versuch: $h(4, 6) = (h'(4) + 6 \cdot 5) \bmod 8 = (4 + 30) \bmod 8 = 2$

Bucket	0	1	2	3	4	5	6	7
Inhalt	8	9	4 ₇	3	23 ₂	17 ₅	1 ₂	15

- (c) Welches Problem tritt auf, wenn zur Kollisionsauflösung lineares Sondieren mit Schrittweite 4 verwendet wird? Warum ist 5 eine bessere Wahl?

Lösungsvorschlag

Beim linearen Sondieren mit der Schrittweite 4 werden nur zwei verschiedene Buckets erreicht, beispielsweise: 1, 5, 1, 5, etc.

Beim linearen Sondieren mit der Schrittweite 5 werden nacheinander alle möglichen Buckets erreicht, beispielsweise: 1, 6, 3, 0, 5, 2, 7, 4.

46115 / 2015 / Herbst / Thema 2 / Aufgabe 4

Gegeben sei die folgende Methode `function`:

```
double function(int n) {  
    if (n == 1)  
        return 0.5 * n;  
    else  
        return 1.0 / (n * (n + 1)) + function(n - 1);  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/Induktion.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/Induktion.java)

Beweisen Sie folgenden Zusammenhang mittels vollständiger Induktion:

$$\forall n \geq 1: \text{function}(n) = f(n) \text{ mit } f(n) := 1 - \frac{1}{n+1}$$

Hinweis: Eventuelle Rechenungenauigkeiten, wie z. B. in Java, bei der Behandlung von Fließkommazahlen (z. B. `double`) sollen beim Beweis nicht berücksichtigt werden - Sie dürfen also annehmen, Fließkommazahlen würden mathematische Genauigkeit aufweisen.

Lösungsvorschlag

Induktionsanfang

— Beweise, dass $A(1)$ eine wahre Aussage ist. _____

$$f(1) := 1 - \frac{1}{1+1} = 1 - \frac{1}{2} = \frac{1}{2}$$

Induktionsvoraussetzung

— Die Aussage $A(k)$ ist wahr für ein beliebiges $k \in \mathbb{N}$. _____

$$f(n) := 1 - \frac{1}{n+1}$$

Induktionsschritt

— Beweise, dass wenn $A(n = k)$ wahr ist, auch $A(n = k + 1)$ wahr sein muss. _____

zu zeigen:

$$f(n+1) := 1 - \frac{1}{(n+1)+1} = f(n)$$

Vorarbeiten (Java in Mathe umwandeln):

$$\text{function}(n) = \frac{1}{n \cdot (n+1)} + f(n-1)$$

$$\begin{aligned}
f(n+1) &= \frac{1}{(n+1) \cdot ((n+1)+1)} + f((n+1)-1) && n+1 \text{ eingesetzt} \\
&= \frac{1}{(n+1) \cdot (n+2)} + f(n) && \text{vereinfacht} \\
&= \frac{1}{(n+1) \cdot (n+2)} + 1 - \frac{1}{n+1} && \text{für } f(n) \text{ Formel eingesetzt} \\
&= 1 + \frac{1}{(n+1) \cdot (n+2)} - \frac{1}{n+1} && 1. \text{ Bruch an 2. Stelle geschrieben} \\
&= 1 + \frac{1}{(n+1) \cdot (n+2)} - \frac{1 \cdot (n+2)}{(n+1) \cdot (n+2)} && 2. \text{ Bruch mit } (n+2) \text{ erweitert} \\
&= 1 + \frac{1 - (n+2)}{(n+1) \cdot (n+2)} && \text{die 2 Brüche subtrahiert} \\
&= 1 + \frac{1 - n - 2}{(n+1) \cdot (n+2)} && - + 2 = -2 \\
&= 1 + \frac{-1 - n}{(n+1) \cdot (n+2)} && 1 - 2 = -1 \\
&= 1 + \frac{-1 \cdot (1+n)}{(n+1) \cdot (n+2)} && (n+1) \text{ ausgeklammert} \\
&= 1 + \left(-1 \cdot \frac{(1+n)}{(n+1) \cdot (n+2)} \right) && \text{minus vor den Bruch bringen} \\
&= 1 - \frac{(1+n)}{(n+1) \cdot (n+2)} && \text{plus minus ist minus} \\
&= 1 - \frac{1}{n+2} && (n+1) \text{ gekürzt} \\
&= 1 - \frac{1}{(n+1)+1} && \text{Umformen zur Verdeutlichung}
\end{aligned}$$

46115 / 2016 / Frühjahr / Thema 1 / Aufgabe 5

Aufgabe 5

Beschreiben Sie, was es heißt, dass ein Problem (Sprache) NP-vollständig ist. Geben Sie ein NP-vollständiges Problem Ihrer Wahl an und erläutern Sie, dass (bzw.) warum es in NP liegt.

Lösungsvorschlag

NP-vollständig: NP-schwer und in NP

[Beliebiges Problem] liegt in NP, da der entsprechende Algorithmus dieses Problem nicht-deterministisch in Polynomialzeit löst → Algorithmus rät nichtdeterministisch Lösung, prüft sie in Polynomialzeit

46115 / 2016 / Frühjahr / Thema 1 / Aufgabe 8

(a) Sortieren Sie das Array mit den Integer Zahlen

25, 1, 12, 27, 30, 9, 33, 34, 18, 16

(i) mit *BubbleSort*

Lösungsvorschlag

25	1	12	27	30	9	33	34	18	16	Eingabe
25	1	12	27	30	9	33	34	18	16	Durchlauf Nr. 1
>25	1<	12	27	30	9	33	34	18	16	vertausche (i 0<>1)
1	>25	12<	27	30	9	33	34	18	16	vertausche (i 1<>2)
1	12	25	27	>30	9<	33	34	18	16	vertausche (i 4<>5)
1	12	25	27	9	30	33	>34	18<	16	vertausche (i 7<>8)
1	12	25	27	9	30	33	18	>34	16<	vertausche (i 8<>9)
1	12	25	27	9	30	33	18	16	34	Durchlauf Nr. 2
1	12	25	>27	9<	30	33	18	16	34	vertausche (i 3<>4)
1	12	25	9	27	30	>33	18<	16	34	vertausche (i 6<>7)
1	12	25	9	27	30	18	>33	16<	34	vertausche (i 7<>8)
1	12	25	9	27	30	18	16	33	34	Durchlauf Nr. 3
1	12	>25	9<	27	30	18	16	33	34	vertausche (i 2<>3)
1	12	9	25	27	>30	18<	16	33	34	vertausche (i 5<>6)
1	12	9	25	27	18	>30	16<	33	34	vertausche (i 6<>7)
1	12	9	25	27	18	16	30	33	34	Durchlauf Nr. 4
1	>12	9<	25	27	18	16	30	33	34	vertausche (i 1<>2)
1	9	12	25	>27	18<	16	30	33	34	vertausche (i 4<>5)
1	9	12	25	18	>27	16<	30	33	34	vertausche (i 5<>6)
1	9	12	25	18	16	27	30	33	34	Durchlauf Nr. 5
1	9	12	>25	18<	16	27	30	33	34	vertausche (i 3<>4)
1	9	12	18	>25	16<	27	30	33	34	vertausche (i 4<>5)
1	9	12	18	16	25	27	30	33	34	Durchlauf Nr. 6
1	9	12	>18	16<	25	27	30	33	34	vertausche (i 3<>4)
1	9	12	16	18	25	27	30	33	34	Durchlauf Nr. 7
1	9	12	16	18	25	27	30	33	34	Ausgabe

(ii) mit *Quicksort*, wenn als Pivotelement das jeweils erste Element gewählt wird.

Beschreiben Sie die Abläufe der Sortierverfahren

- (i) bei *BubbleSort* durch eine Angabe der Zwischenergebnisse nach jedem Durchlauf
- (ii) bei *Quicksort* durch die Angabe der Zwischenergebnisse nach den rekursiven Aufrufen.

(b) Welche Laufzeit (asymptotisch, in O-Notation) hat BubbleSort bei beliebig großen Arrays mit n Elementen. Begründen Sie Ihre Antwort.

46115 / 2016 / Frühjahr / Thema 2 / Aufgabe 2

Aufgabe 2

Ein Raumausstattungsunternehmen steht immer wieder vor dem Problem, feststellen zu müssen, ob ein gegebener rechteckiger Fußboden mit rechteckigen Teppichresten ohne Verschnitt ausgelegt werden kann. Alle Längen sind hier ganzzahlige Meterbeträge. Haben sie beispielsweise zwei Reste der Größen 3×5 und einen Rest der Größe 2×5 , so kann ein Fußboden der Größe 8×5 ausgelegt werden.

Das Unternehmen beauftragt eine Softwarefirma mit der Entwicklung eines Programms, welches diese Frage für beliebige Größen von Fußboden und Teppichresten entscheiden soll. Bei der Abnahme weist die Softwarefirma darauf hin, dass das Programm im wesentlichen alle Möglichkeiten durchprobiert und daher für große Eingaben schnell ineffizient wird. Auf die Frage, ob man das nicht besser machen könne, lautet die Antwort, dass das vorgelegte Problem NP-vollständig sei und daher nach derzeitigem Kenntnisstand der theoretischen Informatik nicht mehr zu erwarten sei.

Exkurs: SUBSET SUM

Das **Teilsommenproblem** (SUBSET SUM oder SSP) ist ein spezielles Rucksackproblem. Gegeben sei eine Menge von ganzen Zahlen $I = \{w_1, w_2, \dots, w_n\}$. Gesucht ist eine Untermenge, deren Elementsumme maximal, aber nicht größer als eine gegebene obere Schranke c ist.

- (a) Fixieren Sie ein geeignetes Format für Instanzen des Problems und geben Sie konkret an, wie die obige Beispielinstant in diesem Format aussieht.

Lösungsvorschlag

Problem L

1. Alternative $I = \{x_1, y_1, \dots, x_n, y_n, c_x, c_y\}$

2. Alternative $I = \{w_1, \dots, w_n, c\}$

- (b) Begründen Sie, dass das Problem in NP liegt.

Lösungsvorschlag

Es existiert ein nichtdeterministischer Algorithmus der das Problem in Polynomialzeit entscheidet:

- nichtdeterministisch Untermenge raten ($\mathcal{O}(n)$)
- Prüfe: ($\mathcal{O}(n)$)

1. Alternative Elementsumme der Produkte (x_i, y_i) aus Untermenge $= c$

2. Alternative Elementsumme der Untermenge $= c$

- (c) Begründen Sie, dass das Problem NP-schwer ist durch Reduktion vom NP-vollständigen Problem SUBSET-SUM.

$$\text{SUBSET SUM} \preceq_p L$$

1. Alternative Die Funktion f ersetzt jedes w_i durch $w_i, 1$ und c durch $c, 1$ und startet TM für L

Berechenbarkeit: Hinzufügen von 1 für jedes Element, offensichtlich in Polynomialzeit

Korrektheit: $w \in \text{SUBSET SUM} \Leftrightarrow f(w) \in L$, offensichtlich, selbes Problem mit lediglich anders notierter Eingabe

2. Alternative Die Funktion f startet TM für L

Berechenbarkeit: Identität, offensichtlich in Polynomialzeit

Korrektheit: $w \in \text{SUBSET SUM} \Leftrightarrow f(w) \in L$ offensichtlich, selbes Problem

46115 / 2016 / Frühjahr / Thema 2 / Aufgabe 3

Sie sollen mithilfe von Falltests eine neue Serie von Smartphones auf Bruchssicherheit testen.

Dazu wird eine Leiter mit n Sprossen verwendet; die höchste Sprosse, von der ein Smartphone heruntergeworfen werden kann ohne zu zerbrechen, heie „*höchste sichere Sprosse*“. Das Ziel ist, die höchste sichere Sprosse zu ermitteln. Man kann davon ausgehen, dass die höchste sichere Sprosse nicht von der Art des Wurfs abhängt und dass alle verwendeten Smartphones sich gleich verhalten. Eine Möglichkeit, die höchste sichere Sprosse zu ermitteln, besteht darin, ein Gerät erst von Sprosse 1, dann von Sprosse 2, etc. abzuwerfen, bis es schließlich beim Wurf von Sprosse k beschädigt wird (oder Sie oben angelangt sind). Sprosse $k - 1$ (bzw. n) ist dann die höchste sichere Sprosse. Bei diesem Verfahren wird maximal ein Smartphone zerstört, aber der Zeitaufwand ist ungünstig.

(a) Bestimmen Sie die Zahl der Würfe bei diesem Verfahren im schlechtesten Fall.

Lösungsvorschlag

Die Zahl der Würfe im schlechtesten Fall ist $\mathcal{O}(k)$, wobei k die Anzahl der Sprossen ist. Geht das Smartphone erst bei der höchsten Sprosse kaputt, muss es k mal heruntergeworfen werden. Die Komplexität entspricht der der linearen Suche.

(b) Geben Sie nun ein Verfahren zur Ermittlung der höchsten sicheren Sprosse an, welches nur $\mathcal{O}(\log n)$ Würfe benötigt, dafür aber möglicherweise mehr Smartphones verbraucht.

Lösungsvorschlag

Man startet bei Sprosse $\frac{n}{2}$. Wenn das Smartphone kaputt geht, macht man

weiter mit der Sprosse in der Mitte der unteren Hälfte, ansonsten mit der Sprosse in der Mitte der oberen Hälfte. Das Ganze rekursiv.

- (c) Es gibt eine Strategie zur Ermittlung der höchsten sicheren Sprosse mit $\mathcal{O}(\sqrt{n})$ Würfeln, bei dessen Anwendung höchstens zwei Smartphones kaputtgehen. Finden Sie diese Strategie und begründen Sie Ihre Funktionsweise und Wurfzahl.

Tipp: der erste Testwurf erfolgt von Sprosse $\lceil \sqrt{n} \rceil$.

Exkurs: Interpolationssuche

Die Interpolationssuche, auch Intervallsuche genannt, ist ein von der binären Suche abgeleitetes Suchverfahren, das auf Listen und Feldern zum Einsatz kommt.

Während der Algorithmus der binären Suche stets das mittlere Element des Suchraums überprüft, versucht der Algorithmus der Interpolationssuche im Suchraum einen günstigeren Teilungspunkt als die Mitte zu erraten. Die Arbeitsweise ist mit der eines Menschen vergleichbar, der ein Wort in einem Wörterbuch sucht: Die Suche nach Zylinder wird üblicherweise am Ende des Wörterbuches begonnen, während die Suche nach Aal im vorderen Bereich begonnen werden dürfte.^a

^ahttps://de.wikipedia.org/wiki/Quadratische_Binärsuche

Exkurs: Quadratische Binärsuche

Quadratische Binärsuche ist ein Suchalgorithmus ähnlich der Binärsuche oder Interpolationssuche. Es versucht durch Reduzierung des Intervalls in jedem Rekursionsschritt die Nachteile der Interpolationssuche zu vermeiden.

Nach dem Muster der Interpolationssuche wird zunächst in jedem rekursiven Schritt die vermutete Position k interpoliert. Anschließend wird – um die Nachteile der Interpolationssuche zu vermeiden – das Intervall der Länge \sqrt{n} gesucht, in dem sich der gesuchte Wert befindet. Auf dieses Intervall wird der nächste rekursive Aufruf der Suche angewendet.

Auf diese Weise verkleinert sich der Suchraum bei gegebener Liste der Länge n bei jedem rekursiven Schritt auf eine Liste der Länge n/\sqrt{n} .^a

^ahttps://de.wikipedia.org/wiki/Quadratische_Binärsuche

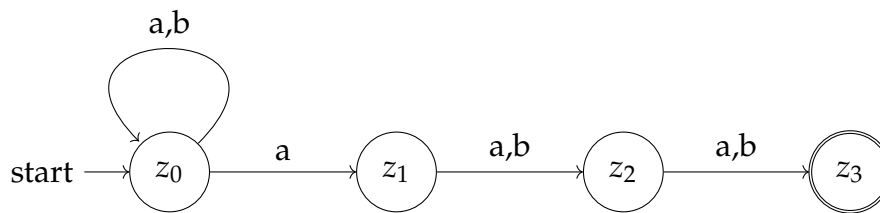
Lösungsvorschlag

Das Vorgehen ist folgendermaßen: Man beginnt auf Stufe 0 und falls das Handy nicht kaputt geht, addiert man jeweils Wurzel n . Falls das Handy kaputt geht, geht man linear in Einerschritten das Intervall von der unteren Grenze (ö von der Stufe vor der letzten Addition) bis zur Kaputtstufe ab.^a

^a<http://www.inf.fu-berlin.de/lehre/WS06/HA/skript/vorlesung6.pdf>

46115 / 2016 / Herbst / Thema 1 / Aufgabe 1

- (a) Konstruieren Sie aus dem NEA mit der Potenzmengenkonstruktion einen (deterministischen) EA, der dieselbe Sprache akzeptiert.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aiqxdazuw

Lösungsvorschlag

Name	Zustandsmenge	Eingabe a	Eingabe b
Z_0	$Z_0\{z_0\}$	$Z_1\{z_0, z_1\}$	$Z_0\{z_0\}$
Z_1	$Z_1\{z_0, z_1\}$	$Z_2\{z_0, z_1, z_2\}$	$Z_3\{z_0, z_2\}$
Z_2	$Z_2\{z_0, z_1, z_2\}$	$Z_4\{z_0, z_1, z_2, z_3\}$	$Z_6\{z_0, z_2, z_3\}$
Z_3	$Z_3\{z_0, z_2\}$	$Z_7\{z_0, z_1, z_3\}$	$Z_5\{z_0, z_3\}$
Z_4	$Z_4\{z_0, z_1, z_2, z_3\}$	$Z_4\{z_0, z_1, z_2, z_3\}$	$Z_6\{z_0, z_2, z_3\}$
Z_5	$Z_5\{z_0, z_3\}$	$Z_1\{z_0, z_1\}$	$Z_0\{z_0\}$
Z_6	$Z_6\{z_0, z_2, z_3\}$	$Z_7\{z_0, z_1, z_3\}$	$Z_5\{z_0, z_3\}$
Z_7	$Z_7\{z_0, z_1, z_3\}$	$Z_2\{z_0, z_1, z_2\}$	$Z_3\{z_0, z_2\}$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aiqnk06c9

(b) Beschreiben Sie möglichst einfach, welche Sprachen von den folgenden regulären Ausdrücken beschrieben werden:

(i) $(a|b)^*a$

Sprache mit dem Alphabet $\Sigma = \{a, b\}$: Alle Wörter der Sprache enden auf a .

(ii) $(a|b)^*a(a|b)^*a(a|b)^*$

Sprache mit dem Alphabet $\Sigma = \{a, b\}$: Alle Wörter der Sprache enthalten mindestens 2 a 's.

(iii) $(a|b)^*a(bb)^*a(a|b)^*$

Sprache mit dem Alphabet $\Sigma = \{a, b\}$: Alle Wörter der Sprache enthalten mindestens 2 a 's, die von einer geradzahligem Anzahl von b 's getrennt sind.

46115 / 2016 / Herbst / Thema 1 / Aufgabe 4

Betrachten Sie die beiden folgenden Probleme:

VERTEX COVER

Gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \{1, 2, 3, \dots\}$

Frage: Gibt es eine Menge $C \subseteq V$ mit $|C| \leq k$, so dass für jede Kante (u, v) aus E mindestens einer der Knoten u und v in C ist?

VERTEX COVER 3

Gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \{3, 4, 5, \dots\}$.

Frage: Gibt es eine Menge $C \subseteq V$ mit $|C| \leq k$, so dass für jede Kante (u, v) aus E mindestens einer der Knoten u und v in C ist?

Geben Sie eine polynomielle Reduktion von VERTEX COVER auf VERTEX COVER 3 an und begründe anschließend, dass die Reduktion korrekt ist.

Exkurs: VERTEX COVER

Das **Knotenüberdeckungsproblem** (VERTEX COVER) fragt, ob zu einem gegebenen einfachen Graphen und einer natürlichen Zahl k eine Knotenüberdeckung der Größe von höchstens k existiert.

Das heißt, ob es eine aus maximal k Knoten bestehende Teilmenge U der Knotenmenge gibt, so

dass jede Kante des Graphen mit mindestens einem Knoten aus U verbunden ist.

Lösungsvorschlag

$\text{VERTEX COVER} \preceq_p \text{VERTEX COVER } 3$

f fügt vier neue Knoten hinzu, von denen jeweils ein Paar verbunden ist. Außerdem erhöht f k um 2.

Total: Jeder Graph kann durch f so verändert werden.

Korrektheit: Wenn VERTEX COVER für k in G existiert, dann existiert auch VERTEX COVER mit $k + 2$ Knoten in G' , da für den eingefügten Teilgraphen ein VERTEX COVER mit $k = 2$ existiert.

In Polynomialzeit berechenbar: Für die Adjazenzmatrix des Graphen müssen lediglich vier neue Spalten und Zeilen eingefügt werden und $k + 2$ berechnet werden.

46115 / 2016 / Herbst / Thema 2 / Aufgabe 2

Geben Sie jeweils die kleinste, gerade noch passende Laufzeitkomplexität folgender Java-Methoden im O-Kalkül (Landau-Notation) in Abhängigkeit von n und ggf. der Länge der Arrays an.

(a)

```
int matrixSumme(int n, int[][] feld) {
    int sum = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            sum += feld[i][j];
        }
    }
    return sum;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2016/herbst/Komplexitaet.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2016/herbst/Komplexitaet.java)

Lösungsvorschlag

Die Laufzeit liegt in $\mathcal{O}(n^2)$.

Begründung (nicht verlangt): Die äußere Schleife wird n -mal durchlaufen. Die innere Schleife wird dann jeweils wieder n -mal durchlaufen. Die Größe des Arrays spielt hier übrigens keine Rolle, da die Schleifen ohnehin immer nur bis zum Wert n ausgeführt werden.

(b)

```

int find(int key, int[] [] keys) {
    int a = 0, o = keys.length;
    while (o - a > 1) {
        int m = (a + o) / 2;
        if (keys[m][0] > key)
            o = m;
        else
            a = m;
    }
    return a;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2016/herbst/Komplexitaet.java](https://github.com/org/bschlangaul/examen/examen_46115/jahr_2016/herbst/Komplexitaet.java)

Lösungsvorschlag

Die Laufzeit liegt in $\mathcal{O}(\log(\text{keys.length}))$. Dabei ist `keys.length` die Größe des Arrays bezüglich seiner ersten Dimension.

Begründung (nicht verlangt): Der Grund für diese Laufzeit ist derselbe wie bei der binären Suche. Die Größe des Arrays bezüglich seiner zweiten Dimension spielt hier übrigens keine Rolle, da diese Dimension hier ja nur einen einzigen festen Wert annimmt.

46115 / 2016 / Herbst / Thema 2 / Aufgabe 4

Mittels Dynamischer Programmierung (auch Memoization genannt) kann man insbesondere rekursive Lösungen auf Kosten des Speicherbedarf beschleunigen, indem man Zwischenergebnisse „abspeichert“ und bei (wiederkehrendem) Bedarf „abruft“, ohne sie erneut berechnen zu müssen.

Gegeben sei folgende geschachtelt-rekursive Funktion für $n, m \geq 0$:

$$a(n, m) = \begin{cases} n + \lfloor \frac{n}{2} \rfloor & \text{falls } m = 0 \\ a(1, m - 1), & \text{falls } n = 0 \wedge m \neq 0 \\ a(n + \lfloor \sqrt{a(n - 1, m)} \rfloor, m - 1), & \text{sonst} \end{cases}$$

- (a) Implementieren Sie die obige Funktion `a(n,m)` zunächst ohne weitere Optimierungen als Prozedur/Methode in einer Programmiersprache Ihrer Wahl.

Lösungsvorschlag

```

public static long a(int n, int m) {
    if (m == 0) {
        return n + (n / 2);
    } else if (n == 0 && m != 0) {
        return a(1, m - 1);
    } else {
        return a(n + ((int) Math.sqrt(a(n - 1, m))), m - 1);
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2016/herbst/DynamischeProgrammierung.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2016/herbst/DynamischeProgrammierung.java)

- (b) Geben Sie nun eine DP-Implementierung der Funktion $a(n, m)$ an, die $a(n, m)$ für $0 \leq n \leq 100000$ und $0 \leq m \leq 25$ höchstens einmal gemäß obiger rekursiver Definition berechnet. Beachten Sie, dass Ihre Prozedur trotzdem auch weiterhin mit $n > 100000$ und $m > 25$ aufgerufen werden können soll.

Lösungsvorschlag

```
static long[][] tmp = new long[100001][26];

public static long aDp(int n, int m) {
    if (n <= 100000 && m <= 25 && tmp[n][m] != -1) {
        return tmp[n][m];
    } else {
        long merker;
        if (m == 0) {
            merker = n + (n / 2);
        } else if (n == 0 && m != 0) {
            merker = aDp(1, m - 1);
        } else {
            merker = aDp(n + ((int) Math.sqrt(aDp(n - 1, m))), m - 1);
        }
        if (n <= 100000 && m <= 25) {
            tmp[n][m] = merker;
        }
        return merker;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2016/herbst/DynamischeProgrammierung.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2016/herbst/DynamischeProgrammierung.java)

Lösungsvorschlag

Kompletter Code

```
public class DynamischeProgrammierung {
    public static long a(int n, int m) {
        if (m == 0) {
            return n + (n / 2);
        } else if (n == 0 && m != 0) {
            return a(1, m - 1);
        } else {
            return a(n + ((int) Math.sqrt(a(n - 1, m))), m - 1);
        }
    }

    static long[][] tmp = new long[100001][26];

    public static long aDp(int n, int m) {
        if (n <= 100000 && m <= 25 && tmp[n][m] != -1) {
            return tmp[n][m];
        } else {
            long merker;

```



```

    if (m == 0) {
        merker = n + (n / 2);
    } else if (n == 0 && m != 0) {
        merker = aDp(1, m - 1);
    } else {
        merker = aDp(n + ((int) Math.sqrt(aDp(n - 1, m))), m - 1);
    }
    if (n <= 100000 && m <= 25) {
        tmp[n][m] = merker;
    }
    return merker;
}
}

public static void main(String[] args) {
    for (int i = 0; i < 100001; i++) {
        for (int j = 0; j < 26; j++) {
            tmp[i][j] = -1;
        }
    }
    System.out.println("schnell mit DP: " + aDp(7,7));
    System.out.println("langsam ohne DP: " + a(7,7));
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2016/herbst/DynamischeProgrammierung.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2016/herbst/DynamischeProgrammierung.java)

46115 / 2017 / Herbst / Thema 2 / Aufgabe 3

Die Methode `pKR` berechnet die n -te Primzahl ($n \geq 1$) kaskadenartig rekursiv und äußerst ineffizient:

```

static long pKR(int n) {
    long p = 2;
    if (n >= 2) {
        p = pKR(n - 1); // beginne die Suche bei der vorhergehenden Primzahl
        int i = 0;
        do {
            p++; // pruefe, ob die jeweils naechste Zahl prim ist, d.h. ...
            for (i = 1; i < n && p % pKR(i) != 0; i++) {
                // pruefe, ob unter den kleineren Primzahlen ein Teiler ist
            } while (i != n); // ... bis nur noch 1 und p Teiler von p sind
        }
        return p;
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2017/herbst/PrimzahlDP.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2017/herbst/PrimzahlDP.java)

Überführen Sie `pKR` mittels *dynamischer Programmierung* (hier also *Memoization*) und mit möglichst *wenigen Änderungen* so in die linear rekursive Methode `pLR`, dass `pLR(n, new long[n + 1])` ebenfalls die n -te Primzahl ermittelt:

```

private long pLR(int n, long[] ps) {
    ps[1] = 2;
    // ...
}

```


Exkurs: Kaskadenartig rekursiv

Kaskadenförmige Rekursion bezeichnet den Fall, in dem mehrere rekursive Aufrufe nebeneinander stehen.

Exkurs: Linear rekursiv

Die häufigste Rekursionsform ist die lineare Rekursion, bei der in jedem Fall der rekursiven Definition höchstens ein rekursiver Aufruf vorkommen darf.

```
static long pLR(int n, long[] ps) {
    ps[1] = 2;
    long p = 2;
    if (ps[n] != 0) // Fall die Primzahl bereits gefunden / berechnet wurde,
        return ps[n]; // gib die berechnete Primzahl zurück.
    if (n >= 2) {
        // der einzige rekursive Aufruf steht hier, damit die Methode linear
        // ↪ rekursiv
        // ist.
        p = pLR(n - 1, ps);
        int i = 0;
        do {
            p++;
            // Hier wird auf das gespeicherte Feld zurückgegriffen.
            for (i = 1; i < n && p % ps[i] != 0; i++) {
            }
        } while (i != n);
    }
    ps[n] = p; // Die gesuchte Primzahl im Feld speichern.
    return p;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2017/herbst/PrimzahlDP.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2017/herbst/PrimzahlDP.java)

Der komplette Quellcode

```
/**
 * Berechne die n-te Primzahl.
 *
 * Eine Primzahl ist eine natürliche Zahl, die größer als 1 und ausschließlich
 * durch sich selbst und durch 1 teilbar ist.
 *
 * <ul>
 * <li>1. Primzahl: 2
 * <li>2. Primzahl: 3
 * <li>3. Primzahl: 5
 * <li>4. Primzahl: 7
 * <li>5. Primzahl: 11
```

```
* <li>6. Primzahl: 13
* <li>7. Primzahl: 17
* <li>8. Primzahl: 19
* <li>9. Primzahl: 23
* <li>10. Primzahl: 29
* </ul>
*/
public class PrimzahlDP {

    /**
     * Die Methode pKR berechnet die n-te Primzahl ({@code n >= 1}) Kaskadenartig
     * → Rekursiv.
     *
     * @param n Die Nummer (n-te) der gesuchten Primzahl. Die Primzahl 2 ist die
     *           erste Primzahl. Die Primzahl 3 ist die zweite Primzahl etc.
     *
     * @return Die gesuchte n-te Primzahl.
     */
    static long pKR(int n) {
        long p = 2;
        if (n >= 2) {
            p = pKR(n - 1); // beginne die Suche bei der vorhergehenden Primzahl
            int i = 0;
            do {
                p++; // pruefe, ob die jeweils naechste Zahl prim ist, d.h. ...
                for (i = 1; i < n && p % pKR(i) != 0; i++) {
                } // pruefe, ob unter den kleineren Primzahlen ein Teiler ist
            } while (i != n); // ... bis nur noch 1 und p Teiler von p sind
        }
        return p;
    }

    /**
     * Die Methode pLR berechnet die n-te Primzahl ({@code n >= 1}) Linear Rekursiv.
     *
     * @param n Die Nummer (n-te) der gesuchten Primzahl. Die Primzahl 2 ist die
     *           erste Primzahl. Die Primzahl 3 ist die zweite Primzahl etc.
     * @param ps Primzahl Speicher. Muss mit n + 1 initialisiert werden.
     *
     * @return Die gesuchte n-te Primzahl.
     */
    static long pLR(int n, long[] ps) {
        ps[1] = 2;
        long p = 2;
        if (ps[n] != 0) // Fall die Primzahl bereits gefunden / berechnet wurde,
            return ps[n]; // gib die berechnete Primzahl zurück.
        if (n >= 2) {
            // der einzige rekursive Aufruf steht hier, damit die Methode linear
            // → rekursiv
            // ist.
            p = pLR(n - 1, ps);
            int i = 0;
            do {
                p++;
```

```

    // Hier wird auf das gespeicherte Feld zurückgegriffen.
    for (i = 1; i < n && p % ps[i] != 0; i++) {
    }
    } while (i != n);
}
ps[n] = p; // Die gesuchte Primzahl im Feld speichern.
return p;
}

static void debug(int n) {

    ↪ System.out.println(String.format("%d. Primzahl: %d (kaskadenartig rekursiv berechnet)",
    ↪ n, pKR(n)));

    ↪ System.out.println(String.format("%d. Primzahl: %d (linear rekursiv berechnet)",
    ↪ n, pLR(n, new long[n + 1])));
}

public static void main(String[] args) {
    System.out.println(pKR(10));
    System.out.println(pLR(10, new long[11]));

    for (int i = 1; i <= 10; i++) {
        debug(i);
    }
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2017/herbst/PrimzahlDP.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_46115/jahr_2017/herbst/PrimzahlDP.java)

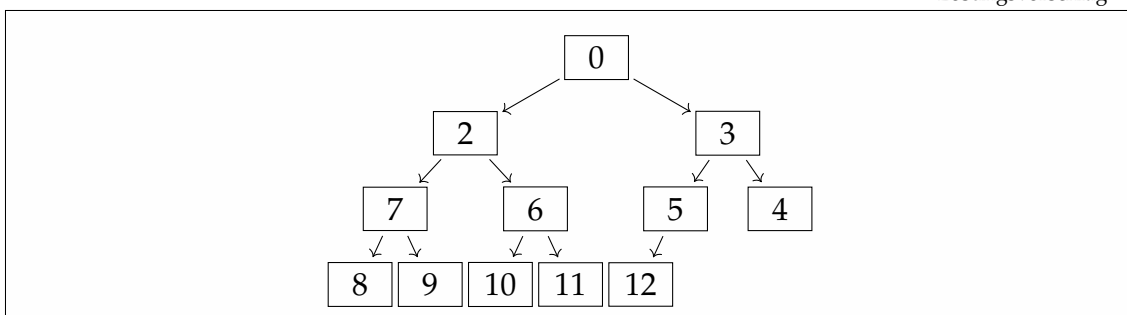
46115 / 2017 / Herbst / Thema 2 / Aufgabe 6

Gegeben sei folgende Feld-Einbettung (Array-Darstellung) einer Min-Halde:

0	1	2	3	4	5	6	7	8	9	10	11
0	2	3	7	6	5	4	8	9	10	11	12

(a) Stellen Sie die Halde graphisch als (links-vollständigen) Baum dar.

Lösungsvorschlag



(b) Entfernen Sie das kleinste Element (die Wurzel 0) aus der obigen initialen Halde, stellen Sie die Haldeneigenschaft wieder her und geben Sie nur das Endergebnis in Felddarstellung an.

Lösungsvorschlag

Insertionsort
Schreibtschlauf (Sortierung)
Implementierung in Java
Implementierung in Java

0	1	2	3	4	5	6	7	8	9	10
2	6	3	7	10	5	4	8	9	12	11

- (c) Fügen Sie nun den Wert 1 in die obige initiale Halde ein, stellen Sie die Haldeneigenschaft wieder her und geben Sie nur das Endergebnis in Felddarstellung an.

Lösungsvorschlag

0	1	2	3	4	5	6	7	8	9	10	11	12
0	2	1	7	6	3	4	8	9	10	11	12	5

Aufgabe 7

- (a) Führen Sie „Sortieren durch Einfügen“ lexikographisch aufsteigend und *in-situ* (*in-place*) so in einem Schreibtischlauf auf folgendem Feld (Array) aus, dass gleiche Elemente ihre relative Abfolge jederzeit beibehalten (also dass z. B. A_1 stets vor A_2 im Feld steht). Jede Zeile stellt den Zustand des Feldes dar, nachdem das jeweils nächste Element in die Endposition verschoben wurde. Der bereits sortierte Teilbereich steht vor |||. Gleiche Elemente tragen zwecks Unterscheidung ihre „Objektidentität“ als Index (z. B. `"A1".equals("A2")` aber `"A1" != "A2"`)

L	A_1	B_1	F	A_2	B_2
---	-------	-------	---	-------	-------

Lösungsvorschlag

L	A_1	B_1	F	A_2	B_2
A_1	L	B_1	F	A_2	B_2
A_1	B_1	L	F	A_2	B_2
A_1	B_1	F	L	A_2	B_2
A_1	A_2	B_1	F	L	B_2
A_1	A_2	B_1	B_2	F	L

- (b) Ergänzen Sie die folgende Methode so, dass sie die Zeichenketten im Feld `a` lexikographisch aufsteigend durch Einfügen sortiert. Sie muss zum vorangehenden Ablauf passen, ösie muss *iterativ* sowie *in-situ* (*in-place*) arbeiten und die relative Reihenfolge gleicher Elemente jederzeit beibehalten. Sie dürfen davon ausgehen, dass kein Eintrag im Feld null ist.

```
void sortierenDurchEinfuegen(String[] a) {
    // Hilfsvariable:
    String tmp;
}
```

```

static void sortierenDurchEinfuegen(String[] a) {
    // Hilfsvariable:
    String tmp;
    for (int i = 1; i < a.length; i++) {
        tmp = a[i];
        int j = i;
        while (j > 0 && a[j - 1].compareTo(tmp) >= 1) {
            a[j] = a[j - 1];
            j = j - 1;
        }
        a[j] = tmp;
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2017/herbst/InsertionSort.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_46115/jahr_2017/herbst/InsertionSort.java)

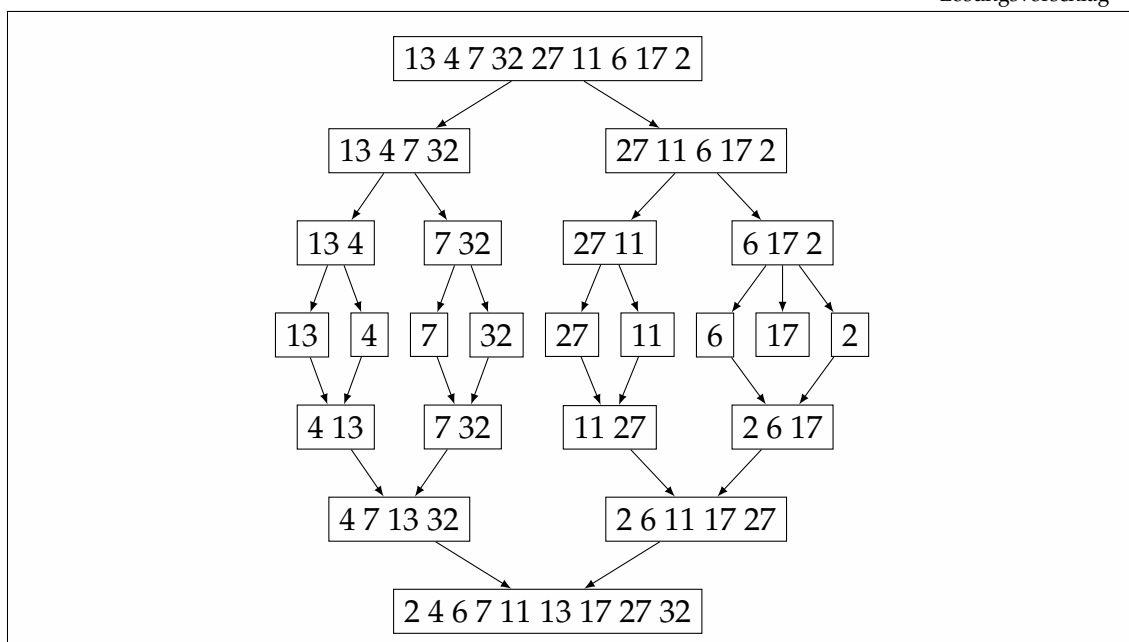
Aufgabe zum Mergesort

(a) Gegeben ist das folgende Array von Zahlen:

[13, 4, 7, 32, 27, 11, 6, 17, 2]

Sortieren Sie das Array mittels Mergesort aufsteigend von links nach rechts. Das Aufteilen einer Liste soll in der Mitte erfolgen und, falls notwendig, die zweite Liste ein Element länger sein als die erste. Listen der Länge zwei dürfen durch direkten Vergleich sortiert werden. Geben Sie die Eingabe und das Ergebnis jedes (rekursiven) Aufrufs an. Geben Sie abschließend die sortierte Liste an.

Lösungsvorschlag



(b) Beantworten Sie folgende Fragen jeweils ohne Begründung oder Beweis.

- (i) Welche Worst-Case-Laufzeit (\mathcal{O} -Notation) hat Mergesort für n Elemente?

Lösungsvorschlag

$$\mathcal{O}(n \cdot \log(n)) \text{ im Best-, Average- und Worst-Case}$$

- (ii) Welche Laufzeit hat Mergesort für n Elemente im Best-Case?

Lösungsvorschlag

$$\mathcal{O}(n \cdot \log(n)) \text{ im Best-, Average- und Worst-Case}$$

- (iii) Kann basierend auf paarweisen Vergleichen von Werten schneller (Laufzeitkomplexität) als Mergesort sortiert werden?

Lösungsvorschlag

Nein. Es lässt sich beweisen, dass ein vergleichsbasiertes Sortierverfahren nicht schneller als $\Omega(n \cdot \log(n))$ sein kann.^a

^a<https://de.wikipedia.org/wiki/Sortierverfahren>

46115 / 2018 / Frühjahr / Thema 1 / Aufgabe 8

Berechnen Sie mithilfe des Algorithmus von Prim ausgehend vom Knoten s einen minimalen Spannbaum des ungerichteten Graphen G , der durch folgende Adjazenzmatrix gegeben ist:

$$\begin{array}{c}
 \begin{array}{cccccccc}
 & s & a & b & c & d & e & f & g & h \\
 \begin{array}{c} s \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{array} & \begin{pmatrix} * & - & 3 & - & - & 7 & - & - & - \\ - & * & - & 0 & 8 & - & 11 & - & - \\ 3 & - & * & - & 5 & - & 10 & - & - \\ - & 0 & - & * & - & - & 1 & - & - \\ - & 8 & 5 & - & * & 2 & 3 & - & 6 \\ 7 & - & - & - & 2 & * & - & - & 11 \\ - & 11 & 10 & 1 & 3 & - & * & 7 & - \\ - & - & - & - & - & - & 7 & * & 4 \\ - & - & - & - & 6 & 11 & - & 4 & * \end{pmatrix}
 \end{array}
 \end{array}$$

- (a) Erstellen Sie dazu eine Tabelle mit zwei Spalten und stellen Sie jeden einzelnen Schritt des Verfahrens in einer eigenen Zeile dar. Geben Sie in der ersten Spalte denjenigen Knoten v , der vom Algorithmus als nächstes in den Ergebnisbaum aufgenommen wird (dieser sog. schwarze Knoten ist damit fertiggestellt) als Tripel (v, p, δ) mit v als Knotenname, p als aktueller Vorgängerknoten und δ als aktuelle Distanz von v zu p an. Führen Sie in der zweiten Spalte alle anderen vom ak-

tuellen Spannbaum direkt erreichbaren Knoten v (sog. graue Randknoten) ebenfalls als Tripel (v, p, δ) auf. Zeichnen Sie anschließend den entstandenen Spannbaum und geben Sie sein Gewicht an.

Lösungsvorschlag

Der Graph muss nicht gezeichnet werden. Der Algorithmus kann auch nur mit der Adjazenzmatrix durchgeführt werden. Möglicherweise geht das Lösen der Aufgabe schneller mit der Matrix von der Hand.

Kompletter Graph

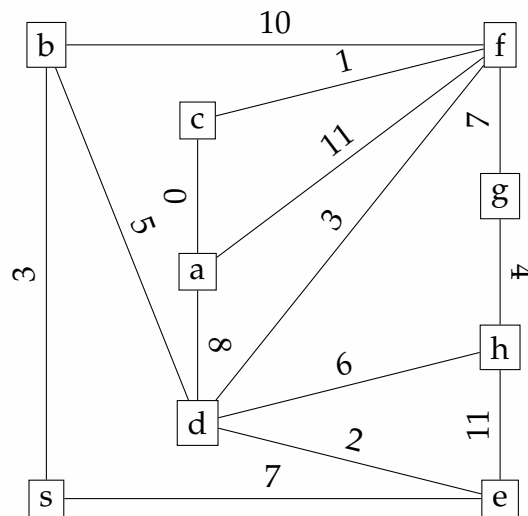
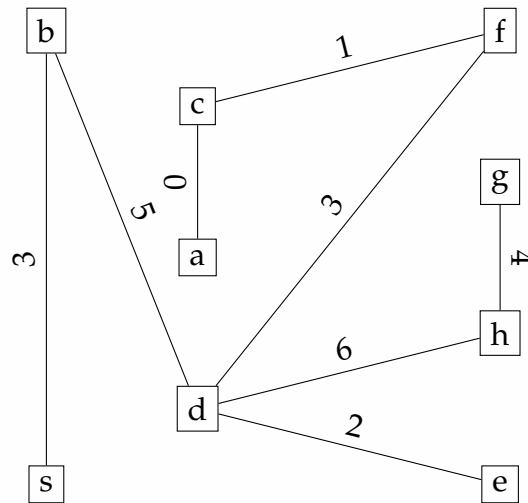


Tabelle schwarz-graue-Knoten

schwarze	graue
(s, null, -)	(b, s, 3); (e, s, 7);
(b, s, 3)	(d, b, 5); (e, s, 7); (f, b, 10);
(d, b, 5)	(a, d, 8); (e, d, 2); (f, d, 3); (h, d, 6);
(e, d, 2)	(a, d, 8); (f, d, 3); (h, d, 6);
(f, d, 3)	(a, d, 8); (c, f, 1); (g, f, 7); (h, d, 6);
(c, f, 1)	(a, c, 0); (g, f, 7); (h, d, 6);
(a, c, 0)	(g, f, 7); (h, d, 6);
(h, d, 6)	(g, h, 4);
(g, h, 4)	

Gewicht des minimalen Spannbaums: 24

Minimaler Spannbaum

Algorithmus von Kruskal
Minimaler Spannbaum
Algorithmus von Prim

- (b) Welche Worst-Case-Laufzeitkomplexität hat der Algorithmus von Prim, wenn die grauen Knoten in einem Heap nach Distanz verwaltet werden? Sei dabei n die Anzahl an Knoten und m die Anzahl an Kanten des Graphen. Eine Begründung ist nicht erforderlich.

Lösungsvorschlag

$$\mathcal{O}(m + n \log n)$$

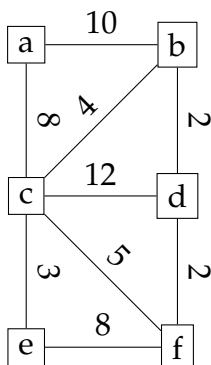
- (c) Beschreiben Sie kurz die Idee des alternativen Ansatzes zur Berechnung eines minimalen Spannbaumes von Kruskal.

Lösungsvorschlag

Kruskal wählt nicht die kürzeste an einen Teilgraphen anschließende Kante, sondern global die kürzeste verbliebene aller Kanten, die keinen Zyklus bildet, ohne dass diese mit dem Teilgraph verbunden sein muss.

46115 / 2018 / Frühjahr / Thema 2 / Aufgabe 4

Sei G der folgende Graph.



- (a) Der Algorithmus von Prim ist ein Algorithmus zur Bestimmung des minimalen Spannbaums in einem Graphen. Geben Sie einen anderen Algorithmus zur Bestimmung des minimalen Spannbaums an.

Lösungsvorschlag

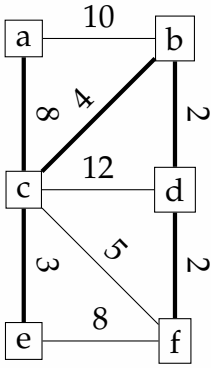
Zum Beispiel der Algorithmus von Kruskal

- (b) Führen Sie den Algorithmus von Prim schrittweise auf G aus. Ausgangsknoten soll der Knoten a sein. Ihre Tabelle sollte wie folgt beginnen:

a	b	c	d	e	f	Warteschlange
---	---	---	---	---	---	---------------

Die Einträge der Tabelle geben an, wie weit der angegebene Knoten vom aktuellen Baum entfernt ist.

Lösungsvorschlag

						
a	b	c	d	e	f	Warteschlange
0	∞	∞	∞	∞	∞	a
0	10	8	∞	∞	∞	c, b
0	4	0	12	3	5	e, b, f, d
0	4	0	12	0	5	b, f, d
0	0	0	2	0	5	d, f
0	0	0	0	0	2	f
0	0	0	0	0	0	

- (c) Erklären Sie, warum der Kürzeste-Wege-Baum (also das gezeichnete Ergebnis des Dijkstra-Algorithmus) und der minimale Spannbaum nicht notwendigerweise identisch sind.

Lösungsvorschlag

Die Wahl der nächsten Kante erfolgt nach völlig verschiedenen Kriterien:

- Beim Kürzeste-Wege-Baum orientiert sie sich an der Entfernung der einzelnen Knoten vom Startknoten.

- Beim Spannbaum orientiert sie sich an der Entfernung der einzelnen Knoten vom bereits erschlossenen Teil des Spannbaums.

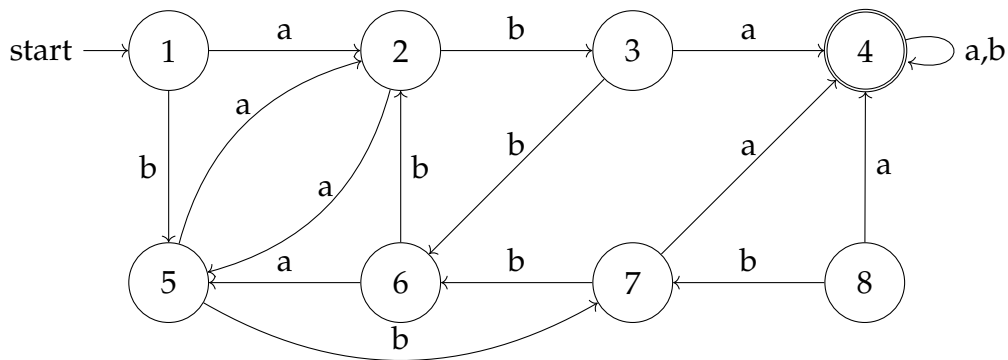
46115 / 2019 / Herbst / Thema 1 / Aufgabe 1

- (a) Geben Sie einen regulären Ausdruck für die Sprache über dem Alphabet $\{a, b\}$ an, die genau alle Wörter enthält, die eine gerade Anzahl a 's haben.

Lösungsvorschlag

$$b^*(ab^*ab^*)^*$$

- (b) Sei A der folgende DEA über dem Alphabet $\{a, b\}$:



Führen Sie den Minimierungsalgorithmus für A durch und geben Sie den minimalen äquivalenten DEA für $L(A)$ als Zeichnung an.

46115 / 2019 / Herbst / Thema 1 / Aufgabe 2

- (a) Betrachten Sie die folgenden Sprachen:

$$L_1 = \{a^n b^m \mid n, m \in \mathbb{N}\}$$

$L_2 = \{a^n b^m \mid n, m \in \mathbb{N}\}$ Zeigen Sie für L_1 und L_2 , ob sie kontextfrei sind oder nicht. Für den Beweis von Kontext-Freiheit in dieser Frage reicht die Angabe eines Automaten oder einer Grammatik. (Beschreiben Sie dann die Konstruktionsidee des Automaten oder der Grammatik.) Für den Beweis von Nicht-Kontext-Freiheit verwenden Sie eine der üblichen Methoden.

(b) Eine kontextfreie Grammatik ist in Chomsky-Normalform, falls die folgenden Bedingungen erfüllt sind:

- alle Regeln sind von der Form $X \rightarrow YZ$ oder $X \rightarrow o$ mit Nichtterminalzeichen X, Y, Z und Terminalzeichen o ,
- alle Nichtterminalzeichen sind erreichbar vom Startsymbol und

- alle Nichtterminalzeichen sind erzeugend, d. h. für jedes Nichtterminalzeichen X gibt es ein Wort w über dem Terminalalphabet, so dass $X \Rightarrow^* w$.

Bringen Sie die folgende Grammatik in Chomsky-Normalform.

$$P = \left\{ \begin{array}{l} S \rightarrow AAB \mid CD \mid abc \\ A \rightarrow AAAA \mid a \\ B \rightarrow BB \mid S \\ C \rightarrow CCC \mid CC \\ D \rightarrow d \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gf7f9tp7z

Das Startsymbol der Grammatik ist S , das Terminalalphabet ist a,b,c,d und die Menge der Nichtterminalzeichen ist S,A,B,C,D .

Lösungsvorschlag

(i) **Elimination der ε -Regeln**

— Alle Regeln der Form $A \rightarrow \varepsilon$ werden eliminiert. Die Ersetzung von A wird durch ε in allen anderen Regeln vorweggenommen.

Ø Nichts zu tun

(ii) **Elimination von Kettenregeln**

— Jede Produktion der Form $A \rightarrow B$ mit $A, B \in S$ wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.

Eine rechte Seite in der C vorkommt, lässt sich wegen $\{C \rightarrow CCC \mid CC\}$ nicht ableiten, weil es zu einer Endlosschleife kommt. Wir entfernen die entsprechenden Regeln.

$$P = \left\{ \begin{array}{l} S \rightarrow AAB \mid abc \\ A \rightarrow AAAA \mid a \\ B \rightarrow BB \mid AAB \mid abc \end{array} \right\}$$

(iii) **Separation von Terminalzeichen**

— Jedes Terminalzeichen σ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal S_σ ersetzt und die Menge der Produktionen durch die Regel $S_\sigma \rightarrow \sigma$ ergänzt.

$$P = \left\{ \right.$$

$$\begin{aligned}
 S &\rightarrow AAB \mid T_a T_b T_c \\
 A &\rightarrow AAAA \mid a \\
 B &\rightarrow BB \mid AAB \mid T_a T_b T_c \\
 T_a &\rightarrow a \\
 T_b &\rightarrow b \\
 T_c &\rightarrow c
 \end{aligned}$$

}

(iv) **Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form $A \rightarrow B_1 B_2 \dots B_n$ werden in die Produktionen $A \rightarrow A_{n-1} B_n, A_{n-1} \rightarrow A_{n-2} B_{n-1}, \dots, A_2 \rightarrow B_1 B_2$ zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. —

$$P = \{$$

$$\begin{aligned}
 S &\rightarrow AS_1 \mid T_a S_2 \\
 A &\rightarrow AA_1 \mid a \\
 B &\rightarrow BB \mid AS_1 \mid T_a S_2 \\
 T_a &\rightarrow a \\
 T_b &\rightarrow b \\
 T_c &\rightarrow c \\
 S_1 &\rightarrow AB \\
 S_2 &\rightarrow T_b T_c \\
 A_1 &\rightarrow AA_2 \\
 A_2 &\rightarrow AA
 \end{aligned}$$

}

46115 / 2019 / Herbst / Thema 1 / Aufgabe 4

In der folgenden Aufgabe soll ein Feld A von ganzen Zahlen *aufsteigend* sortiert werden. Das Feld habe n Elemente $A[1]$ bis $A[n]$. Der folgende Algorithmus sei gegeben:

```

var A : array[1..n] of integer;

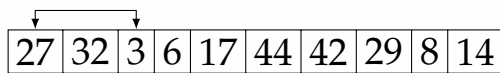
procedure selection_sort
var i, j, smallest, tmp : integer;
begin
  for j := 1 to n-1 do begin
    smallest := j;
    for i := j + 1 to n do begin
      if A[i] < A[smallest] then
        smallest := i;
    end
  end
end

```

```
    tmp = A[j];  
    A[j] = A[smallest];  
    A[smallest] = tmp;  
end  
end
```

- (a) Sortieren Sie das folgende Feld mittels des Algorithmus. Notieren Sie alle Werte, die die Variable *smallest* jeweils beim Durchlauf der inneren Schleife annimmt. Geben Sie die Belegung des Feldes nach jedem Durchlauf der äußeren Schleife in einer neuen Zeile an.

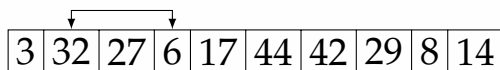
Ausgang



27	32	3	6	17	44	42	29	8	14
----	----	---	---	----	----	----	----	---	----

nach 1. Durchlauf ($j = 1$)

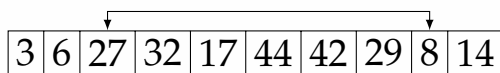
smallest: (1) 3



3	32	27	6	17	44	42	29	8	14
---	----	----	---	----	----	----	----	---	----

nach 2. Durchlauf ($j = 2$)

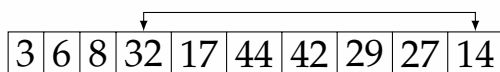
smallest: (2) 3 4



3	6	27	32	17	44	42	29	8	14
---	---	----	----	----	----	----	----	---	----

nach 3. Durchlauf ($j = 3$)

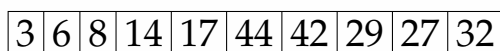
smallest: (3) 5 9



3	6	8	32	17	44	42	29	27	14
---	---	---	----	----	----	----	----	----	----

nach 4. Durchlauf ($j = 4$)

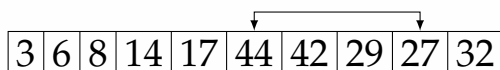
smallest: (4) 5 10



3	6	8	14	17	44	42	29	27	32
---	---	---	----	----	----	----	----	----	----

nach 5. Durchlauf ($j = 5$)

smallest: (5) -



3	6	8	14	17	44	42	29	27	32
---	---	---	----	----	----	----	----	----	----

nach 6. Durchlauf ($j = 6$)

smallest: (6) 7 8 9

3	6	8	14	17	27	42	29	44	32
---	---	---	----	----	----	----	----	----	----

nach 7. Durchlauf ($j = 7$)

smallest: (7) 8

3	6	8	14	17	27	29	42	44	32
---	---	---	----	----	----	----	----	----	----

nach 8. Durchlauf ($j = 8$)

smallest: (8) 10

3	6	8	14	17	27	29	32	44	42
---	---	---	----	----	----	----	----	----	----

nach 9. Durchlauf ($j = 9$)

smallest: (9) 10

3	6	8	14	17	27	29	32	44	42
---	---	---	----	----	----	----	----	----	----

fertig

3	6	8	14	17	27	29	32	42	44
---	---	---	----	----	----	----	----	----	----

- (b) Der Wert der Variablen *smallest* wird bei jedem Durchlauf der äußeren Schleife mindestens ein Mal neu gesetzt. Wie muss das Feld *A* beschaffen sein, damit der Variablen *smallest* ansonsten niemals ein Wert zugewiesen wird? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Wenn das Feld bereits aufsteigend sortiert ist, dann nimmt die Variable *smallest* in der inneren Schleife niemals einen neuen Wert an.

- (c) Welche Auswirkung auf die Sortierung oder auf die Zuweisungen an die Variable *smallest* hat es, wenn der Vergleich in Zeile 9 des Algorithmus statt $A[i] < A[\text{smallest}]$ lautet $A[i] \leq A[\text{smallest}]$? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Der Algorithmus sortiert dann nicht mehr *stabil*, d.h. die Eingabereihenfolge von Elementen mit *gleichem* Wert wird beim Sortieren nicht mehr *bewahrt*.

- (d) Betrachten Sie den Algorithmus unter der Maßgabe, dass Zeile 9 wie folgt geändert wurde:

```
if A[i] > A[smallest] then
```

Welches Ergebnis berechnet der Algorithmus nun?

Lösungsvorschlag

Der Algorithmus sortiert jetzt absteigend.

- (e) Betrachten Sie die folgende *rekursive* Variante des Algorithmus. Der erste Parameter ist wieder das zu sortierende Feld, der Parameter n ist die Größe des Feldes und der Parameter $index$ ist eine ganze Zahl. Die Funktion $\text{min_index}(A, x, y)$ berechnet für $1 \leq x \leq y \leq n$ den Index des kleinsten Elements aus der Menge $\{A[x], A[x+1], \dots, A[y]\}$

```
procedure rek_selection_sort(A, n, index : integer)
var k, tmp : integer;
begin
if (Abbruchbedingung) then return;
  k = min_index(A, index, n);
  if k <> index then begin
    tmp := A[k];
    A[k] := A[index];
    A[index] := tmp;
  end
  (rekursiver Aufruf)
end
```

Der initiale Aufruf des Algorithmus lautet: `rek_selection_sort(A, n, 1)`

Vervollständigen Sie die fehlenden Angaben in der Beschreibung des Algorithmus für

- die Abbruchbedingung in Zeile 4 und

Lösungsvorschlag

`n = index` bzw `n == index`

Begründung: Wenn der aktuelle Index so groß ist wie die Anzahl der Elemente im Feld, dann muss / darf abgebrochen werden, denn dann ist das Feld sortiert.

- den rekursiven Aufruf in Zeile 11.

Lösungsvorschlag

`rek_selection_sort(A, n, index + 1)`

Am Ende der Methode wurde an die Index-Position $index$ das kleinste Element gesetzt, jetzt muss an die nächste Index-Position ($index + 1$) der kleinste Wert, der noch nicht sortieren Zahlen, gesetzt werden.

Begründen Sie Ihre Antworten.

```
import static org.bsclangaul.helfer.Konsole.zeigeZahlenFeld;

public class SelectionSort {

    public static void selectionSort(int[] A) {
        int smallest, tmp;

        for (int j = 0; j < A.length - 1; j++) {
            System.out.println("\nj = " + (j + 1));
            smallest = j;
            for (int i = j + 1; i < A.length; i++) {
                if (A[i] < A[smallest]) {
                    smallest = i;
                    System.out.println(smallest + 1);
                }
            }
            tmp = A[j];
            A[j] = A[smallest];
            A[smallest] = tmp;
            zeigeZahlenFeld(A);
        }
    }

    public static void rekSelectionSort(int[] A, int n, int index) {
        int k, tmp;

        if (index == n - 1) {
            return;
        }
        k = minIndex(A, index, n);
        if (k != index) {
            tmp = A[k];
            A[k] = A[index];
            A[index] = tmp;
        }
        rekSelectionSort(A, n, index + 1);
    }

    public static int minIndex(int[] A, int x, int y) {
        int smallest = x;
        for (int i = x; i < y; i++) {
            if (A[i] < A[smallest]) {
                smallest = i;
            }
        }
        return smallest;
    }

    public static void main(String[] args) {
        int[] A = new int[] { 27, 32, 3, 6, 17, 44, 42, 29, 8, 14 };
        selectionSort(A);

        A = new int[] { 27, 32, 3, 6, 17, 44, 42, 29, 8, 14 };
        rekSelectionSort(A, A.length, 0);
        zeigeZahlenFeld(A);
    }
}
```

```
}  
  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/SelectionSort.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2019/herbst/SelectionSort.java)

46115 / 2019 / Herbst / Thema 1 / Aufgabe 6

Gegeben sei die Implementierung eines Stacks ganzer Zahlen mit folgender Schnittstelle:

```
import java.util.Stack;  
  
/**  
 * Um schnell einen lauffähigen Stack zu bekommen, verwenden wir den Stack aus  
 * der Java Collection.  
 */  
public class IntStack {  
    private Stack<Integer> stack = new Stack<Integer>();  
  
    /**  
     * Legt Element i auf den Stack.  
     *  
     * @param i Eine Zahl, die auf dem Stack gelegt werden soll.  
     */  
    public void push(int i) {  
        stack.push(i);  
    }  
  
    /**  
     * Gibt oberstes Element vom Stack.  
     *  
     * @return Das oberste Element auf dem Stapel.  
     */  
    public int pop() {  
        return stack.pop();  
    }  
  
    /**  
     * Fragt ab, ob Stack leer ist.  
     *  
     * @return Wahr, wenn der Stapel leer ist.  
     */  
    public boolean isEmpty() {  
        return stack.empty();  
    }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/mystery_stack/IntStack.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2019/herbst/mystery_stack/IntStack.java)

Betrachten Sie nun die Realisierung der folgenden Datenstruktur **Mystery**, die zwei Stacks benutzt.

```
public class Mystery {
    private IntStack a = new IntStack();
    private IntStack b = new IntStack();

    public void foo(int item) {
        a.push(item);
    }

    public int bar() {
        if (b.isEmpty()) {
            while (!a.isEmpty()) {
                b.push(a.pop());
            }
        }
        return b.pop();
    }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/mystery_stack/Mystery.java

- (a) Skizzieren Sie nach jedem Methodenaufruf der im folgenden angegebenen Befehlssequenz den Zustand der beiden Stacks eines Objekts `m` der Klasse `Mystery`. Geben Sie zudem bei jedem Aufruf der Methode `bar` an, welchen Wert diese zurückliefert.

```
Mystery m = new Mystery();
m.foo(3);
m.foo(5);
m.foo(4);
m.bar();
m.foo(7);
m.bar();
m.foo(2);
m.bar();
m.bar();
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/mystery_stack/Mystery.java

Lösungsvorschlag

Code	Stack b	Stack a	Rückgabewert
<code>m.foo(3);</code>	{ 3 }	{ }	
<code>m.foo(5);</code>	{ 5, 3 }	{ }	
<code>m.foo(4);</code>	{ 4, 5, 3 }	{ }	
<code>m.bar();</code>	{ }	{ 5, 4 }	3
<code>m.foo(7);</code>	{ 7 }	{ 5, 4 }	
<code>m.bar();</code>	{ 7 }	{ 4 }	5
<code>m.foo(2);</code>	{ 2, 7 }	{ }	
<code>m.bar();</code>	{ 2, 7 }	{ }	4
<code>m.bar();</code>	{ }	{ 2 }	7

- (b) Sei n die Anzahl der in einem Objekt der Klasse `Mystery` gespeicherten Werte. Im folgenden wird gefragt, wieviele Aufrufe von Operationen der Klasse `IntStack` einzelne Aufrufe von Methoden der Klasse `Mystery` verursachen. Begründen Sie jeweils Ihre Antwort.

- (i) Wie viele Aufrufe von Operationen der Klasse `IntStack` verursacht die Methode `foo(x)` im besten Fall?

Lösungsvorschlag

Einen Aufruf, nämlich `a.push(i)`

- (ii) Wie viele Aufrufe von Operationen der Klasse `IntStack` verursacht die Methode `foo(x)` im schlechtesten Fall?

Lösungsvorschlag

Einen Aufruf, nämlich `a.push(i)`

- (iii) Wie viele Aufrufe von Operationen der Klasse `IntStack` verursacht die Methode `bar()` im besten Fall?

Lösungsvorschlag

Wenn der Stack `b` nicht leer ist, dann werden zwei Aufrufe benötigt, nämlich `b.isEmpty()` und `b.pop()`

- (iv) Wie viele Aufrufe von Operationen der Klasse `IntStack` verursacht die Methode `bar()` im schlechtesten Fall?

Lösungsvorschlag

Wenn der Stack `b` leer ist, dann liegen all n Objekte im Stack `a`. Die Objekte im Stack `a` werden in der `while`-Schleife nach `b` verschoben. Pro Objekt sind drei Aufrufe nötig, also $3 \cdot n$. `b.isEmpty()` (erste Zeile in der Methode) und `b.pop()` (letzte Zeile in der Methode) wird immer aufgerufen. Wenn alle Objekt von `a` nach `b` verschoben wurden, wird

zusätzlich noch einmal in der Bedingung der `while`-Schleife `a.isEmpty()` aufgerufen. Im schlechtesten Fall werden also $3 \cdot n + 3$ Operationen der Klasse `IntStack` aufgerufen.

- (c) Welche allgemeinen Eigenschaften werden durch die Methoden `foo` und `bar` realisiert? Unter welchem Namen ist diese Datenstruktur allgemein bekannt?

Lösungsvorschlag

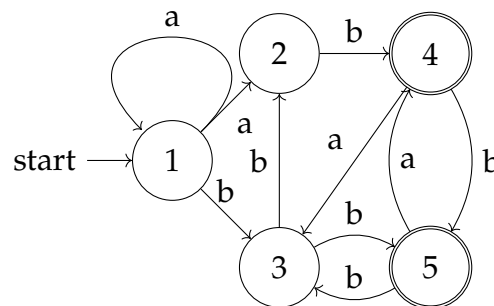
`foo()` Legt das Objekt auf den Stack `a`. Das Objekt wird in die Warteschlange eingereiht. Die Methode müsste eigentlich `enqueue()` heißen.

`bar()` Verschiebt alle Objekte vom Stack `a` in umgekehrter Reihenfolge in den Stack `b`, aber nur dann, wenn Stack `b` leer ist. Entfernt dann den obersten Wert aus dem Stack `b` und gibt ihn zurück. Das zuerst eingereihte Objekt wird aus der Warteschlange entnommen. Die Methode müsste eigentlich `dequeue()` heißen.

Die Datenstruktur ist unter dem Namen Warteschlange oder Queue bekannt

46115 / 2019 / Herbst / Thema 2 / Aufgabe 1

Es sei der nichtdeterministische endliche Automat $A = (\{1, 2, 3, 4, 5\}, \{a, b\}, \delta, \{4, 5\}, 1)$ gegeben, wobei δ durch folgenden Zeichnung beschrieben ist.

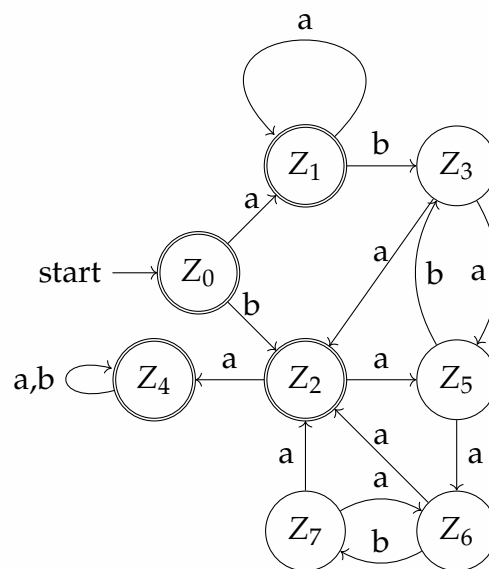


Konstruieren Sie nachvollziehbar einen deterministischen endlichen Automaten A' , der das Komplement von $L(A)$ akzeptiert!

Lösungsvorschlag

Zuerst mit Hilfe der Potenzmengenkonstruktion einen deterministischen endlichen Automaten erstellen und dann die Zustände mit den Endzuständen tauschen.

Name	Zustandsmenge	Eingabe a	Eingabe b
Z_0	$Z_0\{1\}$	$Z_1\{1,2\}$	$Z_2\{3\}$
Z_1	$Z_1\{1,2\}$	$Z_1\{1,2\}$	$Z_3\{3,4\}$
Z_2	$Z_2\{3\}$	$Z_4\{\}$	$Z_5\{2,5\}$
Z_3	$Z_3\{3,4\}$	$Z_2\{3\}$	$Z_5\{2,5\}$
Z_4	$Z_4\{\}$	$Z_4\{\}$	$Z_4\{\}$
Z_5	$Z_5\{2,5\}$	$Z_6\{4\}$	$Z_3\{3,4\}$
Z_6	$Z_6\{4\}$	$Z_2\{3\}$	$Z_7\{5\}$
Z_7	$Z_7\{5\}$	$Z_6\{4\}$	$Z_2\{3\}$



46115 / 2019 / Herbst / Thema 2 / Aufgabe 2

Gegeben ist die rechtslineare Grammatik $G = (\{a, b\}, \{S, A, B, C, D\}, S, P)$ mit

$P = \{$

$$S \rightarrow aA$$

$$A \rightarrow bB$$

$$A \rightarrow aD$$

$$B \rightarrow aC$$

$$B \rightarrow bB$$

$$C \rightarrow bD$$

$$C \rightarrow b$$

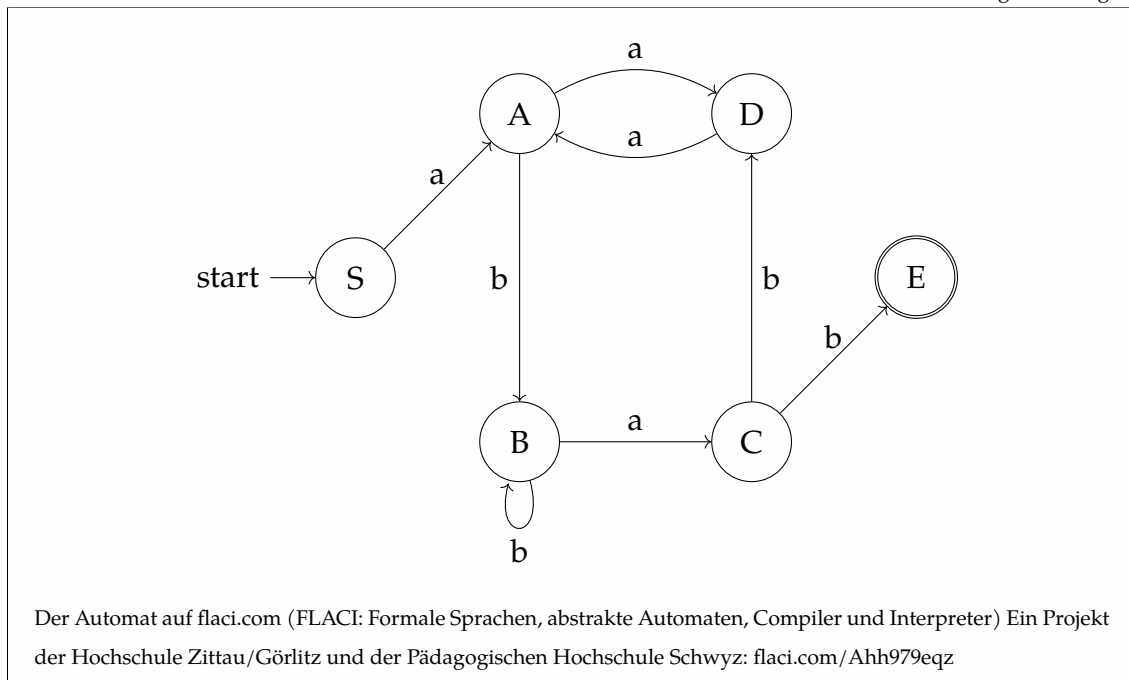
$$D \rightarrow aA$$

. Sei L die von G erzeugte Sprache.

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gpkv4ansc

- (a) Zeichnen Sie einen nichtdeterministischen endlichen Automaten, der L akzeptiert!

Lösungsvorschlag



- (b) Konstruieren Sie auf nachvollziehbare Weise einen regulären Ausdruck α mit $L(\alpha) = L$!

Lösungsvorschlag

$(ab + ab|(a|b+) + ab + ab)$ (von Flaci automatisch konvertiert)

46115 / 2019 / Herbst / Thema 2 / Aufgabe 7

Schreiben Sie in Pseudocode eine Methode `heapify(int[] a)`, welche im übergebenen Array der Länge n die Heapeigenschaft in $\mathcal{O}(n)$ Schritten herstellt. D. h. als Ergebnis soll in a gelten, dass $a[i] \leq a[2i + 1]$ und $a[i] \leq a[i + 2]$.

Lösungsvorschlag

```

import org.bschlangaul.helfer.Konsole;

/**
 * Nach Pseudocode nach
 * https://www.oreilly.com/library/view/algorithms-in-a/9780596516246/ch04s06.html
 */
public class Heapify {

    public static void buildHeap(int a[]) {
        int n = a.length;
    }
  
```

```
    for (int i = n / 2 - 1; i >= 0; i--) {
        heapify(a, i, n);
    }
}

public static void heapify(int a[], int index, int max) {
    int left = 2 * index + 1;
    int right = 2 * index + 2;
    int smallest;

    if (left < max && a[left] < a[index]) {
        smallest = left;
    } else {
        smallest = index;
    }

    if (right < max && a[right] < a[smallest]) {
        smallest = right;
    }

    if (smallest != index) {
        int tmp = a[index];
        a[index] = a[smallest];
        a[smallest] = tmp;
        heapify(a, smallest, max);
    }
}

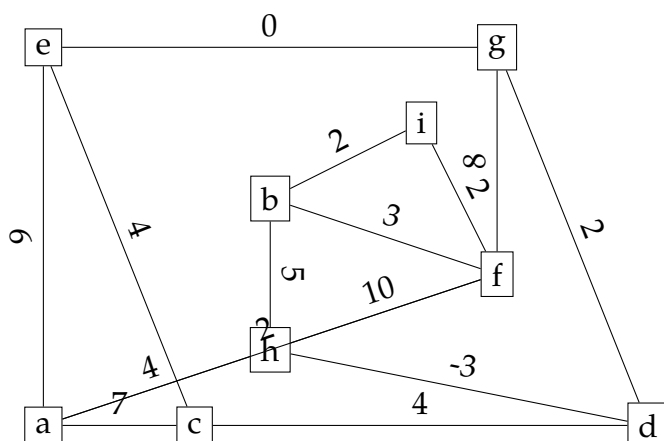
public static void main(String[] args) {
    int[] a = new int[] { 5, 3, 16, 2, 10, 14 };
    buildHeap(a);
    Konsole.zeigeZahlenFeld(a); // 2 3 14 5 10 16
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/Heapify.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2019/herbst/Heapify.java)

46115 / 2019 / Herbst / Thema 2 / Aufgabe 8

- (a) Durch folgende Adjazenzmatrix sei ein ungerichteter Graph G mit Kantenlängen gegeben.

	a	b	c	d	e	f	g	h	i
a	0	0	7	0	9	2	0	4	0
b	0	0	0	0	0	3	0	5	2
c	7	0	0	4	4	0	0	0	0
d	0	0	4	0	0	0	2	-3	0
e	9	0	4	0	0	0	0	0	0
f	2	3	0	0	0	0	8	10	2
g	0	0	0	2	0	8	0	0	0
h	4	5	0	-3	0	10	0	0	0
i	0	2	0	0	0	2	0	0	0



Wenden Sie den Algorithmus von Jarník/Prim auf G ausgehend von Knoten d an, um einen Spannbaum T mit *maximalem* Gewicht zu berechnen. Erstellen Sie dazu eine Tabelle mit zwei Spalten und stellen Sie jeden einzelnen Schritt des Verfahrens in einer eigenen Zeile dar. Geben Sie in der ersten Spalte denjenigen Knoten v an, der vom Algorithmus als nächstes in T aufgenommen wird (dieser sog. „schwarze“ Knoten ist damit fertiggestellt). Führen Sie in der zweiten Spalte alle anderen vom aktuellen Baum T direkt erreichbaren Knoten v (sog. „graue Randknoten“) auf.

Geben Sie in der Tabelle Knoten stets als Tripel $(v, \delta, v.\pi)$ an, mit v als Knotenname, $v.\pi$ als aktueller Vorgängerknoten (anderer Knoten der Kante) und δ als Länge der Kante $\{v, v.\pi\}$.

- (b) Sei $G = (V, E, w)$ ein Graph mit Kantenlängen $w : E \rightarrow \mathbb{N}$ und T ein Spannbaum von G mit maximalem Gewicht. Beweisen oder widerlegen Sie die folgende Aussage:

Längste (einfache) Wege zwischen zwei Knoten $u, v \in V$ enthalten nur Kanten aus T .

46115 / 2020 / Frühjahr / Thema 1 / Aufgabe 1

- (a) Betrachten Sie die formale Sprache $L \subseteq \{0,1\}^*$ aller Wörter, die 01 oder 110 als Teilwort enthalten.

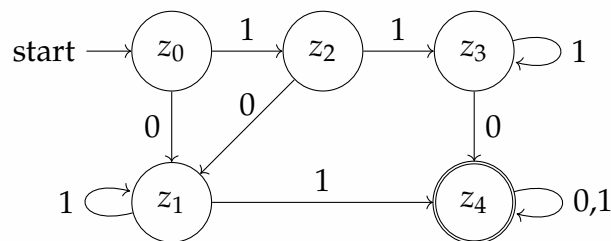
Geben Sie einen regulären Ausdruck für die Sprache L an.

Lösungsvorschlag

$$(0|1)^*(01|110)(0|1)^*$$

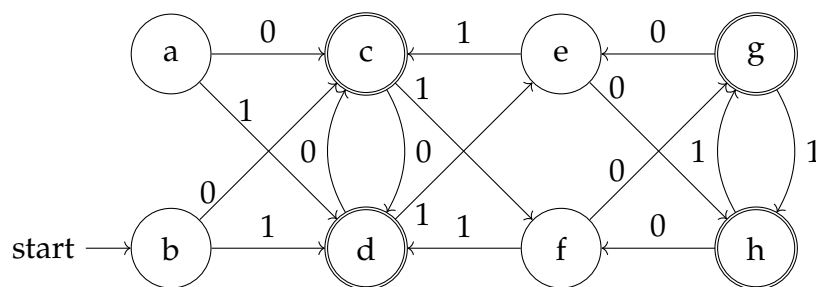
- (b) Entwerfen Sie einen (vollständigen) deterministischen endlichen Automaten, der die Sprache L aus Teilaufgabe (a) akzeptiert. (Hinweis: es werden nicht mehr als 6 Zustände benötigt.)

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A54gek0vz

- (c) Minimieren Sie den folgenden deterministischen endlichen Automaten:
Machen Sie dabei Ihren Rechenweg deutlich!



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajpw4j73w

Lösungsvorschlag

a	∅	∅	∅	∅	∅	∅	∅	∅
b		∅	∅	∅	∅	∅	∅	∅
c	x_1	x_1	∅	∅	∅	∅	∅	∅
d	x_1	x_1		∅	∅	∅	∅	∅
e	x_2	x_2	x_1	x_1	∅	∅	∅	∅
f	x_3	x_3	x_1	x_1		∅	∅	∅
g	x_1	x_1	x_2	x_2	x_1	x_1	∅	∅
h	x_1	x_1	x_2	x_2	x_1	x_1		∅
	<u>a</u>	<u>b</u>	<u>c</u>	<u>d</u>	<u>e</u>	<u>f</u>	<u>g</u>	<u>h</u>

x_1 Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.

x_2 Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

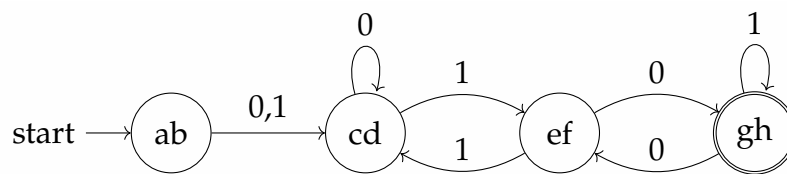
x_3 In weiteren Iterationen markierte Zustände.

x_4 ...

Die Zustandpaare werden aufsteigend sortiert notiert.

Übergangstabelle

Zustandspaar	0	1
(a, b)	(c, c)	(d, d)
(a, e)	(c, h) x_2	(c, d)
(a, f)	(c, g) x_3	(d, d)
(b, e)	(c, h) x_2	(c, d)
(b, f)	(c, g) x_3	(d, g)
(c, d)	(c, d)	(e, f)
(c, g)	(d, e) x_2	(e, f)
(c, h)	(d, f) x_2	(f, f)
(d, g)	(c, e) x_2	(e, e)
(d, h)	(c, f) x_2	(e, f)
(e, f)	(g, h)	(c, d)
(g, h)	(e, f)	(g, h)



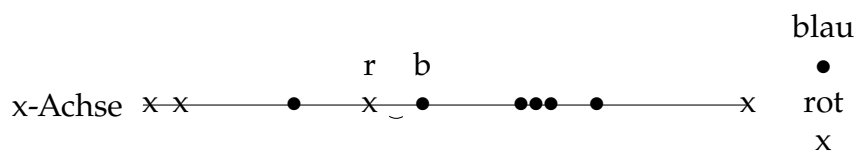
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arzvh5kyz

(d) Ist die folgende Aussage richtig oder falsch? Begründen Sie Ihre Antwort!

„Zu jeder regulären Sprache L über dem Alphabet Σ gibt es eine Sprache $L' \subseteq \Sigma^*$, die L enthält ($\emptyset \subseteq L$) und nicht regulär ist.“

46115 / 2020 / Frühjahr / Thema 2 / Aufgabe 6

Gegeben seien zwei nichtleere Mengen R und B von roten bzw. blauen Punkten auf der x -Achse. Gesucht ist der minimale euklidische Abstand $d(r, b)$ über alle Punktpaare (r, b) mit $r \in R$ und $b \in B$. Hier ist eine Beispielinstantz:



Die Eingabe wird in einem Feld A übergeben. Jeder Punkt $A[i]$ mit $1 \leq i \leq n$ hat eine x -Koordinate $A[i].x$ und eine Farbe $A[i].color \in \{\text{rot}, \text{blau}\}$. Das Feld A ist nach x -Koordinate sortiert, d.h. es gilt $A[1].x < A[2].x < \dots < A[n].x$, wobei $n = |R| + |B|$.

(a) Geben Sie in Worten einen Algorithmus an, der den gesuchten Abstand in $\mathcal{O}(n)$ Zeit berechnet.

Lösungsvorschlag

Pseudo-Code

Algorithmus 1: Minimaler Euklidischer Abstand

```

 $d_{min} := \max ;$  // Setze  $d_{min}$  zuerst auf einen maximalen Wert.
for  $i$  in  $0 \dots \text{vorletzter Index}$  do ; // Iteriere über die Indizes des Punkte-Arrays
     $P$  bis zum vorletzten Index  $P[n-1]$ 

    if  $P[n].color \neq P[n+1].color$  then ; // Berechne den Abstand nur, wenn die
        Punkte unterschiedliche Farben haben

         $d = P[n+1].x - P[n].x$ 
        if  $d < d_{min}$  then
             $d_{min} = d$ 
        end
    end
end

```

Java

```

public double findMinimalDistance() {
    double distanceMin = Double.MAX_VALUE;
    for (int i = 0; i < latestIndex - 1; i++) {
        if (points[i].color != points[i + 1].color) {
            double distance = points[i + 1].x - points[i].x;
            if (distance < distanceMin) {
                distanceMin = distance;
            }
        }
    }
    return distanceMin;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/fruehjahr/RedBluePairCollection.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/fruehjahr/RedBluePairCollection.java)

- (b) Begründen Sie kurz die Laufzeit Ihres Algorithmus.

Lösungsvorschlag

Da das Array der Länge n nur einmal durchlaufen wird, ist die Laufzeit $\mathcal{O}(n)$ sichergestellt.

- (c) Begründen Sie die Korrektheit Ihres Algorithmus.

Lösungsvorschlag

In d_{min} steht am Ende der gesuchte Wert (sofern nicht $d_{min} = \text{Integer.MAX_VALUE}$ geblieben ist)

- (d) Wir betrachten nun den Spezialfall, dass alle blauen Punkte links von allen roten Punkten liegen. Beschreiben Sie in Worten, wie man in dieser Situation den gesuchten Abstand in $o(n)$ Zeit berechnen kann. (Ihr Algorithmus darf also insbesondere nicht Laufzeit $\Theta(n)$ haben.)

Zuerst müssen wir den letzten blauen Punkt finden. Das ist mit einer binären Suche möglich. Wir beginnen mit dem ganzen Feld als Suchbereich und betrachten den mittleren Punkt. Wenn er blau ist, wiederholen wir die Suche in der zweiten Hälfte des Suchbereichs, sonst in der ersten, bis wir einen blauen Punkt gefolgt von einem roten Punkt gefunden haben.

Der gesuchte minimale Abstand ist dann der Abstand zwischen dem gefundenen blauen und dem nachfolgenden roten Punkt. Die Binärsuche hat eine Worst-case-Laufzeit von $\mathcal{O}(\log n)$.

46115 / 2020 / Frühjahr / Thema 2 / Aufgabe 7

Sei H ein Max-Heap, der n Elemente speichert. Für ein Element v in H sei $h(v)$ die Höhe von v , also die Länge eines längsten Pfades von v zu einem Blatt im Teilheap mit Wurzel v .

- (a) Geben Sie eine rekursive Definition von $h(v)$ an, in der Sie sich auf die Höhen der Kinder $v.\text{left}$ und $v.\text{right}$ von v beziehen (falls v Kinder hat).
- (b) Geben Sie eine möglichst niedrige obere asymptotische Schranke für die Summe der Höhen aller Elemente in H an, also für $\sum_{v \in H} h(v)$ und begründen Sie diese.
Tipp: Denken Sie daran, wie man aus einem beliebigen Feld einen Max-Heap macht.
- (c) Sei H' ein Feld der Länge n . Geben Sie einen Algorithmus an, der in Linearzeit testet, ob H ein Max-Heap ist.

46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1

Ein moderner Kaugummi-Automat erhalte 20- und 50-Cent-Münzen. Eine Packung Kaugummi kostet 120 Cent. Die interne Steuerung des Kaugummi-Automaten verwendet einen deterministischen endlichen Automaten, der die Eingabe als Folge von 20- und 50-Cent-Münzen (d. h. als Wort über dem Alphabet 20,50) erhält, und genau die Folgen akzeptiert, die in der Summe 120 Cent ergeben.

- (a) Geben Sie zwei Worte aus $20,50^*$ an, die der Automat akzeptiert.
- (b) Zeichnen Sie einen deterministischen endlichen Automaten als Zustandsgraph, der für die interne Steuerung des Kaugummi-Automaten verwendet werden kann (d. h. der Automat akzeptiert genau alle Folgen von 20- und 50-Cent-Münzen, deren Summe 120 ergibt).
- (c) Geben Sie einen regulären Ausdruck an, der die akzeptierte Sprache des Automaten erzeugt.

46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

- (a) Verwenden Sie den Algorithmus von Cocke, Younger und Kasami (CYK-Algorithmus), um für die folgende kontextfreie Grammatik $G = (V, D, P, 5)$ mit Variablen $V = \{s, A, B, C, D\}$, Terminalzeichen $\Sigma = \{a, b\}$, Produktionen

$P = \{S \rightarrow SB \mid AC \mid a, A \rightarrow a, B \rightarrow b, C \rightarrow DD \mid AB, D \rightarrow AB \mid DC \mid CD\}$

und Startsymbol S zu prüfen, ob das Wort $aabababb$ in der durch G erzeugten Sprache liegt. Erläutern Sie dabei Ihr Vorgehen und den Ablauf des CYK-Algorithmus.

- (b) Mit a^i , wobei $i \in \mathbb{N}_0 = \{0, 1, 2, \dots\}$, wird das Wort bezeichnet, das aus der i -fachen Wiederholung des Zeichens a besteht (d. h. $a^0 = \epsilon$ und $a^i = aa^{i-1}$, wobei ϵ das leere Wort ist).

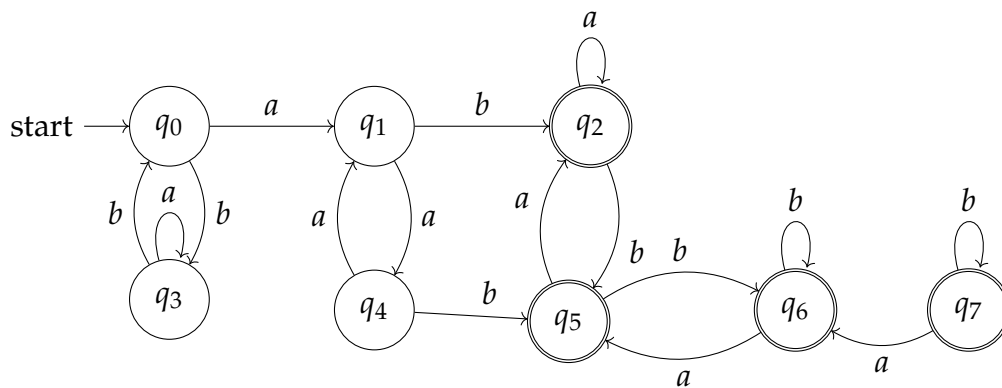
Sei die Sprache L definiert über dem Alphabet $\{a, b\}$ als

$L = \{a^i b^j \mid i \in \mathbb{N}_0, j > 1\}$.

Zeigen Sie, dass die Sprache L nicht vom Typ 3 der Chomsky-Hierarchie (d. h. nicht regulär) ist.

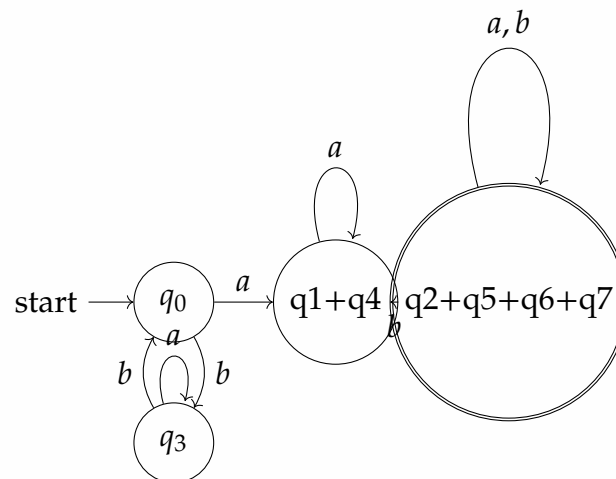
46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Betrachten Sie den unten gezeigten deterministischen endlichen Automaten, der Worte über dem Alphabet $\Sigma = \{a, b\}$ verarbeitet. Bestimmen Sie den dazugehörigen Minimalautomaten, d. h. einen deterministischen endlichen Automaten, der die gleiche Sprache akzeptiert und eine minimale Anzahl an Zuständen benutzt. Erläutern Sie Ihre Berechnung, indem Sie z. B. eine Minimierungstabelle angeben.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ah5v10or9

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apkyuoo1g

46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 4

Beweisen Sie, dass das folgende Entscheidungsproblem semi-entscheidbar ist.

Eingabe: Eine Turingmaschine M Aufgabe: Entscheiden Sie, ob ein Eingabewort existiert, auf das M hält.

46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 1

- (a) Geben Sie für folgende Sortiervverfahren jeweils zwei Felder A und B an, so dass das jeweilige Sortiervverfahren angewendet auf A seine Best-Case-Laufzeit und angewendet auf B seine Worst-Case-Laufzeit erreicht. (Wir messen die Laufzeit durch die Anzahl der Vergleiche zwischen Elementen der Eingabe.) Dabei soll das Feld A die Zahlen $1, 2, \dots, 7$ genau einmal enthalten; das Feld B ebenso. Sie bestimmen also nur die Reihenfolge der Zahlen.

Wenden Sie als Beleg für Ihre Aussagen das jeweilige Sortiervverfahren auf die Felder A und B an und geben Sie nach jedem größeren Schritt des Algorithmus den Inhalt der Felder an.

Geben Sie außerdem für jedes Verfahren asymptotische Best- und Worst-Case-Laufzeit für ein Feld der Länge n an.

Für drei der Sortiervverfahren ist der Pseudocode angegeben. Beachten Sie, dass die Feldindizes hier bei 1 beginnen. Die im Pseudocode verwendete Unterroutine $\text{Swap}(A, i, j)$ vertauscht im Feld A die Elemente mit den Indizes i und j miteinander.

- (i) Insertionsort
- (ii) Bubblesort
- (iii) Quicksort

```
Insertionsort(int[] A) for i = 2 to A.length do key = A[i] j = i-1 while j > 0 and A[j] > key do A[j+1] = A[j] j = j-1 A[j+1] = key
```

```
Bubblesort(int[] A) n := length(A) repeat swapped = false for i = n-1 to 1 do if A[i] > A[i+1] then Swap(A,i,i+1)
```

```
swapped := true
```

```
until not swapped
```

```
Quicksort(int[] A, l = 1, r = A.length) if l < r then m = Partition(A, l, r) | Quicksort(A, l, m-1) Quicksort(A, m+1, r)
```

```
int Partition (int[] A, int l, int r)
```

```
pivot = A[r]
```

```
i = l
```

```
for j = l to r-1 do
```

```
if A[j] < pivot then
```

```
Swap(A, i, j) i = i+1
```

```
Swap(A, i, r)
```

```
return i
```

- (b) Geben Sie die asymptotische Best- und Worst-Case-Laufzeit von Mergesort an.

46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2

- (a) Argumentieren Sie, warum man das Maximum von n Zahlen nicht mit weniger als $n - 1$ Vergleichen bestimmen kann.

Lösungsvorschlag

Wenn die n Zahlen in einem unsortierten Zustand vorliegen, müssen wir alle Zahlen betrachten, um das Maximum zu finden. Wir brauchen dazu $n - 1$ und nicht n Vergleiche, da wir die erste Zahl zu Beginn des Algorithmus als Maximum definieren und anschließend die verbleibenden Zahlen $n - 1$ mit dem aktuellen Maximum vergleichen.

- (b) Geben Sie einen Algorithmus im Pseudocode an, der das Maximum eines Feldes der Länge n mit genau $n - 1$ Vergleichen bestimmt.

Lösungsvorschlag

```
public static int bestimmeMaximum(int[] a) {
    int max = a[0];
    for (int i = 1; i < a.length; i++) {
        if (a[i] > max) {
            max = a[i];
        }
    }
    return max;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java)

- (c) Wenn man das Minimum und das Maximum von n Zahlen bestimmen will, dann kann das natürlich mit $2n - 2$ Vergleichen erfolgen. Zeigen Sie, dass man bei jedem beliebigen Feld mit deutlich weniger Vergleichen auskommt, wenn man die beiden Werte statt in zwei separaten Durchläufen in einem Durchlauf geschickt bestimmt.

Lösungsvorschlag

```
/**
 * Diese Methode ist nicht optimiert. Es werden  $2n - 2$  Vergleiche benötigt.
 *
 * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
→ gesucht
 *         werden soll.
 *
 * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das
→ Minimum,
 *         der zweite Eintrag das Maximum.
 */
public static int[] minMaxNaiv(int[] a) {
    int max = a[0];
    int min = a[0];
    for (int i = 1; i < a.length; i++) {
        if (a[i] > max) {
            max = a[i];
        }
        if (a[i] < min) {
            min = a[i];
        }
    }
    return new int[] { min, max };
}

/**
 * Diese Methode ist optimiert. Es werden immer zwei Zahlen paarweise
 * betrachtet. Die Anzahl der Vergleiche reduziert sich auf  $3n/2 + 2$  bzw.
 *  $3(n-1)/2 + 4$  bei einer ungeraden Anzahl an Zahlen im Feld.
 *
 * nach <a href=
 * "https://www.techiedelight.com/find-minimum-maximum-element-array-using-
→ minimum-comparisons/">techiedelight.com</a>
 *
 * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
→ gesucht
 *         werden soll.
 *
 * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das
→ Minimum,
 *         der zweite Eintrag das Maximum.
 */
public static int[] minMaxIterativPaarweise(int[] a) {
    int max = Integer.MIN_VALUE, min = Integer.MAX_VALUE;
    int n = a.length;
```

```
boolean istUngerade = (n & 1) == 1;
if (istUngerade) {
    n--;
}

for (int i = 0; i < n; i = i + 2) {
    int maximum, minimum;

    if (a[i] > a[i + 1]) {
        minimum = a[i + 1];
        maximum = a[i];
    } else {
        minimum = a[i];
        maximum = a[i + 1];
    }

    if (maximum > max) {
        max = maximum;
    }

    if (minimum < min) {
        min = minimum;
    }
}

if (istUngerade) {
    if (a[n] > max) {
        max = a[n];
    }

    if (a[n] < min) {
        min = a[n];
    }
}

return new int[] { min, max };
}

/**
 * Diese Methode ist nach dem Teile-und-Herrsche-Prinzip optimiert. Er
 * funktioniert so ähnlich wie der Mergesort.
 *
 * nach <a href=
 * "https://www.enjoyalgorithms.com/blog/find-the-minimum-and-maximum-
→ value-in-an-array">enjoyalgorithms.com</a>
 *
 * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
 *          gesucht werden soll.
 * @param l Die linke Grenze.
 * @param r Die rechts Grenze.
 *
 * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das
→ Minimum,
 *          der zweite Eintrag das Maximum.

```

```

*/
int[] minMaxRekursiv(int[] a, int l, int r) {
    int max, min;
    if (l == r) {
        max = a[l];
        min = a[l];
    } else if (l + 1 == r) {
        if (a[l] < a[r]) {
            max = a[r];
            min = a[l];
        } else {
            max = a[l];
            min = a[r];
        }
    } else {
        int mid = l + (r - l) / 2;
        int[] lErgebnis = minMaxRekursiv(a, l, mid);
        int[] rErgebnis = minMaxRekursiv(a, mid + 1, r);
        if (lErgebnis[0] > rErgebnis[0]) {
            max = lErgebnis[0];
        } else {
            max = rErgebnis[0];
        }
        if (lErgebnis[1] < rErgebnis[1]) {
            min = lErgebnis[1];
        } else {
            min = rErgebnis[1];
        }
    }
    int[] ergebnis = { max, min };
    return ergebnis;
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java)

46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 3

Wir betrachten eine Variante der Breitensuche (BFS), bei der die Knoten markiert werden, wenn sie das erste Mal besucht werden. Außerdem wird die Suche einmal bei jedem unmarkierten Knoten gestartet, bis alle Knoten markiert sind. Wir betrachten gerichtete Graphen. Ein gerichteter Graph G ist *schwach zusammenhängend*, wenn der ungerichtete Graph (der sich daraus ergibt, dass man die Kantenrichtungen von G ignoriert) zusammenhängend ist.

Exkurs: Schwach zusammenhängend gerichteter Graph

Beim gerichteten Graphen musst du auf die Kantenrichtung achten. Würde man die Richtungen der Kanten ignorieren wäre aber trotzdem jeder Knoten erreichbar. Einen solchen Graphen nennt man schwach zusammenhängend.^a

Ein gerichteter Graph heißt (schwach) zusammenhängend, falls der zugehörige ungerichtete Graph (also der Graph, der entsteht, wenn man jede gerichtete Kante durch eine ungerichtete Kante

ersetzt) zusammenhängend ist.^b

^a<https://studyflix.de/informatik/grundbegriffe-der-graphentheorie-1285>

^b[https://de.wikipedia.org/wiki/Zusammenhang_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Zusammenhang_(Graphentheorie))

- (a) Beschreiben Sie für ein allgemeines $n \in \mathbb{N}$ mit $n \geq 2$ den Aufbau eines schwach zusammenhängenden Graphen G_n , mit n Knoten, bei dem die Breitensuche $\Theta(n)$ mal gestartet werden muss, bis alle Knoten markiert sind.

Lösungsvorschlag

?

Die Breitensuche benötigt einen Startknoten. Die unten aufgeführten Graphen finden immer nur einen Knoten nämlich den Startknoten.

```

  graph LR
    A[A] --> B[B]
    B --> C[C]
    C --> D[D]
  
```

Oder so:

```

  graph TD
    B[B] --> A[A]
    D[D] --> A
    C[C] --> A
  
```

- (b) Welche asymptotische Laufzeit in Abhängigkeit von der Anzahl der Knoten (n) und von der Anzahl der Kanten (m) hat die Breitensuche über alle Neustarts zusammen? Beachten Sie, dass die Markierungen nicht gelöscht werden. Geben Sie die Laufzeit in Θ -Notation an. Begründen Sie Ihre Antwort.

46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 4

Eine Sondierfolge $s(k, i)$ liefert für einen Schlüssel k aus einem Universum U und Versuchsnummern $i = 0, 1, \dots, m-1$ eine Folge von Indizes für eine Hashtabelle $T[0 \dots m-1]$. Mithilfe einer Sondierfolge wird beim Hashing mit offener Adressierung z. B. beim Einfügen eines neuen Schlüssels k nach einem noch nicht benützten Tabelleneintrag gesucht. Seien h und h' zwei verschiedene Hash-funktionen, die U auf $0, 1, \dots, m-1$ abbilden. Beantworten Sie die folgenden Fragen und geben Sie an, um welche Art von Sondieren es sich jeweils handelt.

- (a) Was ist problematisch an der Sondierfolge $s(k, i) = (h(k) + 2i) \bmod m$, wobei $m = 1023$ die Größe der Hashtabelle ist?

Lösungsvorschlag

Art Es handelt sich um lineares Sondieren.

Problematisch Es wird für einen großen Bereich an Sondierfolgen (512 (0-511, 512-1023)) nur in jeden zweiten Bucket (z. B. geradzahlig) sondiert, erst dann wird in den bisher ausgelassenen Buckets (z. B. ungeradzahlig) sondiert.

- (b) Was ist problematisch an der Sondierfolge $s(k, i) = (h(k) + i(i+1)) \bmod m$, wobei $m = 1024$ die Größe der Hashtabelle ist?

Art Es handelt sich um quadratisches Sondieren

Problematisch $i(i+1)$ gibt immer eine gerade Zahl. Eine gerade Zahl Modulo 1024 gibt auch immer eine grade Zahl. Es wird nie in den ungeraden Buckets sondiert.

- (c) Was ist vorteilhaft an der Sondierfolge $s(k, i) = (h(k) + i \cdot h'(k)) \bmod m$, wobei m die Größe der Hashtabelle ist?

Lösungsvorschlag

Auch die Sondierfolge ist abhängig von dem Schlüsselwert. Die Entstehung von Ballungen ist unwahrscheinlicher bei gut gewählten Hashfunktionen, eine gleichmäßige Verteilung wahrscheinlicher.

- (d) Sei $h(k) = k \bmod 6$ und $h(k) = k^2 \bmod 6$

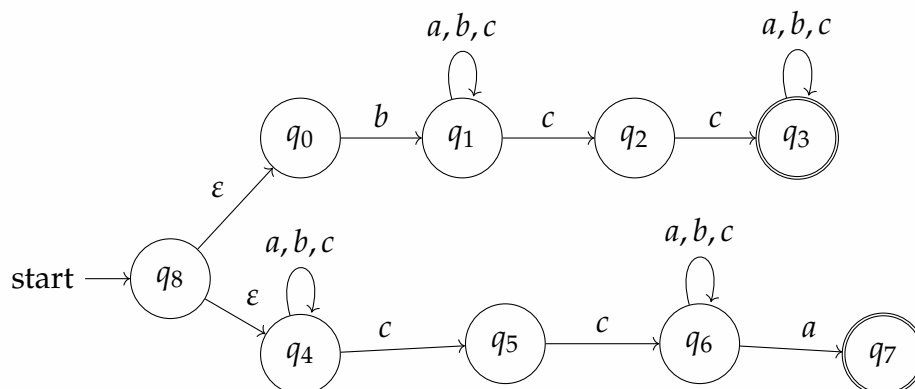
Fügen Sie die Schlüssel 14, 9, 8, 3, 2 in eine Hashtabelle der Größe 7 ein. Verwenden Sie die Sondierfolge $s(k, i) = (h(k) + i \cdot h'(k)) \bmod 7$ und offene Adressierung. Notieren Sie die Indizes der Tabellenfelder und vermerken Sie neben jedem Feld die erfolglosen Sondierungen.

46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1

- (a) Betrachten Sie die formale Sprache $L \subseteq a, b, c^*$: aller Wörter, die entweder mit b beginnen oder mit a enden (aber nicht beides gleichzeitig) und das Teilwort cc enthalten. Entwerfen Sie einen (vollständigen) deterministischen endlichen Automaten, der die Sprache L akzeptiert. (Hinweis: Es werden weniger als 10 Zustände benötigt.)

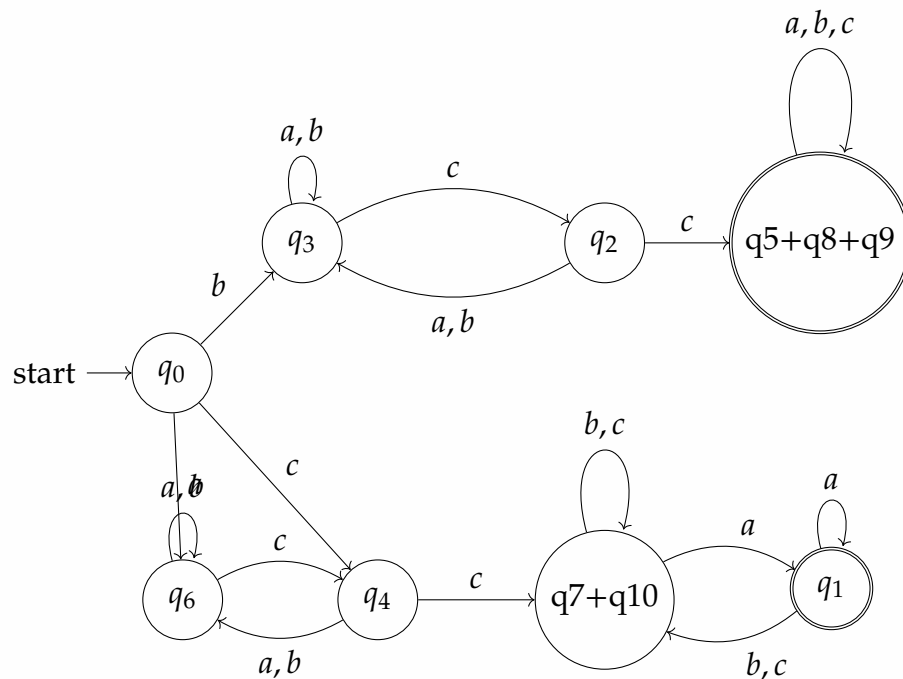
Lösungsvorschlag

NEA:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ar3pvv7ha

konvertierter DEA:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ai89m0txw

(b) Ist die folgende Aussage richtig? Begründen Sie Ihre Antwort.

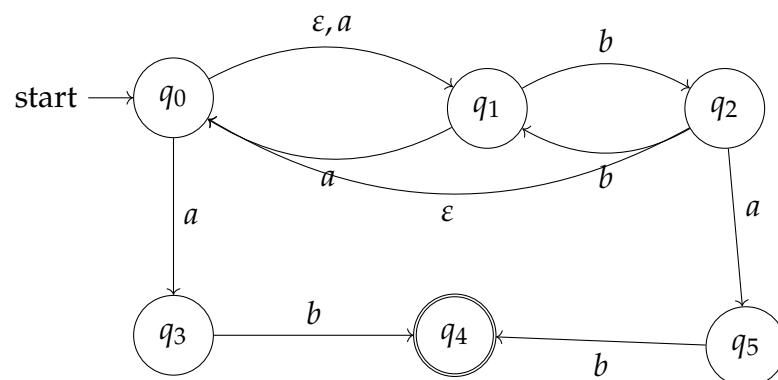
„Jede Teilsprache einer regulären Sprache ist regulär, d. h. für ein Alphabet und formale Sprachen $L' \subseteq L$ ist L' regulär, falls L regulär ist.“

Lösungsvorschlag

Ja. Reguläre Sprachen sind abgeschlossen unter dem Komplement und der Vereinigung

46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2

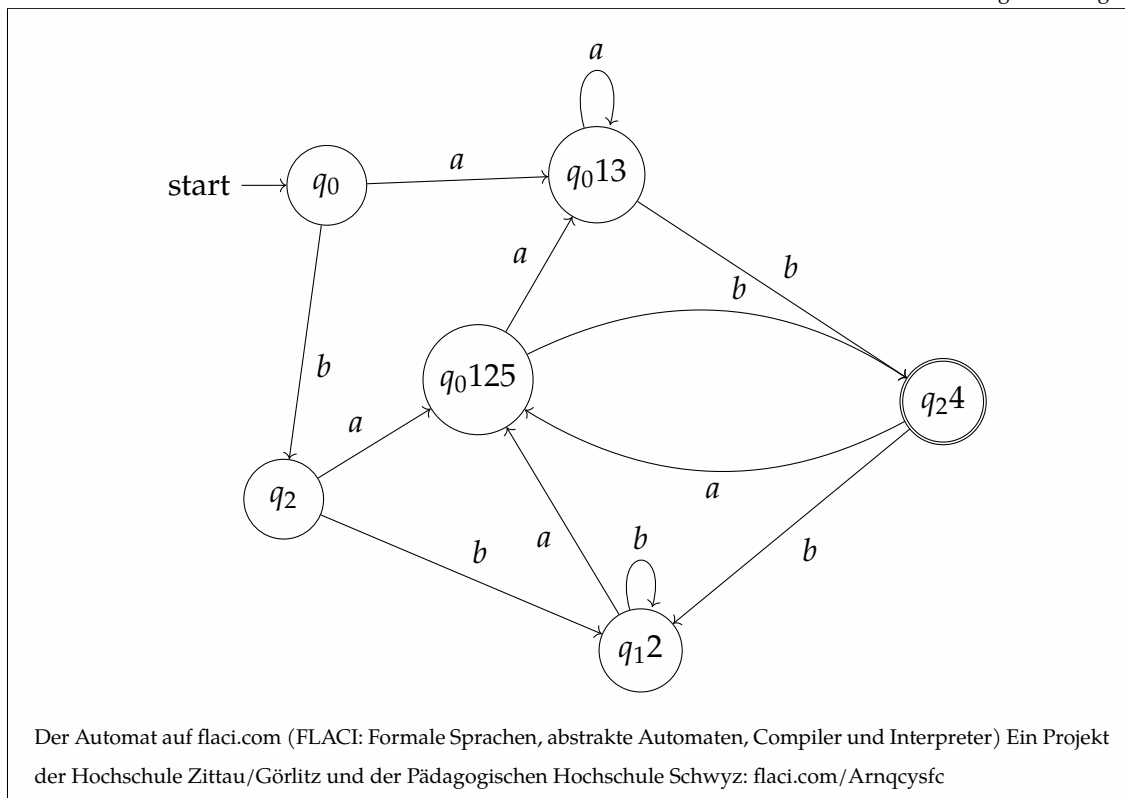
Gegeben sei der folgende e-nichtdeterministische endliche Automat A über dem Alphabet



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A54bbh2iz

- (a) Konstruieren Sie einen deterministischen endlichen Automaten für A. Wenden Sie dafür die Potenzmengenkonstruktion an.

Lösungsvorschlag



- (b) Beschreiben Sie die von A akzeptierte Sprache $L(A)$ mit eigenen Worten und so einfach wie möglich.

Lösungsvorschlag

Endet auf ab, Präfix beliebig auch leer

46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 3

Sei $\mathbb{N}_0 = \{0, 1, 2, \dots\}$ die Menge aller natürlichen Zahlen mit 0. Betrachten Sie die folgenden Sprachen.

- (a) $L_1 = \{a^{3n}b^{2n}a^n \mid n \in \mathbb{N}_0\}$

Lösungsvorschlag

nicht kontextfrei

- (b) $L_2 = \{a^{3n}a^{2n}b^n \mid n \in \mathbb{N}_0\}$

kontextfrei.

Der Ausdruck lässt umformen in: $L_2 = \{ a^{5n}b^n \mid n \in \mathbb{N}_0 \}$

$P = \{$

$$S \rightarrow aaaaaSb \mid \varepsilon$$

$\}$

(c) $L_3 = \{ (ab)^n a (ba)^n b (ab)^n \mid n \in \mathbb{N}_0 \}$

nicht kontextfrei

Geben Sie jeweils an, ob L_1 , L_2 und L_3 kontextfrei und ob L_1 , L_2 und L_3 regulär sind. Beweisen Sie Ihre Behauptung und ordnen Sie jede Sprache in die kleinstmögliche Klasse (regulär, kontextfrei, nicht kontextfrei) ein. Für eine Einordnung in kontextfrei zeigen Sie also, dass die Sprache kontextfrei und nicht regulär ist.

Erfolgt ein Beweis durch Angabe eines Automaten, so ist eine klare Beschreibung der Funktionsweise des Automaten und der Bedeutung der Zustände erforderlich. Erfolgt der Beweis durch Angabe eines regulären Ausdrucks, so ist eine intuitive Beschreibung erforderlich. Wird der Beweis durch die Angabe einer Grammatik geführt, so ist die Bedeutung der Variablen zu erläutern.

46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Sortieren Sie die unten angegebenen Funktionen der O-Klassen a, b, c, d und e bezüglich ihrer Teilmengenbeziehungen. Nutzen Sie ausschließlich die echte Teilmenge \subsetneq sowie die Gleichheit $=$ für die Beziehung zwischen den Mengen. Folgendes Beispiel illustriert diese Schreibweise für einige Funktionen f_1 bis f_5 . (Diese haben nichts mit den unten angegebenen Funktionen zu tun.)

$$f_4 \subsetneq f_3 = f_5 \subsetneq f_1 = f_2$$

Die angegebenen Beziehungen müssen weder bewiesen noch begründet werden.

- $a(n) = \sqrt{n^5} + 4n - 5$

- $b(n) = \log_2(\log_2(n))$

- $c(n) = 2^n$

- $d(n) = n^2 \log(n) + 2n$

- $e(n) = \frac{4^n}{\log_2 n}$

$$na(n)b(n)c(n)d(n)e(n)^5) + (4 * x) - 5; \log_2(\log_2(x)); 2^x; x^2 *$$

46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Gegeben sei die folgende Java-Implementierung eines Stacks.

```
class Stack {
    private Item head;

    public Stack() {
        head = null;
    }

    public void push(int val) {
        if (head == null) {
            head = new Item(val, null);
        } else {
            head = new Item(val, head);
        }
    }

    public int pop() {
        // ...
    }

    public int size() {
        // ...
    }

    public int min() {
        // ...
    }
}

class Item {
    private int val;
    private Item next;

    public Item(int val, Item next) {
        this.val = val;
        this.next = next;
    }
}
```

- (a) Implementieren Sie die Methode `pop` in einer objektorientierten Programmiersprache Ihrer Wahl, die das erste Item des Stacks entfernt und seinen Wert zurückgibt. Ist kein Wert im Stack enthalten, so soll dies mit einer `IndexOutOfBoundsException` oder Ähnlichem gemeldet werden.

Beschreiben Sie nun jeweils die notwendigen Änderungen an den bisherigen Implementierungen, die für die Realisierung der folgenden Methoden notwendig sind.

Lösungsvorschlag

```
public int pop() {
    if (head != null) {
        int val = head.val;
```

```

        size--;
        head = head.next;
        return val;
    } else {
        throw new IndexOutOfBoundsException("The stack is empty");
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java)

- (b) `size` gibt in Laufzeit 1 die Anzahl der enthaltenen Items zurück.

Lösungsvorschlag

```

public void push(int val) {
    if (head == null) {
        head = new Item(val, null);
    } else {
        head = new Item(val, head);
    }
    if (min > val) {
        min = val;
    }
    size++;
}

public int pop() {
    if (head != null) {
        int val = head.val;
        size--;
        head = head.next;
        return val;
    } else {
        throw new IndexOutOfBoundsException("The stack is empty");
    }
}

public int size() {
    return size;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java)

- (c) `min` gibt (zu jedem Zeitpunkt) in Laufzeit 1 den Wert des kleinsten Elements im Stack zurück.

Lösungsvorschlag

```

public void push(int val) {
    if (head == null) {
        head = new Item(val, null);
    } else {
        head = new Item(val, head);
    }
    if (min > val) {
        min = val;
    }
}

```

```

    }
    size++;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java)

```

public int min() {
    return min;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java)

Sie dürfen jeweils alle anderen angegebenen Methoden der Klasse verwenden, auch wenn Sie diese nicht implementiert haben. Sie können anstelle von objekt-orientiertem Quellcode auch eine informelle Beschreibung Ihrer Änderungen angeben.

46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 3

Gegeben ist ein aufsteigend sortiertes Array A von n ganzen Zahlen und eine ganze Zahl x . Es wird der Algorithmus BinarySearch betrachtet, der A effizient nach dem Wert x absucht. Ergebnis ist der Index i mit $x = A[i]$ oder NIL, falls $x \notin A$.

Funktion BinarySearch(int A, int r)

```

l = 1;
r = A.length;
while r ≥ l do
    if x < A[m] then
        r = m - 1;
    else if x = A[m] then
        return m;
    else
        l = m + 1;
    end
    return NIL;
end

```

- (a) Durchsuchen Sie das folgende Feld jeweils nach den in (i) bis (iii) angegebenen Werten mittels binärer Suche. Geben Sie für jede Iteration die Werte l, r, m und den betretenen if-Zweig an. Geben Sie zudem den Ergebnis-Index bzw. NIL an.

Index

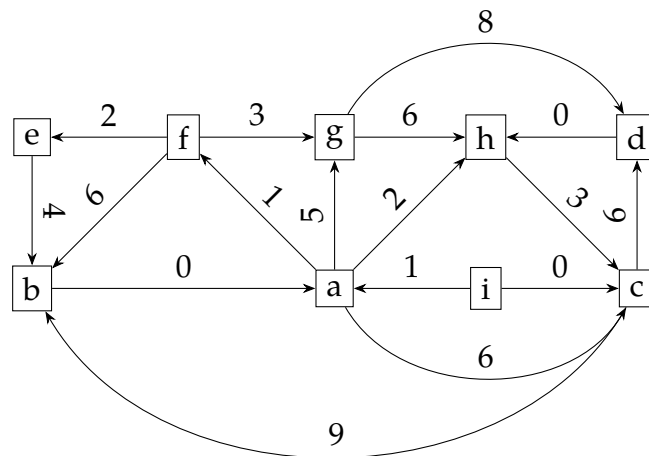
i]s] «| «| 2] 4] off

wen [ilsfol7] io] w]u]al ale!

- (i) 10
(ii) 13
(iii) 22
- (b) Betrachten Sie auf das Array aus Teilaufgabe a). Für welche Werte durchläuft der Algorithmus nie den letzten else-Teil in Zeile 11? Hinweis: Unterscheiden Sie auch zwischen enthaltenen und nicht-enhaltenen Werten.
- (c) Wie ändert sich das Ergebnis der binären Suche, wenn im sortierten Eingabefeld zwei aufeinanderfolgende, unterschiedliche Werte vertauscht wurden? Betrachten Sie hierbei die betroffenen Werte, die anderen Feldelemente und nicht enthaltene Werte in Abhängigkeit vom Ort der Vertauschung.
- (d) Angenommen, das Eingabearray A für den Algorithmus für die binäre Suche enthält nur die Zahlen 0 und 1, aufsteigend sortiert. Zudem ist jede der beiden Zahlen mindestens ein Mal vorhanden. Ändern Sie den Algorithmus für die binäre Suche so ab, dass er den bzw. einen Index k zurückgibt, für den gilt: $A[k] = 1$ und $A[k-1] = 0$.
- (e) Betrachten Sie die folgende rekursive Variante von BinarySearch.
- ```
1 int RekBinarySearch(int[] A, int x, int l, int r)
| mi
3 | (rekursive Implementierung)
```
- Der initiale Aufruf der rekursiven Variante lautet: `RekBinarySearch (A, z, 1, A.length)`

## 46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 4

- (a) Berechnen Sie im gegebenen gerichteten und gewichteten Graph  $G = (V, E, w)$  mit Kantenlängen  $w : E \rightarrow \mathbb{R}_0^+$  mittels des Dijkstra-Algorithmus die kürzesten (gerichteten) Pfade ausgehend vom Startknoten  $a$ .



Knoten, deren Entfernung von  $a$  bereits feststeht, seien als **schwarz** bezeichnet und Knoten, bei denen lediglich eine obere Schranke  $\neq \infty$  für ihre Entfernung von  $a$  bekannt ist, seien als *grau* bezeichnet.

- (i) Geben Sie als Lösung eine Tabelle an. Fügen Sie jedes mal, wenn der Algorithmus einen Knoten schwarz färbt, eine Zeile zu der Tabelle hinzu. Die Tabelle soll dabei zwei Spalten beinhalten: die linke Spalte zur Angabe des aktuell schwarz gewordenen Knotens und die rechte Spalte mit der bereits aktualisierten Menge grauer Knoten. Jeder Tabelleneintrag soll anstelle des nackten Knotennamens  $v$  ein Tripel  $(v, v.d, v.\pi)$  sein. Dabei steht  $v.d$  für die aktuell bekannte kürzeste Distanz zwischen  $a$  und  $v$ .  $v.\pi$  ist der direkte Vorgänger von  $v$  auf dem zugehörigen kürzesten Weg von  $a$ .

Lösungsvorschlag

| Nr.               | besucht       | a           | b        | c                                                 | d         | e        | f        | g        | h        | i        |
|-------------------|---------------|-------------|----------|---------------------------------------------------|-----------|----------|----------|----------|----------|----------|
| 0                 |               | 0           | $\infty$ | $\infty$                                          | $\infty$  | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| 1                 | a             | <b>0</b>    | $\infty$ | 6                                                 | $\infty$  | $\infty$ | 1        | 5        | 2        | $\infty$ |
| 2                 | f             |             | 7        | 6                                                 | $\infty$  | 3        | <b>1</b> | 4        | 2        | $\infty$ |
| 3                 | h             |             | 7        | 5                                                 | $\infty$  | 3        |          | 4        | <b>2</b> | $\infty$ |
| 4                 | e             |             | 7        | 5                                                 | $\infty$  | <b>3</b> |          | 4        |          | $\infty$ |
| 5                 | g             |             | 7        | 5                                                 | 12        |          |          | <b>4</b> |          | $\infty$ |
| 6                 | c             |             | 7        | <b>5</b>                                          | 11        |          |          |          |          | $\infty$ |
| 7                 | b             |             | <b>7</b> |                                                   | 11        |          |          |          |          | $\infty$ |
| 8                 | d             |             |          |                                                   | <b>11</b> |          |          |          |          | $\infty$ |
| 9                 | i             |             |          |                                                   |           |          |          |          |          | $\infty$ |
| nach              | Entfernung    | Reihenfolge |          | Pfad                                              |           |          |          |          |          |          |
| a $\rightarrow$ a | 0             | 1           |          |                                                   |           |          |          |          |          |          |
| a $\rightarrow$ b | 7             | 7           |          | a $\rightarrow$ f $\rightarrow$ b                 |           |          |          |          |          |          |
| a $\rightarrow$ c | 5             | 6           |          | a $\rightarrow$ h $\rightarrow$ c                 |           |          |          |          |          |          |
| a $\rightarrow$ d | 11            | 8           |          | a $\rightarrow$ h $\rightarrow$ c $\rightarrow$ d |           |          |          |          |          |          |
| a $\rightarrow$ e | 3             | 4           |          | a $\rightarrow$ f $\rightarrow$ e                 |           |          |          |          |          |          |
| a $\rightarrow$ f | 1             | 2           |          | a $\rightarrow$ f                                 |           |          |          |          |          |          |
| a $\rightarrow$ g | 4             | 5           |          | a $\rightarrow$ f $\rightarrow$ g                 |           |          |          |          |          |          |
| a $\rightarrow$ h | 2             | 3           |          | a $\rightarrow$ h                                 |           |          |          |          |          |          |
| a $\rightarrow$ i | 2.147483647E9 | 9           |          | a $\rightarrow$ i                                 |           |          |          |          |          |          |

- (ii) Zeichnen Sie zudem den entstandenen Kürzeste-Pfade-Baum.



- (b) Warum berechnet der Dijkstra-Algorithmus auf einem gerichteten Eingabegraphen mit potentiell auch negativen Kantengewichten  $w : E \rightarrow \mathbb{R}$  nicht immer einen korrekten Kürzesten-Wege-Baum von einem gewählten Startknoten aus? Geben Sie ein Beispiel an, für das der Algorithmus die falsche Antwort liefert. <sup>SQL</sup>
- (c) Begründen Sie, warum das Problem nicht gelöst werden kann, indem der Betrag des niedrigsten (also des betragsmäßig größten negativen) Kantengewichts im Graphen zu allen Kanten addiert wird.

## 46116 (Softwaretechnologie / Datenbanksysteme (nicht vertieft))

46116 / 2012 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3

### Aufgabe 3

Gegeben ist folgendes einfache Datenbankschema zur Personalverwaltung (Schlüsselattribute unterstrichen, Fremdschlüssel kursiv): Angestellter (PersNr, Name, Gehalt, Beruf, *AbtNr*, Ort)

Abteilung (AbtNr, Name, Ort)

Formulieren Sie folgende Datenbankoperationen in SQL:

- (a) Welche Angestellten sind von Beruf Koch und verdienen mehr als 5000 €?

Lösungsvorschlag

```
SELECT name
FROM Angestellter
WHERE Beruf = 'Koch' AND Gehalt > 5000;
```

- (b) Welche Angestellten der Abteilung B17 sind aus München?

Lösungsvorschlag

```
SELECT Name
FROM Angestellter, Abteilung
WHERE
 Angestellter.AbtNr = Abteilung.AbtNr AND
 Abteilung.Name = 'B17' AND
 Angestellter.Ort = 'München';
```

- (c) Hans Meier aus der Abteilung C4 zieht von München nach Erlangen um.

Lösungsvorschlag

```
UPDATE Angestellter SET Ort = "Erlangen"
WHERE Name = "Hans Meier" AND (
 SELECT ab.AbtNr
 FROM Abteilung ab
```

```
WHERE ab.Name = "C4"
) = AbtNr;
```

Klassendiagramm

- (d) Abteilung C4 wird aufgelöst.
- (e) Formulieren Sie folgende SQL-Anfrage umgangssprachlich und so exakt wie möglich.

```
SELECT AbtNr
FROM Abteilung a, (
 SELECT PersNr, Name, AbtNr
 FROM Angestellter
 WHERE Gehalt < 2000) t
WHERE
a.AbtNr = t.AbtNr AND a.Ort = "Nuernberg";
```

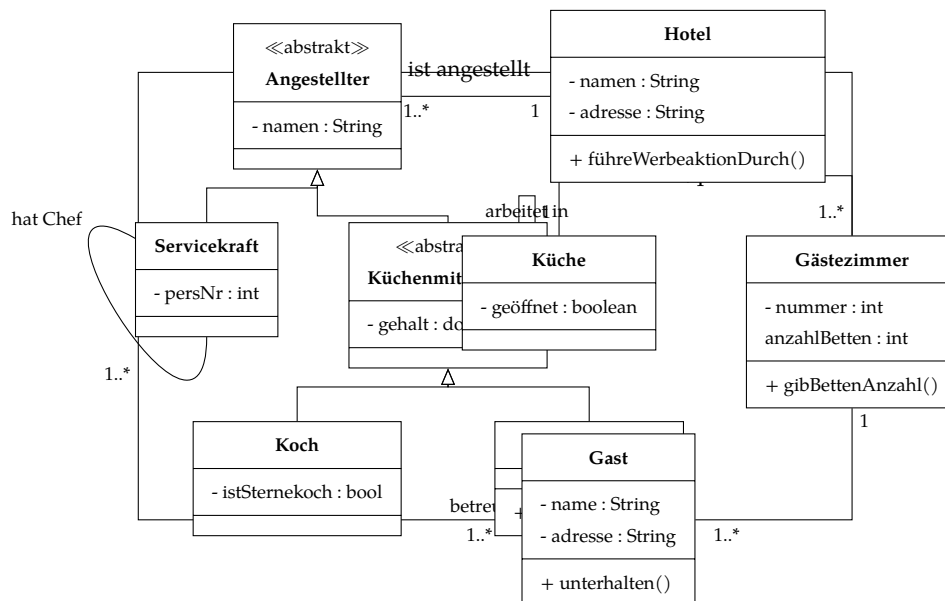
## 46116 / 2012 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1

### Hotel-Verwaltung

Erstellen Sie zu der folgenden Beschreibung eines Systems zur Organisation eines Hotels ein Klassendiagramm, das Attribute und Assoziationen mit Kardinalitäten sowie Methoden enthält. Setzen Sie dabei das Konzept der Vererbung sinnvoll ein.

- Ein **Hotel** besteht aus genau einer Küche und mehreren Gästezimmern.
- Es hat einen Namen, eine Adresse und kann Werbeaktionen durchführen.
- Alle Mitarbeiter, sowohl Servicekräfte als auch Küchenmitarbeiter, sind Angestellte des Hotels.
- Die Küche kann geöffnet oder geschlossen sein.
- Gästezimmer haben eine Nummer und eine bestimmte Anzahl an Betten, die ausgegeben werden kann.
- In diesen Gästezimmern wohnen Gäste, die von einer oder mehreren Servicekräften umsorgt werden.
- Servicekräfte sind nur für die ihnen zugeordneten Gästezimmer verantwortlich.
- Jede Servicekraft hat einen Namen und eine Personalnummer sowie einen Vorgesetzten, der auch eine Servicekraft ist.
- In der Küche arbeiten Küchenmitarbeiter, die einen Namen haben und ein Gehalt bekommen.
- Küchenmitarbeiter sind entweder Köche oder Aushilfen. Köche können zudem Sterneköche sein, also mit einem oder mehreren Sternen dekoriert sein.
- Aushilfen bauen die Buffets für die Mahlzeiten auf.

- Gäste können sich unterhalten. Jeder Gast hat einen Namen und eine Adresse und ist seinem Zimmer zugeordnet.



## 46116 / 2013 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Erläutern Sie in jeweils ca. 3 Sätzen die folgenden Begriffe aus der modernen Software-technologie: *Pair Programming*, *Refactoring*, *agile Softwareentwicklung*.

Lösungsvorschlag

**Pair Programming** Beim Pair Programming entwickeln zwei Personen die Software zusammen. Es gibt einen sogenannten Driver - die Person, die tippt - und einen sogenannten Navigator - die Person, die kritisch den Code überprüft. Das Pair Programming ist wichtiger in agilen Entwicklungsmethoden wie z. B dem Extreme Programming (XP).

**Refactoring** Unter Refactoring versteht man die Verbesserung der Code- und Systemstruktur mit dem Ziel einer besseren Wartbarkeit. Dabei sollen Lesbarkeit, Verständlichkeit, Wartbarkeit und Erweiterbarkeit verbessert werden, mit dem Ziel, den jeweiligen Aufwand für Fehleranalyse und funktionale Erweiterungen deutlich zu senken. Im Refactoring werden keine neuen Funktionalitäten programmiert.

**agile Softwareentwicklung** Agile Entwicklungsprozesse gehen auf das Agile Manifest, das im Jahr 2001 veröffentlicht wurde, zurück. Dieses Agile Manifest bildet die Basis für agile Entwicklungsprozesse wie eXtreme Programming oder SCRUM. Das Manifest beinhaltet 12 Grundprinzipien für die moderne agile Software-Entwicklung. Hauptzielsetzung ist, rasch eine funktionsfähige Software-Lösung auszuliefern, um den Kundenwunsch bestmöglich zu erfüllen. Dabei spielt die Interaktion mit dem Kunden eine zentrale Rolle.

**46116 / 2013 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 1**

Sie sollen ein System zur Verwaltung von Pferderennen entwerfen. Gehen Sie dabei von folgendem Szenario aus:

- **Unternehmen** werden ihre eindeutige *Unternehmens-ID* identifiziert. Sie haben eine *Adresse* und besitzen Rennställe.
 

☐ E: Unternehmen  
☐ A: Unternehmens-ID  
☐ A: Adresse  
☒ R: besitzen  
☐ A: Name  
☐ E: Rennstalls  
☐ A: Gründungsdatum  
☐ E: Pferde  
☒ R: gehören
- Der *Name* eines **Rennstalls** ist nur innerhalb eines Unternehmens eindeutig. Für jeden Rennstall wird das *Gründungsdatum* gespeichert.
 

☐ A: Name  
☐ E: Rennstalls  
☐ A: Gründungsdatum  
☐ E: Pferde  
☒ R: gehören
- **Pferde** gehören immer zu einem Rennstall. *Pferdenamen* werden in einem Rennstall nur jeweils maximal einmal vergeben.
 

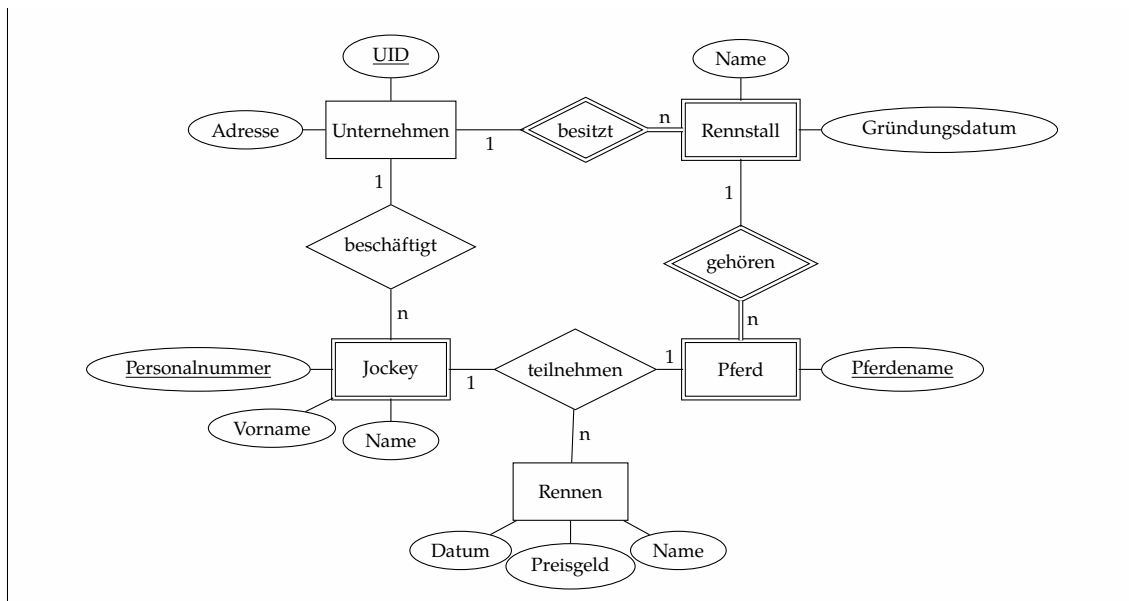
☐ A: Pferdenamen  
☐ E: Jockeys  
☒ R: beschäftigt
- **Jockeys** sind in einem Rennstall beschäftigt. Jeder Rennstall vergibt seine eigenen *Personalnummern*. Für jeden Jockey werden *Vorname* und *Name* gespeichert.
 

☐ A: Personalnummern  
☐ A: Vorname  
☐ A: Name  
☐ E: Rennen  
☐ A: Datum  
☐ A: Preisgeld  
☐ A: Namen  
☒ R: unterstützen  
☒ R: nehmen
- **Rennen** haben ein *Datum*, ein *Preisgeld* und einen *Namen*, über den sie identifiziert werden.
- Unternehmen unterstützen Rennen finanziell mit einem bestimmten Betrag.
- Jockeys nehmen mit Pferden an Rennen teil. Im Rennen erreichen sie einen bestimmten Platz. Die Kombination aus Jockey und Pferd ist nicht fest, bei unterschiedlichen Rennen können Jockeys verschiedene Pferde reiten. Jockeys können auch mit Rennpferden von fremden Rennställen, die anderen Unternehmen gehören können, an Rennen teilnehmen.

(a) Entwerfen Sie für das beschriebene Szenario ein ER-Modell. Bestimmen Sie hierzu:

- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- ein passendes ER-Diagramm,
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
- die Funktionalitäten der Relationship-Typen, welche Sie ebenfalls in das ER-Diagramm eintragen.

Lösungsvorschlag



- (b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

```

Unternehmen (UnternehmensID, Adresse)
Rennstall (S_Name, UnternehmensID[Unternehmen], Gründungsdatum)
Jockey (PersNr, S_Name[Rennstall], ID[Unternehmen], Vorname, Name)
Rennen (R_Name, Datum, Preisgeld)
Pferd (P_Name, S_Namen, UnternehmensID[Unternehmen])

teilnehmen (R_Name[Rennen], PersNr[Jockey], S_Name1, ID1, P_Namen, S_Namen2,
 ↳ ID2, Platz)
unterstuetzen (UnternehmensID[Unternehmen], R_Name[Rennen], Betrag)

```

## 46116 / 2013 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2

### 2. Anfragen

Zu einer Website, auf der Besucher im Browser Online-Spiele spielen können, liegt das folgende relationale Schema einer Datenbank vor:

```

Team : {[TNR, Name, Teamfarbe]}
Spieler : {[SNr, Name, Icon, TNr, EMail]}
Minispiel : {[MNR, Name, Kategorie, Schwierigkeit]}
Wettkampf : {[WNR, Sieger, Geschlagener, MNR, Dauer]}

```

Auf der Website treten jeweils zwei Spieler gegeneinander in Minispielen an. In diesen ist es das Ziel, den gegnerischen Spieler in möglichst kurzer Zeit zu besiegen. Minispiele gibt es dabei in verschiedenen Schwierigkeitsstufen („leicht“, „mittel“, „schwer“,

„sehr schwer“) und verschiedenen Kategorien („Denkspiel“, „Geschicklichkeitsspiel“, usw.). Die Attribute *Sieger* und *Geschlagener* sind jeweils Fremdschlüsselattribute, die auf das Attribut *SNr* der Relation *Spieler* verweisen. Beachten Sie, dass das Dauer-Attribut der Wettkampf-Relation die Dauer eines Wettkampfes in der Einheit Sekunden speichert.

- (a) Formulieren Sie geeignete Anfragen in relationaler Algebra für die folgenden Teilaufgaben:

- (i) Geben Sie die *Namen* der *Minispiele* zurück, die zur *Kategorie* „Denkspiele“ zählen.

Lösungsvorschlag

$$\pi_{\text{Name}}(\sigma_{\text{Kategorie}='Denkspiele'}(\text{Minispiele}))$$

- (ii) Geben Sie die *Namen* und *E-Mail-Adressen* aller Spieler zurück, die in einem Minispiel des Typs „Geschicklichkeitsspiel“ gewonnen haben.

Lösungsvorschlag

$$\pi_{\text{Spieler.Name}, \text{Spieler.Email}}(\sigma_{\text{Kategorie}='Geschicklichkeitsspiel'}(\text{Minispiele}) \bowtie_{\text{Minispiel.MNr}=\text{Wettkampf.MNr}} \text{Wettkampf} \bowtie_{\text{Wettkampf.Sieger}=\text{Spieler.SNr}} \text{Spieler})$$

- (b) Formulieren Sie geeignete SQL-Anfragen für die folgenden Teilaufgaben. Beachten Sie dabei, dass Ihre Ergebnisrelationen keine Duplikate enthalten sollen.

- (i) Geben Sie jede Spielekategorie aus, für die ein Minispiel der Schwierigkeitsstufe sehr schwer vorhanden ist.

Lösungsvorschlag

```
SELECT DISTINCT Kategorie
FROM Minispiel
WHERE Schwierigkeit = 'sehr schwer';
```

- (ii) Geben Sie die Wettkämpfe aus, deren Dauer unter der durchschnittlichen Dauer der Wettkämpfe liegt.

Lösungsvorschlag

```
SELECT WNr
FROM Wettkampf
WHERE Dauer < (
 SELECT AVG(Dauer) FROM Wettkampf
);
```

- (iii) Geben Sie für jeden *Spieler* seine *SNr*, seinen *Namen*, die Anzahl seiner Siege, die durchschnittliche Dauer seiner siegreichen Wettkämpfe und die Anzahl der Teams, aus denen er bereits mindestens einen Spieler besiegt hat, zurück.

```

SELECT
 SNr,
 Name,
 (
 SELECT COUNT(*)
 FROM Wettkampf
 WHERE Wettkampf.Sieger = SNr
) AS Anzahl_Siege,
 (
 SELECT AVG(Dauer)
 FROM Wettkampf
 WHERE Wettkampf.Sieger = SNr
),
 (
 SELECT COUNT(DISTINCT Team.TNr)
 FROM Team, Spieler, Wettkampf
 WHERE
 Team.TNr = Spieler.TNr AND
 Wettkampf.Geschlagener = Spieler.SNr AND
 Wettkampf.Sieger = SNr
)
FROM Spieler;

```

Lösungsvorschlag

```

SELECT
 s.SNr,
 s.Name,
 COUNT(*) AS AnzahlSiege,
 AVG(w.Dauer) AS DurchschnittlicheWettkampfdauer,
 COUNT(DISTINCT g.TNr) AS TeamsBesiegt
FROM Spieler s, Wettkampf w, Spieler g
WHERE
 s.SNr = w.Sieger AND
 g.SNr = w.Geschlagener
GROUP BY s.SNr, s.Name;

```

(c) Verwenden Sie den relationalen Tupelkalkül, um die folgenden Anfragen zu formulieren:

- (i) Finden Sie die Namen der Spieler des Teams Dream Team.
- (ii) Geben Sie die Namen der Minispiele zurück, bei denen bereits Spieler gegeneinander angetreten sind, deren Teams dieselbe Teamfarbe haben.

Hinweis: Die Anfragen im relationalen Tupelkalkül dürfen auch nicht sicher sein.

**46116 / 2013 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1**

## Aufgabe 1

Gegeben sei folgendes Klassendiagramm:

- (a) Implementieren Sie die Klassen „Order“, „Payment“, „Check“, „OrderDetail“ und „Item“ in einer geeigneten objektorientierten Sprache Ihrer Wahl. Beachten Sie dabei insbesondere Sichtbarkeiten, Klassen- vs. Instanzzugehörigkeiten und Schnittstellen bzw. abstrakte Klassen.

Lösungsvorschlag

```
public class Check extends Payment {
 String bankName;
 long bankId;
 public boolean authorized() {
 return true;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2013/fruehjahr/t2\\_ta1\\_a1/Check.java](#)

```
@SuppressWarnings("unused")
public class Item {
 private double weight;
 public String description;

 public double getPrice() {
 return 42;
 }

 public double getWeight() {
 return 23;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2013/fruehjahr/t2\\_ta1\\_a1/Item.java](#)

```
import java.util.Date;

public class Order {
 public static double VAT = 0.19;

 protected Date date;

 protected boolean shipped;

 public double calcPrice() {
 return 0.1d;
 }

 public double calcWeight() {
 return 0.2d;
 }

 public double calcTax(double tax) {
 return 0.3d;
 }
}
```



Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2013/fruehjahr/t2\\_ta1\\_a1/Order.java](#)

```
@SuppressWarnings("unused")
public class OrderDetail {
 private long quantity;

 Item item;

 public double calcPrice() {
 return 0.1d;
 }

 public double calcWeight() {
 return 0.2d;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2013/fruehjahr/t2\\_ta1\\_a1/OrderDetail.java](#)

```
public abstract class Payment {
 double amount;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2013/fruehjahr/t2\\_ta1\\_a1/Payment.java](#)

- (b) Erstellen Sie ein Sequenzdiagramm (mit konkreten Methodenaufrufen und Nummerierung der Nachrichten) für folgendes Szenario:
- (i) „Erika Mustermann“ aus „Rathausstraße 1, 10178 Berlin“ wird als neue Kundin angelegt.
  - (ii) Frau Mustermann bestellt heute 1 kg Gurken und 2 kg Kartoffeln.
  - (iii) Sie bezahlt mit ihrer Visa-Karte, die im August 2014 abläuft und die Nummer „1234 567891 23456“ hat — die Karte erweist sich bei der Prüfung als gültig.
  - (iv) Am Schluss möchte sie noch wissen, wie viel ihre Bestellung kostet — dabei wird der Anteil der Mehrwertsteuer extra ausgewiesen.

## 46116 / 2014 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1

Gegeben sei folgende Methode zur Berechnung der Anzahl der notwendigen Züge beim Spiel „Die Türme von Hanoi“:

```
int hanoi(int nr, char from, char to) {
 char free = (char) ('A' + 'B' + 'C' - from - to);
 if (nr > 0) {
 int moves = 1;
 moves += hanoi(nr - 1, from, free);
 System.out.println("Move piece nr. " + nr + " from " + from + " to " + to);
 moves += hanoi(nr - 1, free, to);
 return moves;
 }
}
```

```
 } else {
 return 0;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2014/fruehjahr/Hanoi.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_46116/jahr_2014/fruehjahr/Hanoi.java)

- (a) Beweisen Sie formal mittels vollständiger Induktion, dass zum Umlegen von  $k$  Scheiben (z. B. vom Turm A zum Turm C) insgesamt  $2^k - 1$  Schritte notwendig sind, also dass für  $k \geq 0$  folgender Zusammenhang gilt:

$$\text{hanoi}(k, 'A', 'C') = 2^k - 1$$

Zu zeigen:

$$\text{hanoi}(k, 'A', 'C') = 2^k - 1$$

### Induktionsanfang

— Beweise, dass  $A(1)$  eine wahre Aussage ist. \_\_\_\_\_

$$k = 0$$

$$\text{hanoi}(0, 'A', 'C') = 0$$

$$2^0 - 1 = 1 - 1 = 0$$

### Induktionsvoraussetzung

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . \_\_\_\_\_

$$\text{hanoi}(k, 'A', 'C') = 2^k - 1$$

### Induktionsschritt

— Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss.

$$\text{hanoi}(k, 'A', 'C') = 1 + \text{hanoi}(k - 1, 'A', 'B') + \text{hanoi}(k - 1, 'B', 'C')$$

$$k \rightarrow k + 1$$

$$\begin{aligned}
\text{hanoi}(k+1, 'A', 'C') &= 1 + \text{hanoi}((k+1) - 1, 'A', 'B') + \\
&\quad \text{hanoi}((k+1) - 1, 'B', 'C') \\
&= 1 + \text{hanoi}(k, 'A', 'B') + \\
&\quad \text{hanoi}(k, 'B', 'C') && k+1-1=k \\
&= 1 + 2^k - 1 + 2^k - 1 && \text{Formeln eingesetzt} \\
&= 2^k + 2^k - 1 && 1-1-1=-1 \\
&= 2 \cdot 2^k - 1 && 2^k + 2^k = 2 \cdot 2^k \\
&= 2^{k+1} - 1 && 2 \cdot 2^k = 2^{k+1}
\end{aligned}$$

- (b) Geben Sie eine geeignete Terminierungsfunktion an und begründen Sie kurz Ihre Wahl!

Lösungsvorschlag

Betrachte die Argumentenfolge  $k, k-1, k-2, \dots, 0$ . Die Terminierungsfunktion ist offenbar  $T(k) = k$ .  $T(k)$  ist bei jedem Rekursionsschritt auf der Folge der Argumente streng monoton fallend. Bei der impliziten Annahme  $k$  ist ganzzahlig und  $k \geq 0$  ist  $T(k)$  nach unten durch 0 beschränkt.

## 46116 / 2014 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 2

### Aufgabe 2: Relationale Algebra

Gegeben sei das folgende relationale Schema mitsamt Beispieldaten für eine Datenbank von Mitfahrgelegenheiten. Die Primärschlüssel-Attribute sind jeweils unterstrichen, Fremdschlüssel sind überstrichen.

„Kunde“:

| <u>KID</u> | Name    | Vorname   | <u>Stadt</u> |
|------------|---------|-----------|--------------|
| K1         | Meier   | Stefan    | S3           |
| K2         | Müller  | Peta      | S3           |
| K3         | Schmidt | Christine | S2           |
| K4         | Schulz  | Michael   | S4           |

„Stadt“

„Angebot“:

| <u>SID</u> | SName     | Bundesland          |
|------------|-----------|---------------------|
| S1         | Berlin    | Berlin              |
| S2         | Nürn      | Bayern              |
| S3         | Köln      | Nordrhein-Westfalen |
| S4         | Stuttgart | Baden-Württemberg   |
| S5         | München   | Bayer               |

| <u>KID</u> | <u>Start</u> | <u>Ziel</u> | <u>Datum</u> | Plätze |
|------------|--------------|-------------|--------------|--------|
| K4         | S4           | S5          | 08.07.2011   | 3      |
| K4         | S5           | S4          | 10.07.2011   | 3      |
| K1         | S1           | S5          | 08.07.2011   | 3      |
| K3         | S2           | S3          | 15.07.2011   | 1      |
| K4         | S4           | S1          | 15.07.2011   | 3      |
| K1         | S5           | S5          | 09.07.2011   | 2      |

„Anfrage“:

| <u>KID</u> | <u>Start</u> | <u>Ziel</u> | <u>Datum</u> |
|------------|--------------|-------------|--------------|
| K2         | S4           | S5          | 08.07.2011   |
| K2         | S5           | S4          | 10.07.2011   |
| K3         | S2           | S3          | 08.07.2011   |
| K3         | S3           | S2          | 10.07.2011   |
| K2         | S4           | S5          | 05.07.2011   |
| K2         | S5           | S4          | 17.07.2011   |

(a) Formulieren Sie die folgenden Anfragen auf das gegebene Schema in relationaler Algebra:

- Finden Sie die Namen aller Städte in Bayern!

Lösungsvorschlag

$$\pi_{\text{SName}}(\sigma_{\text{Bundesland}=\text{Bayern}}(\text{Stadt}))$$

- Finden Sie die SIDs aller Städte, für die weder als Start noch als Ziel eine Anfrage vorliegt!

Lösungsvorschlag

$$\pi_{\text{SID}}(\text{Stadt}) - \pi_{\text{Start}}(\text{Anfrage}) - \pi_{\text{Ziel}}(\text{Anfrage})$$

- Finden Sie alle IDs von Kunden, welche eine Fahrt in ihrer Heimatstadt starten.

Lösungsvorschlag

$$\pi_{\text{KID}}(\text{Kunde} \bowtie_{\text{Kunde.KID}=\text{Anfrage.KID} \wedge \text{Kunde.Stadt}=\text{Anfrage.Stadt}} \text{Anfrage})$$

$$\wedge$$

$$\pi_{\text{KID}}(\text{Kunde} \bowtie_{\text{Kunde.KID}=\text{Angebot.KID} \wedge \text{Kunde.Stadt}=\text{Angebot.Stadt}} \text{Angebot})$$

- Geben Sie das Datum aller angebotenen Fahrten von München nach Stuttgart aus!

Lösungsvorschlag

$$\pi_{\text{Datum}}((\text{Angebot} \bowtie_{\text{Start}=\text{SID} \wedge \text{SName}=\text{'München'}} \text{Stadt}) \bowtie_{\text{Ziel}=\text{SID} \wedge \text{SName}=\text{'Stuttgart'}} \text{Stadt})$$

## Variante 2:

Lösungsvorschlag

$$\pi_{\text{Datum}}(\sigma_{\text{Sname}='München' \wedge \text{Zname}='Stuttgart'}(\rho_{\text{Zname} \leftarrow \text{Sname}, \text{SID1} \leftarrow \text{SID}}(\text{Stadt}) \bowtie_{\text{Ziel}=\text{SID1}} \text{Angebot} \bowtie_{\text{Start}=\text{SID}} \text{Stadt}))$$

(b) Geben Sie das Ergebnis (bezüglich der Beispieldaten) der folgenden Ausdrücke der relationalen Algebra als Tabellen an:

-  $\pi_{\text{KID}}(\text{Angebot}) \bowtie \text{Kunde}$

Lösungsvorschlag

Zeile mit der Petra Müller fällt weg.

| <u>KID</u> | Name    | Vorname   | <u>Stadt</u> |
|------------|---------|-----------|--------------|
| K1         | Meier   | Stefan    | S3           |
| K3         | Schmidt | Christine | S2           |
| K4         | Schulz  | Michael   | S4           |

-  $\pi_{(\text{KID}, \text{Stadt})}(\text{Kunde}) \bowtie_{\text{Kunde.Stadt}=\text{Angebot.Ziel}} \pi_{\text{Plaetze}}(\text{Angebot})$

Lösungsvorschlag

| KID | Stadt | Plätze |
|-----|-------|--------|
| K1  | S3    | 1      |
| K2  | S3    | 1      |
| K4  | S4    | 1      |
| K4  | S4    | 2      |

## 46116 / 2014 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 3

„Kunde“:

| <u>KID</u> | Name    | Vorname   | <u>Stadt</u> |
|------------|---------|-----------|--------------|
| K1         | Meier   | Stefan    | S3           |
| K2         | Müller  | Peta      | S3           |
| K3         | Schmidt | Christine | S2           |
| K4         | Schulz  | Michael   | S4           |

„Stadt“

| <u>SID</u> | SName     | Bundesland          |
|------------|-----------|---------------------|
| S1         | Berlin    | Berlin              |
| S2         | Nürnberg  | Bayern              |
| S3         | Köln      | Nordrhein-Westfalen |
| S4         | Stuttgart | Baden-Württemberg   |
| S5         | München   | Bayern              |

„Angebot“:

| <u>KID</u> | <u>Start</u> | <u>Ziel</u> | <u>Datum</u> | Plätze |
|------------|--------------|-------------|--------------|--------|
| K4         | S4           | S5          | 08.07.2011   | 3      |
| K4         | S5           | S4          | 10.07.2011   | 3      |
| K1         | S1           | S5          | 08.07.2011   | 3      |
| K3         | S2           | S3          | 15.07.2011   | 1      |
| K4         | S4           | S1          | 15.07.2011   | 3      |
| K1         | S5           | S5          | 09.07.2011   | 2      |

„Anfrage“:

| <u>KID</u> | <u>Start</u> | <u>Ziel</u> | <u>Datum</u> |
|------------|--------------|-------------|--------------|
| K2         | S4           | S5          | 08.07.2011   |
| K2         | S5           | S4          | 10.07.2011   |
| K3         | S2           | S3          | 08.07.2011   |
| K3         | S3           | S2          | 10.07.2011   |
| K2         | S4           | S5          | 05.07.2011   |
| K2         | S5           | S4          | 17.07.2011   |

```
CREATE TABLE Stadt (
 SID VARCHAR(100) NOT NULL PRIMARY KEY,
 SName VARCHAR(100) NOT NULL,
 Bundesland VARCHAR(100) NOT NULL
);
```

```
CREATE TABLE Anfrage (
 KID VARCHAR(100) NOT NULL,
 Start VARCHAR(100) DEFAULT NULL REFERENCES Stadt (SID),
 Ziel VARCHAR(100) DEFAULT NULL REFERENCES Stadt (SID),
 Datum date NOT NULL,
 PRIMARY KEY (KID, Start, Ziel, Datum)
);
```

```
CREATE TABLE Angebot (
 KID VARCHAR(100) NOT NULL,
 Start VARCHAR(100) DEFAULT NULL REFERENCES Stadt (SID),
 Ziel VARCHAR(100) DEFAULT NULL REFERENCES Stadt (SID),
 Datum date NOT NULL,
 Plätze integer DEFAULT NULL,
 PRIMARY KEY (Datum, KID)
);
```

```
CREATE TABLE Kunde (
 KID VARCHAR(100) NOT NULL PRIMARY KEY,
 Name VARCHAR(100) DEFAULT NULL,
 Vorname VARCHAR(100) DEFAULT NULL,
 Stadt VARCHAR(100) DEFAULT NULL REFERENCES Stadt (SID)
);
```

```
INSERT INTO Stadt (SID, SName, Bundesland) VALUES
('S1', 'Berlin', 'Berlin'),
('S2', 'Nürnberg', 'Bayern'),
```

```
('S3', 'Köln', 'NRW'),
('S4', 'Stuttgart', 'BW'),
('S5', 'München', 'Bayern');
```

```
INSERT INTO Anfrage (KID, Start, Ziel, Datum) VALUES
```

```
('K2', 'S4', 'S5', '2011-07-05'),
('K2', 'S4', 'S5', '2011-07-08'),
('K3', 'S2', 'S3', '2011-07-08'),
('K2', 'S5', 'S4', '2011-07-10'),
('K3', 'S3', 'S2', '2011-07-10'),
('K2', 'S5', 'S4', '2011-07-17');
```

```
INSERT INTO Kunde (KID, Name, Vorname, Stadt) VALUES
```

```
('K1', 'Meier', 'Stefan', 'S3'),
('K2', 'Müller', 'Petra', 'S3'),
('K3', 'Schmidt', 'Christine', 'S2'),
('K4', 'Schulz', 'Michael', 'S4');
```

```
INSERT INTO Angebot (KID, Start, Ziel, Datum, Plätze) VALUES
```

```
('K1', 'S1', 'S5', '2011-07-08', 3),
('K4', 'S4', 'S5', '2011-07-08', 3),
('K1', 'S5', 'S4', '2011-07-09', 2),
('K4', 'S5', 'S4', '2011-07-10', 3),
('K3', 'S2', 'S3', '2011-07-15', 1),
('K4', 'S4', 'S1', '2011-07-15', 3);
```

(a) Formulieren Sie die folgenden Anfragen in SQL:

- (i) Geben Sie alle Attribute aller Anfragen aus, für die passende Angebote existieren! Ein Angebot ist passend zu einer Anfrage, wenn Start, Ziel und Datum identisch sind!

Lösungsvorschlag

```
SELECT Anfrage.KID, Anfrage.Start, Anfrage.Ziel, Anfrage.Datum
FROM Anfrage, Angebot
WHERE
 Anfrage.Start = Angebot.Start AND
 Anfrage.Ziel = Angebot.Ziel AND
 Anfrage.Datum = Angebot.Datum;
```

- (ii) Finden Sie Nachnamen und Vornamen aller Kunden, für die kein Angebot existiert!

Lösungsvorschlag

```
SELECT k.Name, k.Vorname
FROM Kunde k
WHERE NOT EXISTS (SELECT * FROM Angebot a WHERE a.KID = k.KID)

oder:

SELECT k.Name, k.Vorname
FROM Kunde k
WHERE k.KID NOT IN (SELECT KID FROM Angebot);
```



- (iii) Geben Sie das Datum aller angebotenen Fahrten von München nach Stuttgart aus und sortieren Sie das Ergebnis aufsteigend!

```
SELECT Datum
FROM Angebot, Stadt
WHERE
 (SID = Start OR
 SID = Ziel)
AND
 (SName = 'München' OR SName = 'Stuttgart')
```

- (iv) Geben Sie für jeden Startort einer Anfrage den Namen der Stadt und die Anzahl der Anfragen aus.

```
SELECT SName, COUNT(*)
FROM Anfrage, Stadt
WHERE SID = Start
GROUP BY SID;
```

- (b) Wie sieht die Ergebnisrelation zu folgenden Anfragen auf den Beispieldaten aus?

```
SELECT *
FROM
Stadt
WHERE
NOT EXISTS (SELECT *
FROM Anfrage
WHERE Start = SID OR Ziel = SID) ;
```

Lösungsvorschlag

|                  |
|------------------|
| S1 Berlin Berlin |
|------------------|

```
SELECT KID, SUM (Plätze)
FROM Angebot
WHERE Plätze > 2
GROUP BY KID
HAVING SUM (Plätze) > 4;
```

## 46116 / 2014 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1

### Aufgabe 1: Allgemeine SWT, Vorgehensmodelle und Requirements Engineering

Kreuzen Sie für die folgenden Multiple-Choice-Fragen genau die richtigen Antworten deutlich an. Es kann mehr als eine Antwort richtig sein.

Jedes korrekt gesetzte oder korrekt nicht gesetzte Kreuz wird mit 1 Punkt gewertet. Jedes falsch gesetzte oder falsch nicht gesetzte Kreuz wird mit -1 Punkt gewertet. Eine Frage kann entwertet werden, dann wird sie nicht in der Korrektur berücksichtigt. Einzelne Antworten können nicht entwertet werden. Entwerten Sie eine Frage wie folgt

Die gesamte Aufgabe wird nicht mit weniger als 0 Punkten gewertet.

(a) Welche Aussage ist wahr?

- ☐ Je früher ein Fehler entdeckt wird, umso teurer ist seine Korrektur.
- ☐ Je später ein Fehler entdeckt wird, umso teurer ist seine Korrektur.
- ☐ Der Zeitpunkt der Entdeckung hat keinen Einfluss auf die Kosten.

Lösungsvorschlag

2 ist richtig: Je später der Fehler entdeckt wird, desto mehr wurde er schon in das Projekt „eingearbeitet“, daher dauert das Beseitigen des Fehlers länger und das kostet mehr Geld.

(b) Mit welcher Methodik können Funktionen spezifiziert werden?

- ☐ Als Funktionsvereinbarung in einer Programmiersprache
- ☐ Mit den Vor- und Nachbedingungen von Kontrakten
- ☐ Als Zustandsautomaten

Lösungsvorschlag

2 und 3 ist richtig: Die Spezifikation soll unabhängig von einer Programmiersprache sein.

(c) Welche Vorgehensmodelle sind für Projekte mit häufigen Änderungen geeignet?

- ☐ Extreme Programming (XP)
- ☐ Das V-Modell 97
- ☐ Scrum

Lösungsvorschlag

1 und 3 ist richtig. Das V-Modell ist ein starres Vorgehensmodell, bei dem alle Anforderungen zu Beginn vorhanden sein müssen.

(d) Welche der folgenden Aussagen ist korrekt?

- ☐ Mittels Prototyping versucht man die Anzahl an nötigen Unit-Tests zu reduzieren.
- ☐ Ein Ziel von Prototyping ist die Erhöhung der Qualität während der Anforderungsanalyse.
- ☐ Mit Prototyping versucht man sehr früh Feedback von Stakeholdern zu erhalten.

2 und 3 ist richtig: Prototypen müssen auch getestet werden. Es kann nicht an Tests gespart werden. Durch das häufige Feedback des Kunden / der Stakeholder können die Anforderungen immer genauer und klarer erfasst werden.

(e) Welche der folgenden Aussagen ist korrekt?

- ☐ Bei der Architektur sollten funktionale und nicht-funktionale Anforderungen beachtet werden.
- ☐ Bei der Architektur sollten nur funktionale Anforderungen beachtet werden.
- ☐ Bei der Architektur sollten nur nicht-funktionale Anforderungen beachtet werden.
- ☐ Bei der Architektur sollte auf die mögliche Änderungen von Komponenten geachtet werden.

1 und 4 ist richtig: Mögliche Änderungen werden durch klar definierte Schnittstellen und wenig Kopplung der Komponenten erleichtert. (Kopplung handelt von Abhängigkeiten zwischen Modulen. Kohäsion handelt von Abhängigkeiten zwischen Funktionen innerhalb eines Moduls.)

## 46116 / 2014 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2

Software wird oft in definierten Prozessen entwickelt. Diese nennt man Vorgehensmodelle.

Allgemein

(a) Was sind die Aufgaben eines Vorgehensmodells im Allgemeinen?

- liefert Erfahrungen und bewährte Methoden
- beschreibt die am Projekt beteiligten Rollen
- legt Aufgaben und Aktivitäten fest
- definiert einheitliche Begriffe
- gibt Techniken, Werkzeuge, Richtlinien / Standards an

(b) Was sind die wesentlichen Bestandteile eines Vorgehensmodells und in welcher Beziehung stehen diese zueinander?

Bestandteile: Anforderungsanalyse, Modellierung, Implementierung, Test, Auslieferung, Wartung

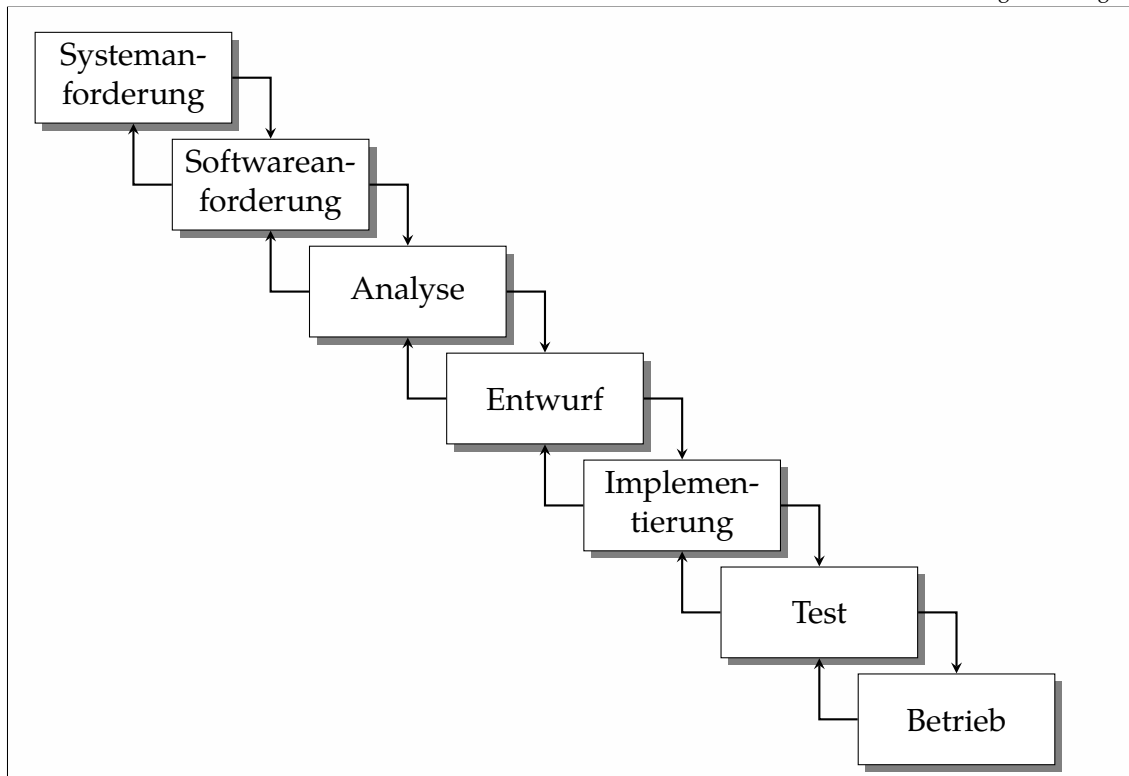
Die einzelnen Phasen bauen immer aufeinander aus. Je nach Vorgehensmodell können sich Phasen auch wiederholen (Prototyping, Scrum...).

Wasserfallmodell

Ein frühes Vorgehensmodell ist das von Dr. Winston Royce 1970 formalisierte Wasserfallmodell.

(a) Geben Sie eine schematische Darstellung des Wasserfallmodells an.

Lösungsvorschlag



(b) Nennen Sie zwei Probleme des Modells und erläutern Sie diese kurz.

Lösungsvorschlag

Fehler werden ggf. erst am Ende des Entwicklungsprozesses erkannt, da erst dort das Testen stattfindet. Dadurch kann die Behebung eines Fehlers sehr aufwändig und somit teuer werden.

Der Kunde / Endanwender wird erst nach der Implementierung wieder eingebunden. Das bedeutet, dass er nach der Stellung der Anforderungen keinen Einblick mehr in den Prozess hat und somit auch nicht gegensteuern kann, falls ihm etwas nicht gefällt oder er etwas nicht bedacht hat.

(c) In welchen Situationen lässt sich das Wasserfallmodell gut einsetzen?

Lösungsvorschlag

Das Wasserfallmodell ist geeignet, wenn es sich um ein von Anfang an klar definiertes Projekt ohne große Komplexität handelt, bei dem alle Anforderungen, Aufwand und Kosten schon zu Beginn des Projekts feststehen bzw.

abgeschätzt werden können.

Barry Boehm erweiterte das Wasserfallmodell 1979 zum so genannten V-Modell.

- (a) Geben Sie eine schematische Darstellung des V-Modells an.
- (b) Nennen Sie zwei Probleme des Modells und erläutern Sie diese kurz.

Lösungsvorschlag

- Die Nachteile des Wasserfallmodells bestehen weiterhin!
- Nicht für kleine Projekte geeignet, da aufwändige Tests vorgesehen sind, die im Kleinen detailliert meist nicht stattfinden (können).

- (c) Welchen Vorteil hat das V-Modell gegenüber dem Wasserfallmodell?

Lösungsvorschlag

Für jedes Dokument besteht ein entsprechender Test (Validierung / Verifikation). Dabei kann die Planung der Tests schon vor der eigentlichen Durchführung geschehen, so dass Aktivitäten im Projektteam parallelisiert werden können. So kann zum Beispiel der Tester die Testfälle für den Akzeptanztest (=Test des Systementwurfs) entwickeln, auch wenn noch keine Implementierung existiert.

In neuerer Zeit finden immer häufiger iterative und inkrementelle Vorgehensweisen Anwendung.

- (a) Erklären Sie den Begriff iterative Softwareentwicklung.

Lösungsvorschlag

Iterativ heißt, dass der Entwicklungsprozess mehrfach wiederholt wird: statt den „Wasserfall“ einmal zu durchlaufen, werden „kleine Wasserfälle“ hintereinander gesetzt.

- (b) Erklären Sie den Begriff inkrementelle Softwareentwicklung und grenzen Sie ihn von iterativer Softwareentwicklung ab.

Lösungsvorschlag

Bei der inkrementellen Entwicklung wird das System Schritt für Schritt fertig gestellt. D. h., dass ein Prototyp immer etwas mehr kann als der Prototyp davor. Dies wird durch die iterative Entwicklung unterstützt, da bei jeder Wiederholung des Entwicklungsprozesses ein neues Inkrement entsteht, d. h. ein neuer Prototyp, der mehr Funktionalitäten benutzt als der vorangegangene.

- (c) Nennen Sie jeweils zwei Vor- und Nachteile eines iterativen und inkrementellen Vorgehens im Vergleich zum Wasserfallmodell.

**Vorteile:**

- Risiken können früher erkannt werden.
- volatile Anforderungen können besser berücksichtigt werden.
- inkrementelle Auslieferung wird erleichtert.

**Nachteile:**

- komplexeres Projektmanagement
- schwerer messbar
- (Mehrarbeit)<sup>a</sup>

<sup>a</sup>Quelle: <https://www.pst.ifi.lmu.de/Lehre/WS0607/pm/vorlesung/PM-02-Prozess.pdf>

**46116 / 2014 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3**

Gegeben sei folgender Sachverhalt:

Für eine Verwaltungssoftware einer Behörde soll ein Bestellsystem entwickelt werden. Dabei sollen die Nutzer ihre Raummaße eingeben können. Anschließend können die Nutzer über ein Web-Interface das Büro gestalten und Möbel (wie zum Beispiel Wandschränke) und andere Einrichtungsgegenstände in einem virtuellen Büro platzieren. Aus dem Web-Interface kann die Einrichtung dann direkt bestellt werden. Dazu müssen die Nutzer ihre Büro-Nummer und den Namen und die Adresse der Behörde eingeben und die Bestellung bestätigen.

Weiterhin können Nutzer auch Büromaterialien über das Web-Interface bestellen. Dazu ist anstatt der Eingabe der Raummaße nur das Eingeben von Büro-Nummer und des Namens und der Adresse der Behörde erforderlich.

Zusätzlich zum Standard-Nutzer können sich auch Administratoren im System anmelden und Möbel zur Kollektion hinzufügen und aus der Kollektion entfernen. Die Möbel können eindeutig durch ihre Inventurnummer identifiziert werden.

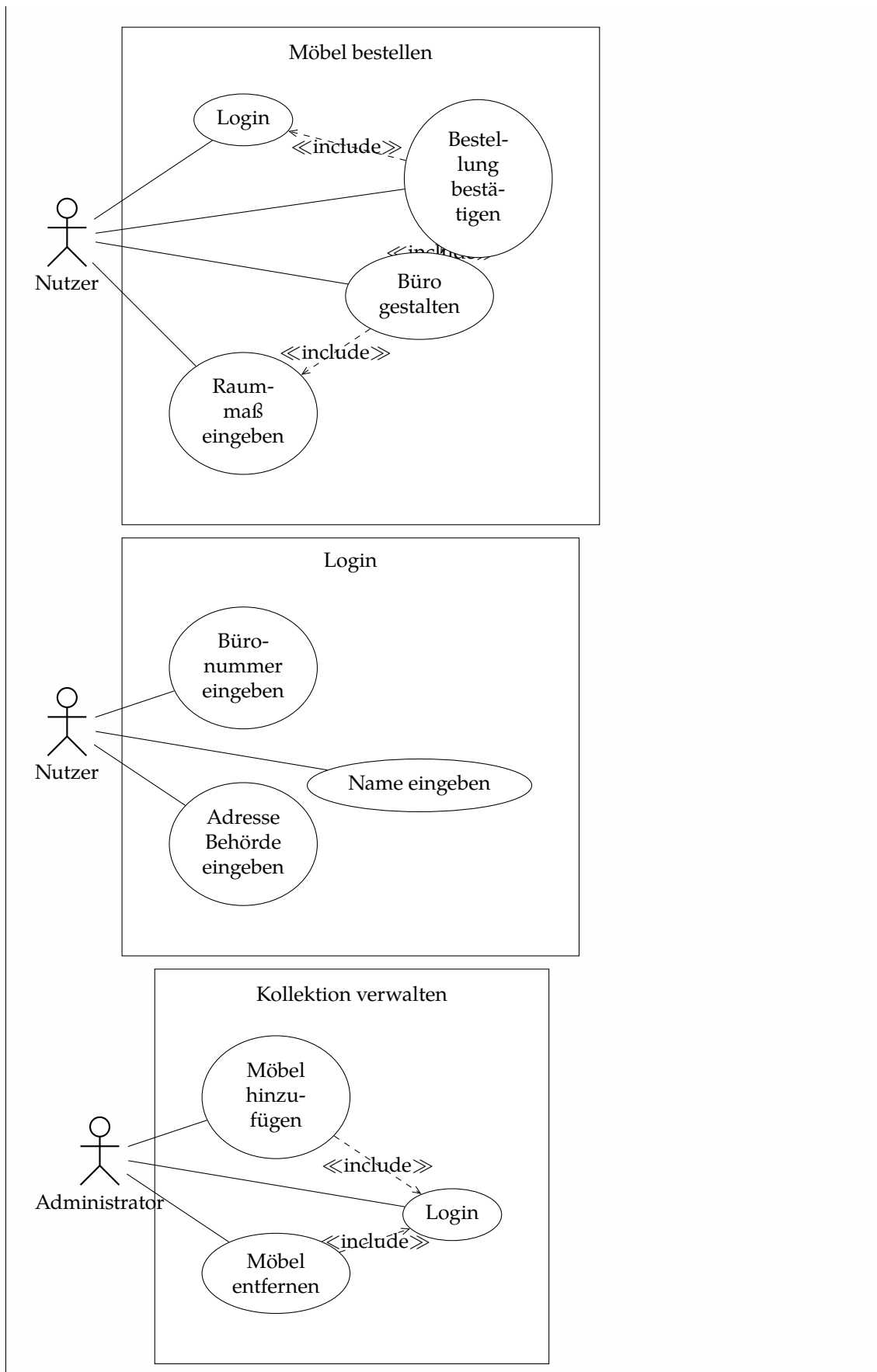
Um jegliche Veränderungen im System protokollieren zu können müssen Nutzer und Administratoren zur Bestätigung eingeloggt sein.

- (a) Erfassen Sie die drei Systemfunktionen *Möbel bestellen*, *Login* und *Kollektion verwalten* in einem UML-konformen Use Case Diagramm.

Login

Kollektion verwalten

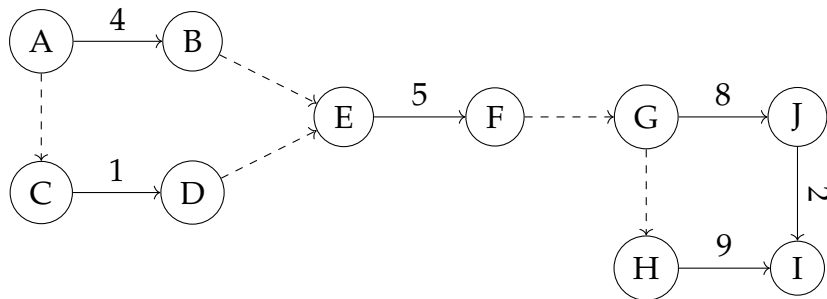
Lösungsvorschlag



- (b) Erstellen Sie ein UML-Klassendiagramm, welches die Beziehungen und sinnvolle Attribute der Klassen „Nutzer, Büro, Möbelstück und Wandschrank“ darstellt.
- (c) Instanzieren Sie das Klassendiagramm in einem Objektdiagramm mit den zwei Nutzern mit Namen Ernie und Bernd, einem Büro mit der Nummer CAPITOL2 und zwei Schränken mit den Inventurnummern S1.88 und S1.77.
- (d) Geben Sie für ein Büromöbelstück ein Zustandsdiagramm an. Überlegen Sie dazu, welche möglichen Zustände ein Möbelstück während des Bestellvorgangs haben kann und finden Sie geeignete Zustandsübergänge.

### 46116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Betrachten Sie folgendes CPM-Netzwerk:



- (a) Berechnen Sie die früheste Zeit für jedes Ereignis, wobei angenommen wird, dass das Projekt zum Zeitpunkt 0 startet.

Lösungsvorschlag

| $i$ | Nebenrechnung                                             | $FZ_i$ |
|-----|-----------------------------------------------------------|--------|
| A   |                                                           | 0      |
| B   |                                                           | 4      |
| C   |                                                           | 0      |
| D   |                                                           | 1      |
| E   | $\max(4, 1)$                                              | 4      |
| F   |                                                           | 9      |
| G   |                                                           | 9      |
| H   |                                                           | 9      |
| J   |                                                           | 17     |
| I   | $\max(9_{(\rightarrow H)} + 9, 17_{(\rightarrow J)} + 2)$ | 19     |

- (b) Setzen Sie anschließend beim letzten Ereignis die späteste Zeit gleich der frühesten Zeit und berechnen Sie die spätesten Zeiten.



| $i$ | Nebenrechnung | $SZ_i$ |
|-----|---------------|--------|
| A   | $\min(3,0)$   | 0      |
| B   |               | 4      |
| C   |               | 3      |
| D   |               | 4      |
| E   |               | 4      |
| F   |               | 9      |
| G   | $\min(10,9)$  | 9      |
| H   |               | 10     |
| J   |               | 17     |
| I   |               | 19     |

(c) Berechnen Sie nun für jedes Ereignis die Pufferzeiten.

| $i$    | A | B | C | D | E | F | G | H  | J  | I  |
|--------|---|---|---|---|---|---|---|----|----|----|
| $FZ_i$ | 0 | 4 | 0 | 1 | 4 | 9 | 9 | 9  | 17 | 19 |
| $SZ_i$ | 0 | 4 | 3 | 4 | 4 | 9 | 9 | 10 | 17 | 19 |
| GP     | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 1  | 0  | 0  |

(d) Bestimmen Sie den kritischen Pfad.

|                                                                                         |
|-----------------------------------------------------------------------------------------|
| $A \rightarrow B \rightarrow E \rightarrow F \rightarrow G \rightarrow J \rightarrow I$ |
|-----------------------------------------------------------------------------------------|

(e) Was ist ein Gantt-Diagramm? Worin unterscheidet es sich vom CPM-Netzwerk?

|  |
|--|
|  |
|--|

## 46116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 1

Gegeben sei folgendes relationales Schema, dessen Attribute nur atomare Attributwerte besitzen.

Computer: {IP, Name, Hersteller, Modell, Standort}

(a) Geben Sie für die folgenden Anfragen einen relationalen Ausdruck an:

(i) Geben Sie die IP-Adresse des Computers mit Namen „Chiemsee“ aus.

Lösungsvorschlag

SQL  
GROUP BY  
HAVING
$$\pi_{IP}(\sigma_{Name=Chiemsee}(Computer))$$

- (ii) Geben Sie 2er-Tupel von IP-Adressen der Computer am selben Standort aus.

Lösungsvorschlag

$$\pi_{c1.IP, c2.IP}(\sigma_{c1.Standort=c2.Standort}(\rho_{c1}(Computer) \times \rho_{c2}(Computer)))$$

- (b) Formulieren Sie die folgenden Anfragen in SQL:

- (i) Geben Sie die IP-Adressen der Rechner am Standort „Büro2“ aus.

Lösungsvorschlag

```
SELECT IP FROM Computer WHERE Standort = 'Büro2';
```

- (ii) Geben Sie alle Computer-Namen in aufsteigender Ordnung mit ihren IP-Adressen aus.

Lösungsvorschlag

```
SELECT Name, IP FROM Computer ORDER BY Name ASC;
```

- (iii) Geben Sie für jeden Hersteller die Anzahl der unterschiedlichen Modelle aus.

Lösungsvorschlag

```
SELECT COUNT(DISTINCT Modell), Hersteller
FROM Computer
GROUP BY Hersteller;
```

- (iv) Geben Sie für jeden Hersteller, welcher mindestens 2 unterschiedliche Modelle hat, die Anzahl der unterschiedlichen Modelle aus.

Lösungsvorschlag

```
SELECT Hersteller, COUNT(*) FROM Modelle GROUP BY Hersteller HAVING
↪ COUNT(*) > 1;
```

oder

Lösungsvorschlag

```
SELECT COUNT(DISTINCT Modell), Hersteller
FROM Computer
GROUP BY Hersteller
HAVING COUNT(DISTINCT Modell) >= 2;
```

```
-- AB 2 Einstieg Sql

-- Aufgabe 3: SQL-Anfragen auf einer Tabelle & Relationale Algebra

-- sudo mysql < Computer.sql
-- DROP DATABASE IF EXISTS Computer;
-- CREATE DATABASE Computer;
-- USE Computer;

CREATE TABLE Computer (
 IP VARCHAR(15) PRIMARY KEY NOT NULL,
 Name VARCHAR(30),
 Hersteller VARCHAR(30),
 Modell VARCHAR(30),
 Standort VARCHAR(30)
);

INSERT INTO Computer VALUES ('10.11.12.1', 'Chiemsee', 'HP', 'Spectre', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.2', 'Computer2', 'HP', 'Elite', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.3', 'Computer3', 'HP', 'Spectre', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.4', 'Computer4', 'HP', 'Elite', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.5', 'Computer5', 'HP', 'Spectre', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.6', 'Computer6', 'HP', 'Elite', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.7', 'Computer7', 'HP', 'Envy', 'Büro1');
INSERT INTO Computer VALUES ('10.11.12.8', 'Computer8', 'DELL', 'G3', 'Büro2');
INSERT INTO Computer VALUES ('10.11.12.9', 'Computer9', 'DELL', 'G7', 'Büro2');
INSERT INTO Computer VALUES ('10.11.12.10', 'Computer10', 'DELL', 'Latitude',
↪ 'Büro2');
INSERT INTO Computer VALUES ('10.11.12.11', 'Computer11', 'DELL', 'Alienware',
↪ 'Büro2');
INSERT INTO Computer VALUES ('10.11.12.12', 'Computer12', 'DELL', 'Inspiron',
↪ 'Büro2');
INSERT INTO Computer VALUES ('10.11.12.13', 'Computer13', 'DELL', 'XPS', 'Büro2');
INSERT INTO Computer VALUES ('10.11.12.14', 'Computer14', 'Apple', 'MacBook Air',
↪ 'Büro2');
INSERT INTO Computer VALUES ('10.11.12.15', 'Computer15', 'Apple', 'MacBook Air',
↪ 'Büro2');
INSERT INTO Computer VALUES ('10.11.12.16', 'Computer16', 'Apple', 'MacBook Air',
↪ 'Büro3');
INSERT INTO Computer VALUES ('10.11.12.17', 'Computer17', 'Apple', 'MacBook Air',
↪ 'Büro3');
INSERT INTO Computer VALUES ('10.11.12.18', 'Computer18', 'Apple', 'MacBook Air',
↪ 'Büro3');
INSERT INTO Computer VALUES ('10.11.12.19', 'Computer19', 'Apple', 'MacBook Air',
↪ 'Büro3');
INSERT INTO Computer VALUES ('10.11.12.20', 'Computer20', 'Apple', 'MacBook Air',
↪ 'Büro3');
INSERT INTO Computer VALUES ('10.11.12.21', 'Computer21', 'Apple', 'MacBook Air',
↪ 'Büro3');
INSERT INTO Computer VALUES ('10.11.12.22', 'Computer22', 'Apple', 'MacBook Air',
↪ 'Büro3');
INSERT INTO Computer VALUES ('10.11.12.23', 'Computer23', 'Apple', 'MacBook Air',
↪ 'Büro3');
```

**46116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 3**

Es sind folgende Informationen zu einer Datenbank für Konsulate gegeben:

- Jedes Konsulat hat einen Sitz in einer Stadt
- Zu einem **Konsulat** soll ein eindeutiger *Name* (KonsulatName) (z. B. Konsulat Bayern), die *Adresse* und der *Vor-* (KVorname) bzw. *Nachname* (KNachname) des Konsuls gespeichert werden.
- Für jede **Stadt** sollen der *Name* (StadtName), die *Anzahl der Einwohner* (EinwohnerAnzahl) sowie das Land in dem es liegt, festgehalten werden. Gehen Sie davon aus, dass eine Stadt nur in Zusammenhang mit dem zugehörigen Land identifizierbar ist.
- Für ein **Land** soll der Name in *Landessprache*, der *Name des Staatspräsidenten* (Staatspräsident) und eine eindeutige *ID* (LandesID) gespeichert werden.

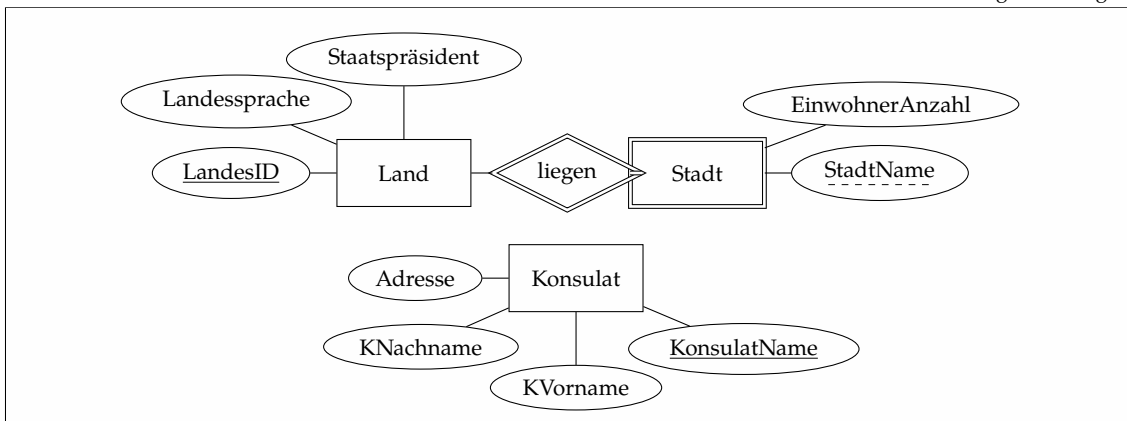
- ☐ E: Konsulat
- ☐ A: Name
- ☐ A: Adresse
- ☐ A: Vor-
- ☐ A: Nachname
- ☐ E: Stadt
- ☐ A: Name
- ☐ A: Anzahl der Einwohner
- ☐ R: liegt
- ☐ E: Land
- ☐ A: Landessprache
- ☐ A: Name des Staatspräsidenten
- ☐ A: ID

- (a) Entwerfen Sie für das obige Szenario ein ER-Diagramm in Chen-Notation. Bestimmen Sie hierzu:

- Die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- Die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
- Die Funktionalitäten der Relationship-Typen.

Hinweis: Achten Sie darauf, alle Totalitäten einzutragen.

Lösungsvorschlag



- (b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

Lösungsvorschlag

Konsulat(KonsulatName, KVorname, KNachname, Adresse, StadtName, LandesID)

Stadt(LandesID, StadtName, EinwohnerAnzahl)  
Land(LandesID, Landessprache, Staatspraesident)

Gantt-Diagramm

**46116 / 2015 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3**

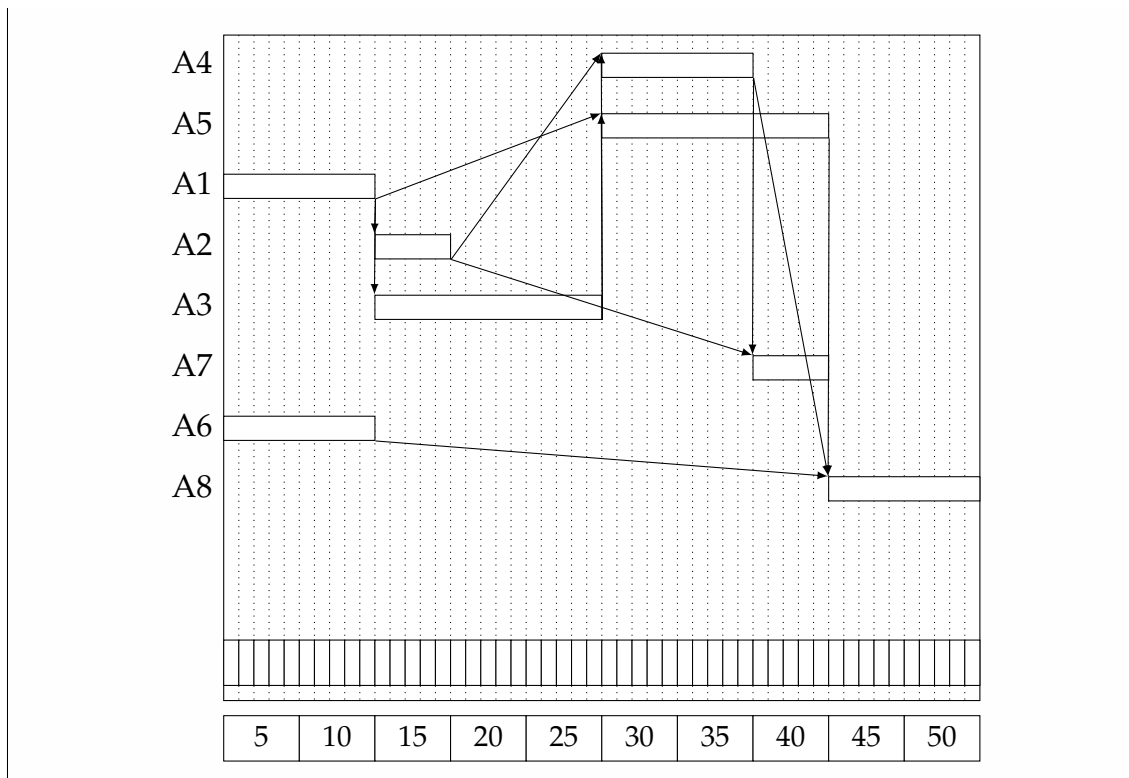
Betrachten Sie die folgende Tabelle zum Projektmanagement:

| Name | Dauer (Tage) | Abhängig von |
|------|--------------|--------------|
| A1   | 10           |              |
| A2   | 5            | A1           |
| A3   | 15           | A1           |
| A4   | 10           | A2, A3       |
| A5   | 15           | A1, A3       |
| A6   | 10           |              |
| A7   | 5            | A2, A4       |
| A8   | 10           | A4, A5, A6   |

Tabelle 1: Übersicht Arbeitspakete

- (a) Erstellen Sie ein Gantt-Diagramm, das die in der Tabelle angegebenen Abhängigkeiten berücksichtigt.

Lösungsvorschlag



- (b) Wie lange dauert das Projekt mindestens?
- (c) Geben Sie den oder die kritischen Pfad(e) an.
- (d) Konstruieren Sie ein PERT-Chart zum obigen Problem.

### 46116 / 2015 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 1

Erläutern Sie die folgenden Begriffe in knappen Worten:

- (a) Datenunabhängigkeit

Lösungsvorschlag

Änderungen an der physischen Speicher- oder der Zugriffsstruktur (beispielsweise durch das Anlegen einer Indexstruktur) haben keine Auswirkungen auf die logische Struktur der Datenbasis, also auf das Datenbankschema.

- (b) Superschlüssel

Lösungsvorschlag

Ein Superschlüssel ist ein Attribut oder Attributkombination, von der alle Attribute einer Relation funktional abhängen.

- (c) Referentielle Integrität

Unter Referentieller Integrität verstehen wir Bedingungen, die zur Sicherung der Datenintegrität bei Nutzung relationaler Datenbanken beitragen können. Demnach dürfen Datensätze über ihre Fremdschlüssel nur auf existierende Datensätze verweisen.

Danach besteht die Referentielle Integrität grundsätzlich aus zwei Teilen:

- (i) Ein neuer Datensatz mit einem Fremdschlüssel kann nur dann in einer Tabelle eingefügt werden, wenn in der referenzierten Tabelle ein Datensatz mit entsprechendem Wert im Primärschlüssel oder einem eindeutigen Alternativschlüssel existiert.
- (ii) Eine Datensatzlöschung oder Änderung des Schlüssels in einem Primär-Datensatz ist nur möglich, wenn zu diesem Datensatz keine abhängigen Datensätze in Beziehung stehen.

## 46116 / 2015 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 4

Das Datenbankschema der Versicherungsgesellschaft „Insurance Pro“ enthält sowohl Kunden- als auch Mitarbeiterdaten, sowie Informationen über Schadensfälle. Das Schema ist von folgender Gestalt, wobei FS für Fremdschlüssel und PS für Primärschlüssel steht:

Versicherungsnehmer (KNr, Name, Anschrift, Geburtsdatum)

Fahrzeug (Kennzeichen, Farbe, Fahrzeugtyp)

Mitarbeiter (MNr, Name, Kontaktdaten, Abteilung, Leiter)

Abteilung (ANr, Bezeichnung, Ort)

Versicherungsvertrag (VNr, Abschlussdatum, Art, Versicherungsnehmer, Fahrzeug, Mitarbeiter)

Schadensfall (SNr, Datum, Gesamtschaden, Beschreibung, Mitarbeiter)

Beteiligung (Schadensfall, Fahrzeug, Fahrzeugschaden)

- *Abteilung* ist FS und bezieht sich auf den PS von *Abteilung*.
- *Versicherungsnehmer* ist FS und bezieht sich auf den PS von *Versicherungsnehmer*.
- *Fahrzeug* ist FS und bezieht sich auf den PS von *Fahrzeug*.
- *Mitarbeiter* ist FS und bezieht sich auf den PS von *Mitarbeiter*.

- Attribut *Art* kann die Werte *HP* (Haftpflicht), *TK* (Teilkasko) und *VK* (Vollkasko) annehmen.
  - *Mitarbeiter* ist FS und bezieht sich auf den PS von *Mitarbeiter*.
  - *Schadensfall* ist FS und bezieht sich auf den PS von *Schadensfall*.
  - *Fahrzeug* ist FS und bezieht sich auf den PS von *Fahrzeug*.
- (a) Herr Meier schließt eine Teilkaskoversicherung bei „Insurance Pro“ am 11.11.2011 für seinen neuen Wagen, einen roten VW Golf VII mit amtlichem Kennzeichen BT-BT 2011, ab. Der Vertrag erhält die laufende Nummer 1631 und wird von Frau Schmied mit der Personalnummer 27 bearbeitet. Herr Meier erhält die Kundennummer 588, da er bisher noch kein Kunde war.

Geben Sie die SQL-Befehle an, um die neue Versicherung in die Datenbank von „Insurance Pro“ einzutragen. Werte, die hierbei nicht bekannt sind, sollen weggelassen werden, wobei angenommen werden darf, dass die entsprechenden Attribute nicht mit der Bedingung NOT NULL versehen sind.

```
INSERT INTO Fahrzeug
 (Kennzeichen, Farbe, Fahrzeugtyp) VALUES
 ('BT-BT 2011', 'rot', 'VW Golf VII');

INSERT INTO Versicherungsnehmer
 (KNr, Name) VALUES
 (588, 'Meier');

INSERT INTO Versicherungsvertrag
 (VNr, Abschlussdatum, Art, Versicherungsnehmer, Fahrzeug, Mitarbeiter)
 → VALUES
 (1631, 11.11.2011, 'TK', 588, 'BT-BT 2011', 27);
```

- (b) Beschreiben Sie umgangssprachlich, wonach mit folgendem Ausdruck gesucht wird:

$$\pi_{\text{Versicherungsnehmer}}(\sigma_{\text{Fahrzeugtyp}='Fiat500'}(FZ \bowtie_{\rho_{\text{Kennzeichen} \leftarrow \text{Fahrzeug}}} (VV)))$$

**Anmerkung:** Die Abkürzung FZ steht für die Tabelle *Fahrzeug* und VV für die Tabelle *Versicherungsvertrag*.

Es wird nach der Kundennummer aller Versicherungsnehmer gesucht, die einen Versicherungsvertrag für einen „Fiat 500“ abgeschlossen haben.

- (c) Die Angaben der Mitarbeiter (Nr., Name und Kontakt), deren Abteilung ihren Sitz in München oder Stuttgart hat, sollen explizit alphabetisch nach Namen sortiert ausgegeben werden. Geben Sie einen SQL-Befehl hierfür an.

```
SELECT m.MNr AS Nr, m.Name, m.Kontakt AS Kontakt
FROM Mitarbeiter m, Abteilung a
WHERE
 a.ANr = m.Abteilung AND
 a.Ort IN ('München', 'Stuttgart')
```



```
ORDER BY m.Name;
```

- (d) Es soll ausgegeben werden, wie oft jeder Versicherungsnehmer (KNr, Name, Anschrift) an einem Schadensfall beteiligt war. Hierbei sollen nur die Kunden, die an mindestens drei Schadensfällen beteiligt waren, in absteigender Reihenfolge aufgelistet werden.

```
SELECT vn.KNr, vn.Name, vn.Anschrift, COUNT(*) AS Schadensfaelle
FROM
 Versicherungsnehmer vn,
 Schadensfall s,
 Beteiligung b,
 Versicherungsvertrag vv
WHERE
 vn.KNr = vv.Versicherungsnehmer AND
 s.SNr = b.Schadensfall AND
 b.Fahrzeug = vv.Fahrzeug
GROUP BY vn.KNr, vn.Name, vn.Anschrift
HAVING COUNT(*) >= 3
ORDER BY Schadensfaelle DESC;
```

Es ist nicht genau angegeben, nach was sortiert werden soll. Ich sortiere nach Anzahl an Schadensfällen, weil dass meiner Meinung nach am meisten Sinn macht.

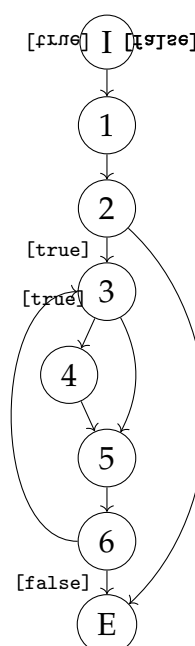
## 46116 / 2015 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2

Gegeben sei folgende Methode `isPalindrom` und ihr Kontrollflussgraph:

**Abkürzungen:** I = Import, E = Export

```
boolean isPalindrom(String s) {
 boolean yesItIs = true;
 if (s != null && s.length() > 1) {
 do {
 if (s.charAt(0) != s.charAt(s.length() - 1)) {
 yesItIs = false;
 }
 s = s.substring(1, s.length() - 1);
 } while (yesItIs && s.length() > 1);
 }
 return yesItIs;
}
```

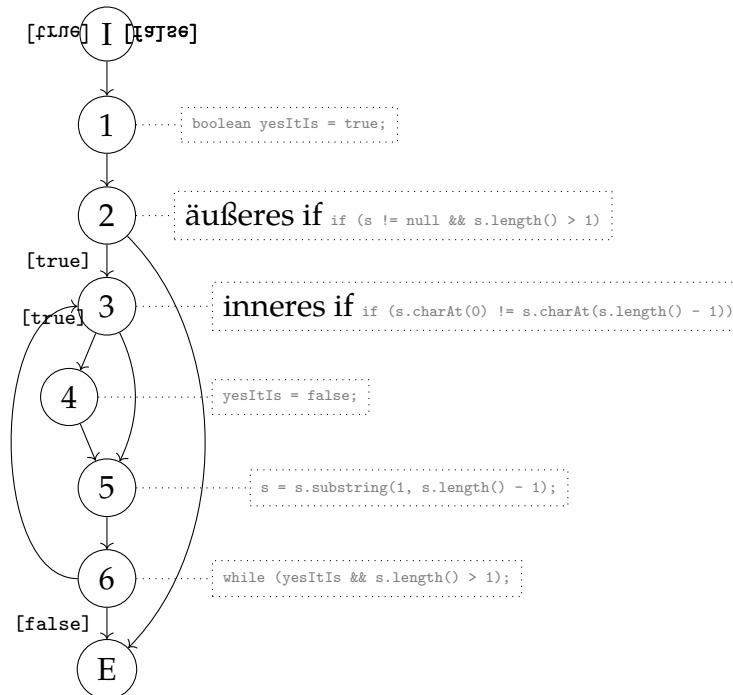
Code-Beispiel auf Github ansehen:  
[src/main/java/org/bschlangaul/aufgaben/sosy/pu\\_5/Aufgabe2.java](https://github.com/bschlangaul/aufgaben/sosy/pu_5/Aufgabe2.java)



- (a) Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, die zum Erzielen einer vollständigen ... mit **minimaler** Testfallanzahl und **möglichst kurzen** Pfaden genügen würden.

Lösungsvorschlag

Bemerkung: In der Aufgabenstellung steht „Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, [...]“. Das bedeutet, dass es hier erstmal egal ist, ob ein Pfad im Code möglich ist oder nicht!



- (i) Verzweigungsüberdeckung (Branch-Coverage,  $C_1$ )

Lösungsvorschlag

**Pfad 1 (p1)** ① - ① - ② - ⑤

(äußere **if**-Bedingung **false**)

**Pfad 2 (p2)** ① - ① - ② - ③ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - ⑤

(äußere **if**-Bedingung **true**, innere **if**-Bedingung **false**, Wiederholung, innere **if**-Bedingung **true**, keine Wiederholung)

- (ii) Schleife-Inneres-Überdeckung (Boundary-Interior-Coverage,  $C_{\infty,2}$ )

Lösungsvorschlag

**ohne Ausführung der Wiederholung (äußere Pfade):** p1 (siehe oben)

① - ① - ② - ⑤

**Boundary-Test:** (alle Pfade, die die Wiederholung betreten, aber nicht wiederholen; innerhalb des Schleifenrumpfes alle Pfade!)

**interior-Test:** (alle Pfade mit *einer* Wiederholung des Schleifenrumpfes; innerhalb des Schleifenrumpfes wieder alle Pfade!)

```

innere if-Bedingung true: ③ - ④ - ⑤ - ⑥
innere if-Bedingung false: ③ - ⑤ - ⑥
p5 ① - ① - ② - ③ - ④ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - E
 (innere if-Bedingung true, innere if-Bedingung true)
p2 (siehe oben) ① - ① - ② - ③ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - E
 (innere if-Bedingung false, innere if-Bedingung true)
p6 ① - ① - ② - ③ - ④ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - E
 (innere if-Bedingung true, innere if-Bedingung false)
p7 ① - ① - ② - ③ - ⑤ - ⑥ - ③ - ⑤ - ⑥ - E
 (innere if-Bedingung false, innere if-Bedingung false)

```

mit **minimaler** Testfallanzahl und **möglichst kurzen** Pfaden genügen würden.

- (b) Welche der vorangehend ermittelten Pfade für die  $C_{\infty,2}$ -Überdeckung sind mittels Testfällen tatsächlich überdeckbar („feasible“)? Falls der Pfad ausführbar ist, geben Sie den zugehörigen Testfall an - andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

Lösungsvorschlag

```

p1 s = "a";
p2 s = "abaa";
p3 s = "ab";
p4 s = "aa";
p5 nicht überdeckbar, da yesItIs = false, wenn innere if-Bedingung t
 rue) keine Wiederholung!
p6 nicht überdeckbar, da yesItIs = false, wenn innere if-Bedingung t
 rue) keine Wiederholung!
p7 s = "abba";

```

- (c) Bestimmen Sie anhand des Kontrollflussgraphen des obigen Code-Fragments die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

Lösungsvorschlag

```

M = b + p = 3 + 1 = 4
(b: Anzahl Binärverzweigungen, p: Anzahl Zusammenhangskomponenten)

Alternativ

M = e - n + 2p = 10 - 8 + 2 = 4
(e: Anzahl Kanten, n: Anzahl Knoten, p: Anzahl Zusammenhangskomponenten)

```

- (d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

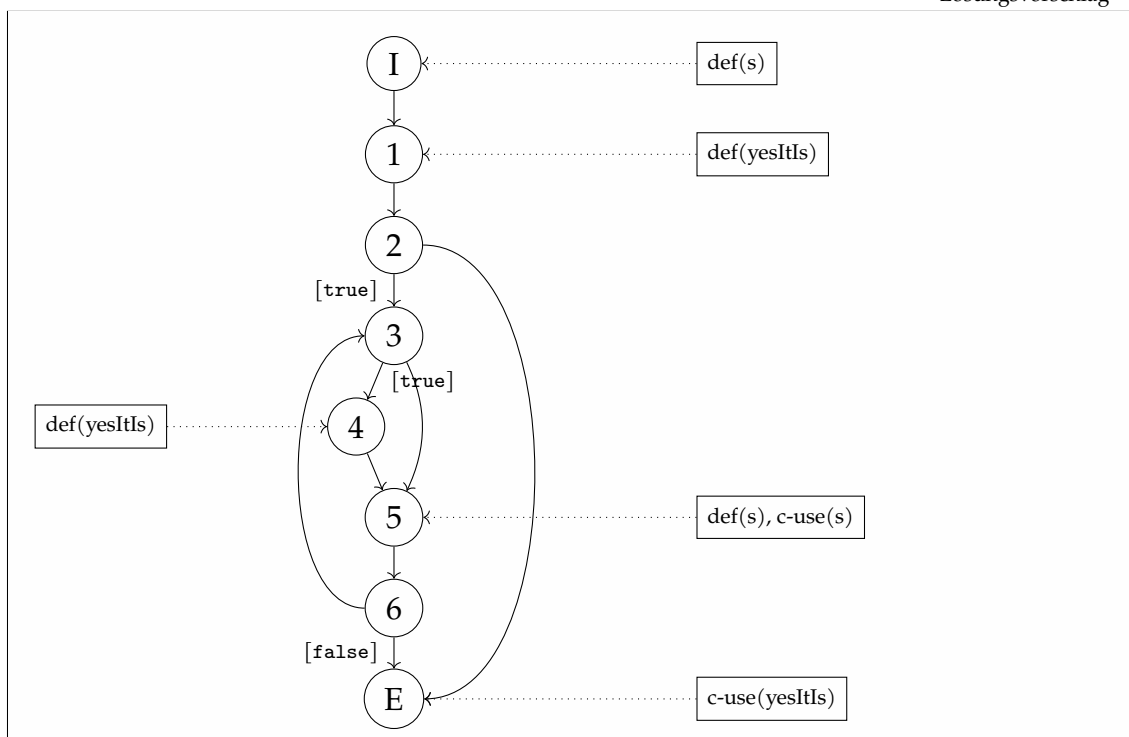
C2a Vollständige  
Pfadüberdeckung (Full  
Path Coverage)  
Datenfluss-annotierter  
Kontrollflussgraph  
Formale Verifikation  
wp-Kalkül

Lösungsvorschlag

Eine 100%-ige Pfadüberdeckung kann nicht erzielt werden, da es zum einen unüberdeckbare Pfade gibt (vgl. Teilaufgabe b). Zum anderen ist das Testen aller Testfälle nicht möglich, da die Anzahl an Zeichen des übergebenen Wortes nicht begrenzt ist und es somit eine unendliche Anzahl an Testfällen gibt.

- (e) Übernehmen Sie den vorgegebenen Kontrollflussgraphen und annotieren Sie ihn mit allen relevanten Datenflussereignissen. Geben Sie jeweils an, ob die Verwendungen berechnend (c-use) oder prädikativ (p-use) sind.

Lösungsvorschlag



### 46116 / 2015 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3

Sei  $wp(A, Q)$  die schwächste Vorbedingung (weakest precondition) eines Programmfragments  $A$  bei gegebener Nachbedingung  $Q$  so, dass  $A$  alle Eingaben, die  $wp(A, Q)$  erfüllen, auf gültige Ausgaben abbildet, die  $Q$  erfüllen.

Bestimmen Sie schrittweise und formal (mittels Floyd-Hoare-Kalkül) jeweils  $wp(A, Q)$  für folgende Code-Fragmente  $A$  und Nachbedingungen  $Q$  und vereinfachen Sie dabei den jeweils ermittelten Ausdruck so weit wie möglich.

Die Variablen  $x$ ,  $y$  und  $z$  in folgenden Pseudo-Codes seien ganzzahlig (vom Typ  $\text{int}$ ). Zur Vereinfachung nehmen Sie bitte im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können.

- (a) Sequenz:

```

x = -2 * (x + 2 * y);
y += 2 * x + y + z;
z -= x - y - z;

```

$$Q : \equiv x = y + z$$

Lösungsvorschlag

Code umformulieren:

```

x = -2 * (x + 2 * y);
y = y + 2 * x + y + z;
z = z - (x - y - z);

```

$$\text{wp}("x=-2*(x+2*y);y=2*y+2*x+z;z=z-(x-y-z);", x = y + z)$$

z einsetzen

$$\equiv \text{wp}("x=-2*(x+2*y);y=2*y+2*x+z;", x = y + (z - (x - y - z)))$$

Innere Klammer auflösen

$$\equiv \text{wp}("x=-2*(x+2*y);y=2*y+2*x+z;", x = y + (-x + y - 2z))$$

Klammer auflösen

$$\equiv \text{wp}("x=-2*(x+2*y);y=2*y+2*x+z;", x = -x + 2y + 2z)$$
 $-x$  auf beiden Seiten
$$\equiv \text{wp}("x=-2*(x+2*y);y=2*y+2*x+z;", 0 = -2x + 2y + 2z)$$
 $\div 2$  auf beiden Seiten
$$\equiv \text{wp}("x=-2*(x+2*y);y=2*y+2*x+z;", 0 = -x + y + z)$$

y einsetzen

$$\equiv \text{wp}("x=-2*(x+2*y);", 0 = -x + (2y + 2x + z) + z)$$

Term vereinfachen

$$\equiv \text{wp}("x=-2*(x+2*y);", 0 = x + 2y + 2z)$$

x einsetzen

$$\equiv \text{wp}("", 0 = (-2(x + 2y)) + 2y + 2z)$$

wp weglassen

$$\equiv 0 = (-2(x + 2y)) + 2y + 2z$$

ausmultiplizieren

$$\equiv 0 = (-2x - 4y) + 2y + 2z$$

Klammer auflösen, vereinfachen

$$\equiv 0 = -2x - 2y + 2z$$

$\div 2$  auf beiden Seiten

$$\equiv 0 = -x - y + z$$

$x$  nach links holen mit  $+x$  auf beiden Seiten

$$\equiv x = -y + z$$

$y$  ganz nach links schreiben

$$\equiv x = z - y$$

$$x = -2 \cdot (x + 2 \cdot y)$$

(b) Verzweigung:

```
if (x < y) {
 x = y + z;
} else if (y > 0) {
 z = y - 1;
} else {
 x -= y -= z;
}
```

$Q \equiv x > z$

Lösungsvorschlag

**1. Fall:**  $x < y$

**2. Fall:**  $x \geq y \wedge y > 0$

**3. Fall:**  $x \geq y \wedge y \leq 0$

Code umformulieren:

```
if (x < y) {
 x = y + z;
} else if (x >= y && y > 0) {
 z = y - 1;
} else {
 y = y - z;
 x = x - y;
}
```

$\text{wp}(\text{"if}(x < y)\{x=y+z;\}\text{else if}(x \geq y \ \&\& \ y > 0)\{z=y-1;\}\text{else}\{y=y-z; x=x-y;\}", x > z)$

$\equiv$

(In mehrere kleinere wp-Kalküle aufsplitten)

$$\begin{aligned}
& \left( (x < y) \wedge \text{wp}("x=y+z;", x > z) \right) \vee \\
& \left( (x \geq y \wedge y > 0) \wedge \text{wp}("z=y-1;", x > z) \right) \vee \\
& \left( (x \geq y \wedge y \leq 0) \wedge \text{wp}("y=y-z; x=x-y;", x > z) \right)
\end{aligned}$$

≡

(Code einsetzen)

$$\begin{aligned}
& \left( (x < y) \wedge \text{wp}("", y + z > z) \right) \vee \\
& \left( (x \geq y \wedge y > 0) \wedge \text{wp}("", x > y - 1) \right) \vee \\
& \left( (x \geq y \wedge y \leq 0) \wedge \text{wp}("y=y-z;", x - y > z) \right)
\end{aligned}$$

≡

(wp-Kalkül-Schreibweise weg lassen, Code weiter einsetzen)

$$\begin{aligned}
& \left( (x < y) \wedge y + z > z \right) \vee \\
& \left( (x \geq y \wedge y > 0) \wedge x > y - 1 \right) \vee \\
& \left( (x \geq y \wedge y \leq 0) \wedge \text{wp}("", x - (y - z) > z) \right)
\end{aligned}$$

≡

(Terme vereinfachen, wp-Kalkül-Schreibweise weg lassen)

$$\begin{aligned}
& \left( x < y \wedge y > 0 \right) \vee \\
& \left( x \geq y^a \wedge y > 0 \right) \vee \\
& \left( (x \geq y \wedge y \leq 0) \wedge x - (y - z) > z \right)
\end{aligned}$$

≡

(letzten Term vereinfachen)

$$\begin{aligned}
 & (x < y \wedge y > 0) \vee \\
 & (x \geq y \wedge y > 0) \vee \\
 & ((x \geq y \wedge y \leq 0) \wedge x - y > 0) \\
 \equiv & \hspace{15em} (\text{ein } \wedge \text{ eliminieren}) \\
 & (x < y \wedge y > 0) \vee \\
 & (x \geq y \wedge y > 0) \vee \\
 & (y \leq 0 \wedge x > y)
 \end{aligned}$$


---

$^a x > y - 1 \wedge x \geq y$  ergibt  $x \geq y$   
 $^a x > y - 1 \wedge x \geq y$  ergibt  $x \geq y$

(c) Mehrfachauswahl:

```

switch (z) {
 case "x":
 y = "x";
 case "y":
 y = --z;
 break;
 default:
 y = 0x39 + "?";
}

```

$Q \equiv 'x' = y$

Hinweis zu den ASCII-Codes

- $'x' = 120_{(10)}$
- $'y' = 121_{(10)}$
- $0x39 = 57_{(10)}$
- $'?' = 63_{(10)}$

Lösungsvorschlag

Mehrfachauswahl in Bedingte Anweisungen umschreiben. Dabei beachten, dass bei fehlendem **break** die Anweisungen im folgenden Fall bzw. ggf. in den folgenden Fällen ausgeführt werden:



```

if (z == "x") {
 y = "x";
 y = z - 1;
} else if (z == "y") {
 y = z - 1;
} else {
 y = 0x39 + "?";
}

```

Da kein `break` im Fall `z == "x"`. `--z` bedeutet, dass die Variable erst um eins verringert und dann zugewiesen wird.

```

if (z == 120) {
 y = 120;
 y = 120 - 1;
} else if (z == 121) {
 y = 121 - 1;
} else {
 y = 57 + 63;
}

```

Vereinfachung / Zusammenfassung:

```

if (z == 120) {
 y = 120;
 y = 119;
} else if (z == 121) {
 y = 120;
} else {
 y = 120;
}

```

$\text{wp}(\text{"if}(z==120)\{y=120;y=119;\}\text{else if}(z==121)\{y=120;\}\text{else}\{y=120;\}", 120 = y)$

$\equiv$

(In mehrere kleinere wp-Kalküle aufsplitten)

$$\begin{aligned}
 & \left( (z = 120) \wedge \text{wp}("y=120;y=119;", 120 = y) \right) \vee \\
 & \left( ((z \neq 120) \wedge (z = 121)) \wedge \text{wp}("y=120;", 120 = y) \right) \vee \\
 & \left( ((z \neq 120) \wedge (z \neq 121)) \wedge \text{wp}("y=120;", 120 = y) \right)
 \end{aligned}$$

$\equiv$

(Code einsetzen)

$$\begin{aligned}
& \left( (z = 120) \wedge \text{wp}("y=120;", 120 = 119) \right) \vee \\
& \left( ((z \neq 120) \wedge (z = 121)) \wedge \text{wp}("", 120 = 120) \right) \vee \\
& \left( ((z \neq 120) \wedge (z \neq 121)) \wedge \text{wp}("", 120 = 120) \right) \\
\equiv & \hspace{15em} (\text{vereinfachen}) \\
& \text{false} \vee \\
& \left( (z = 121) \wedge \text{true} \right) \vee \\
& \left( ((z \neq 120) \wedge (z \neq 121)) \wedge \text{true} \right) \\
\equiv & \text{false} \vee (z = 121) \vee ((z \neq 120) \wedge (z \neq 121)) \\
\equiv & (z = 121) \vee (z \neq 121) \\
\equiv & z \neq 121 \\
& \text{Alle Zahlen außer 120 sind möglich bzw. alle Zeichen außer 'x'}.
\end{aligned}$$

## 46116 / 2016 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 5

- (a) Nennen Sie die vier wesentlichen Eigenschaften einer Transaktion und erläutern Sie jede Eigenschaft kurz (ein Satz pro Eigenschaft).

Lösungsvorschlag

**Atomicity** Eine Transaktion ist atomar, d.h. von den vorgesehenen Änderungsoperationen auf die Datenbank haben entweder alle oder keine eine Wirkung auf die Datenbank.

**Consistency** Eine Transaktion überführt einen korrekten (konsistenten) Datenbankzustand wieder in einen korrekten (konsistenten) Datenbankzustand.

**Isolation** Eine Transaktion bemerkt das Vorhandensein anderer (parallel ablaufender) Transaktionen nicht und beeinflusst auch andere Transaktionen nicht.

**Durability** Die durch eine erfolgreiche Transaktion vorgenommenen Änderungen sind dauerhaft (persistent).

- (b) Gegeben ist die folgende Historie (Schedule) dreier Transaktionen:

$r_1(B) \rightarrow w_1(C) \rightarrow r_3(C) \rightarrow r_1(A) \rightarrow c_1 \rightarrow r_2(C) \rightarrow r_3(C) \rightarrow r_2(C) \rightarrow w_2(B) \rightarrow c_2 \rightarrow c_3$

Zeichnen Sie den Serialisierbarkeitsgraphen zu dieser Historie und begründen Sie, warum die Historie serialisierbar ist oder nicht.

### Exkurs: Historie

In Transaktionssystemen existiert ein Ausführungsplan für die parallele Ausführung mehrerer Transaktionen. Der Plan wird auch Historie genannt und gibt an, in welcher Reihenfolge die einzelnen Operationen der Transaktion ausgeführt werden. Als serialisierbar bezeichnet man eine Historie, wenn sie zum selben Ergebnis führt wie eine nacheinander (seriell) ausgeführte Historie über dieselben Transaktionen.

Lösungsvorschlag

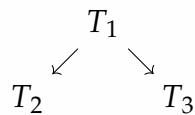
Der Algorithmus geht schrittweise durch den Ablaufplan unten und hebt die Abhängigkeiten der aktiven Transaktion zu allen anderen Transaktionen hervor. Hierfür werden in allen nachfolgenden Schritten solche Operationen gesucht, die einen Konflikt mit der aktuellen Operation hervorrufen. Konflikt-Operationen sind: read after write, write after read und write after write auf denselben Datenobjekt.

| $T_1$    | $T_2$    | $T_3$    |
|----------|----------|----------|
| $r_1(B)$ |          |          |
| $w_1(C)$ |          |          |
|          |          | $r_3(C)$ |
| $r_1(A)$ |          |          |
| $c_1$    |          |          |
|          | $r_2(C)$ |          |
|          |          | $r_3(C)$ |
|          | $r_2(C)$ |          |
|          | $w_2(B)$ |          |
|          | $c_2$    |          |
|          |          | $c_3$    |

### Konfliktoperation

- $r_1(B) < w_2(B)$ : Kante von  $T_1$  nach  $T_2$
- $w_1(C) < r_3(C)$ : Kante von  $T_1$  nach  $T_3$
- $w_1(C) < r_2(C)$ : Kante von  $T_1$  nach  $T_2$

### Serialisierbarkeitsgraph



Zwei-Phasen-Sperrprotokoll  
Petri-Netz  
Erreichbarkeitsgraph

Es gibt keinen Zyklus im Graph. Er ist deshalb serialisierbar. Wenn ein Zyklus auftreten würde, dann wäre er nicht serialisierbar.

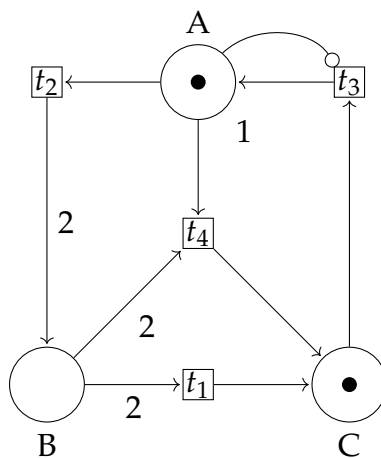
- (c) Geben Sie an, wodurch die erste und die zweite Phase des Zwei-Phasen-Sperrprotokolls jeweils charakterisiert sind (ein Satz pro Phase).

Lösungsvorschlag

Die zwei Phasen des Protokolls bestehen aus einer *Sperrphase*, in der alle benötigten Objekte für die Transaktion gesperrt werden. In der zweiten Phase werden die *Sperren wieder freigegeben*, sodass die Objekte von anderen Transaktionen genutzt werden können.

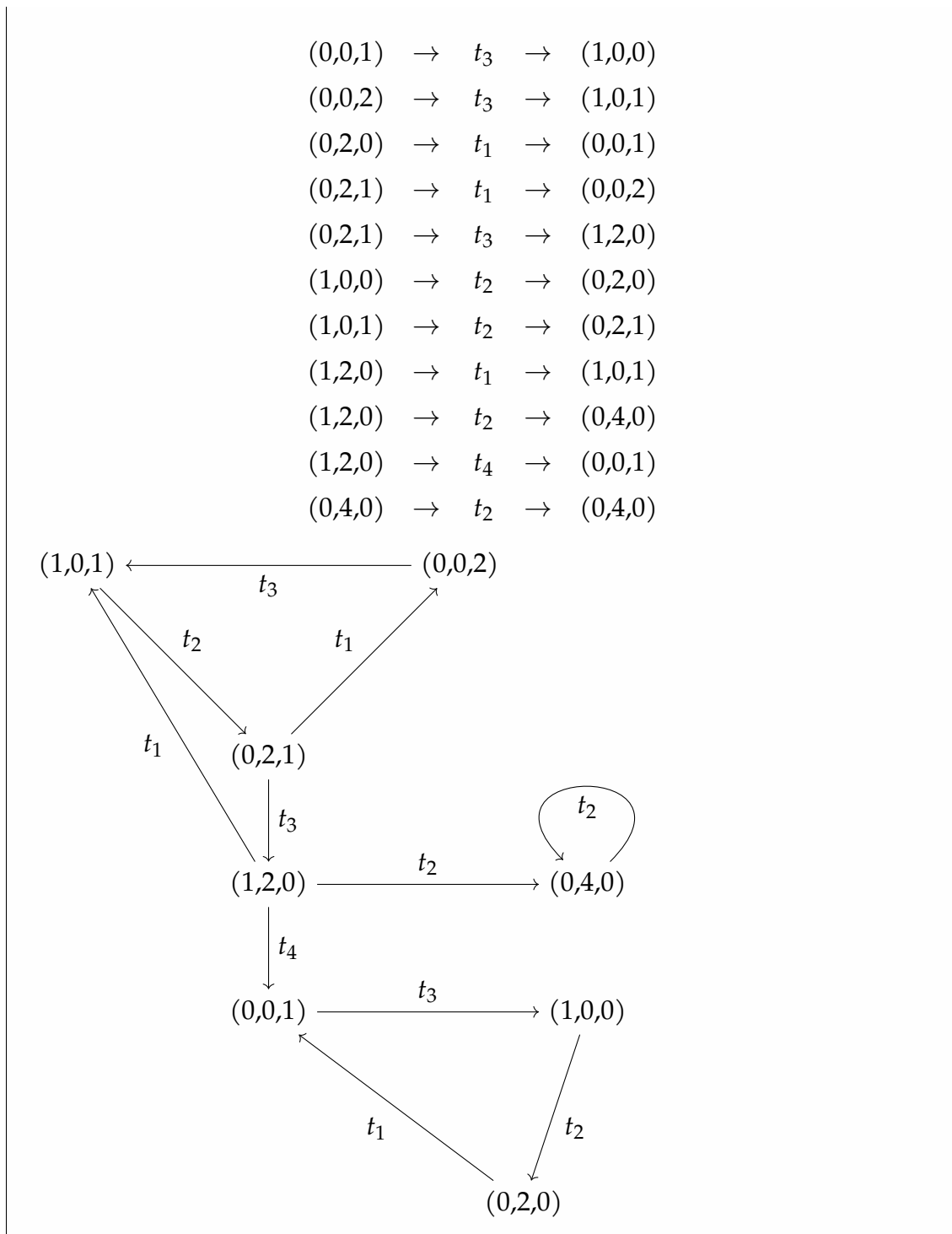
## 46116 / 2016 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2

Gegeben sei das folgende Petri-Netz:



- (a) Erstellen Sie den zum Petri-Netz gehörenden Erreichbarkeitsgraphen. Die Belegungen sind jeweils in der Form  $[A, B, C]$  anzugeben. Beschriften Sie auch jede Kante mit der zugehörigen Transition. Beachten Sie die auf 1 beschränkte Kapazität von Stelle A oder alternativ die Inhibitor-Kante von A zu  $t_3$  (beides ist hier semantisch äquivalent).

Lösungsvorschlag

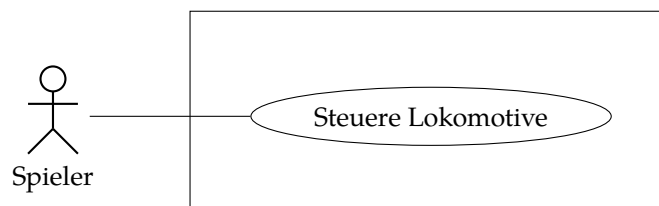


- (b) Wie kann man mit Hilfe des Erreichbarkeitsgraphen feststellen, ob ein Petri-Netz lebendig ist?
- (c) Aufgrund von Transition  $t_4$  ist das gegebene Petri-Netz nicht stark lebendig. Wie müssten die Pfeilgewichte der Transition  $t_4$  verändert werden, damit das Petri-Netz mit der gegebenen Startmarkierung beschränkt bleibt und lebendig wird?

$t_4$  nach C mit Gewicht 2 versehen

## 46116 / 2016 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2

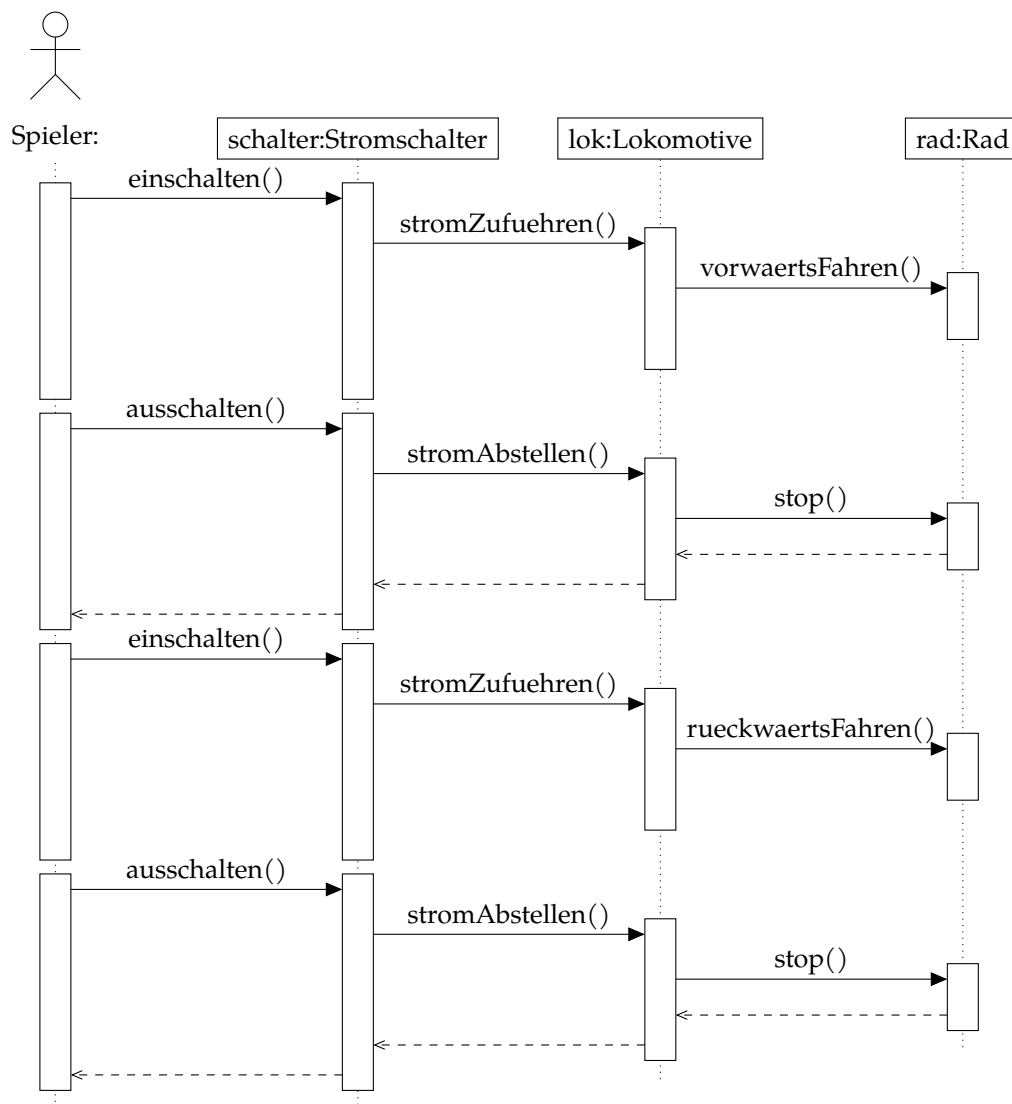
Die Fahrtrichtung der ersten elektrischen Spielzeugeisenbahnen wurde häufig durch Stromunterbrechung gesteuert. Dazu betrachten wir das Anwendungsfall-Diagramm.



An dem Anwendungsfall „*Steuere Lokomotive*“ sind ein Spieler, als Akteur außerhalb des Systems, und jeweils ein Objekt der Klassen *Stromschalter*, *Lokomotive*, *Scheinwerfer* und *Rad*, als Objekte innerhalb des Systems, beteiligt. Zur Vereinfachung wird nur ein Objekt der Klasse *Rad* stellvertretend für alle vier Räder modelliert. Der Anwendungsfall zum Steuern einer Lokomotive wird durch folgendes Szenario beschrieben.

- Der Spieler schaltet den Stromschalter ein, woraufhin der Schalter der Lokomotive *Strom zuführt*.
- Die Lokomotive schickt nun den Rädern ein Signal um *vorwärts* zu fahren.
- Dann schaltet der Spieler den Stromschalter aus, woraufhin der Schalter die Stromzufuhr bei der Lokomotive *abstellt*.
- Daraufhin schickt die Lokomotive das Signal *stop* an die Räder.
- Der Spieler schaltet jetzt den Stromschalter wieder ein, woraufhin der Schalter der Lokomotive *Strom zuführt*.
- Die Lokomotive schickt den Rädern ein Signal um *rückwärts* zu fahren.
- Nun schaltet der Spieler den Stromschalter wieder aus, woraufhin der Schalter die Stromzufuhr bei die Lokomotive *abstellt*.
- Daraufhin schickt die Lokomotive wieder das Signal *stop* an die Räder.

Geben Sie ein Sequenzdiagramm an, das die oben beschriebenen Interaktionen zwischen Spieler, Stromschalter, Lokomotive und Rädern beschreibt.



## 46116 / 2016 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 4

Gegeben sei folgende rekursive Methodendeklaration in der Sprache Java. Es wird als Vorbedingung vorausgesetzt, dass die Methode `cn` nur für Werte  $n \geq 0$  aufgerufen wird.

```
int cn(int n) {
 if (n == 0)
 return 1;
 else
 return (4 * (n - 1) + 2) * cn(n - 1) / (n + 1);
}
```

Sie können im Folgenden vereinfachend annehmen, dass es keinen Überlauf in der Berechnung gibt, dass der Datentyp `int` für die Berechnung des Ergebnisses stets ausreicht.

- (a) Beweisen Sie mittels vollständiger Induktion, dass der Methodenaufruf `cn(n)` für jedes  $n \geq 0$  die  $n$ -te Catalan-Zahl  $C_n$  berechnet, wobei

$$C_n = \frac{(2n)!}{(n+1)! \cdot n!}$$

### Exkurs: Fakultät

Für alle natürlichen Zahlen  $n$  ist

$$n! = 1 \cdot 2 \cdot 3 \cdots n = \prod_{k=1}^n k$$

als das Produkt der natürlichen Zahlen von 1 bis  $n$  definiert. Da das leere Produkt stets 1 ist, gilt

$$0! = 1$$

Die Fakultät lässt sich auch rekursiv definieren:

$$n! = \begin{cases} 1, & n = 0 \\ n \cdot (n-1)!, & n > 0 \end{cases}$$

Fakultäten für negative oder nicht ganze Zahlen sind nicht definiert. Es gibt aber eine Erweiterung der Fakultät auf solche Argumente<sup>a</sup>

<sup>a</sup>[https://de.wikipedia.org/wiki/Fakultät\\_\(Mathematik\)](https://de.wikipedia.org/wiki/Fakultät_(Mathematik))

### Exkurs: Catalan-Zahl

Die Catalan-Zahlen bilden eine Folge natürlicher Zahlen, die in vielen Problemen der Kombinatorik auftritt. Sie sind nach dem belgischen Mathematiker Eugène Charles Catalan benannt.

Die Folge der Catalan-Zahlen  $C_0, C_1, C_2, C_3, \dots$  beginnt mit  $1, 1, 2, 5, 14, 42, 132, \dots$ <sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/Catalan-Zahl>

Beim Induktionsschritt können Sie die beiden folgenden Gleichungen verwenden:

(i)  $(2(n+1))! = (4n+2) \cdot (n+1) \cdot (2n)!$

(ii)  $(a+2)! \cdot (n+1)! = (n+2) \cdot (n+1) \cdot (n+1)! \cdot n!$

Lösungsvorschlag

### Induktionsanfang

— Beweise, dass  $A(1)$  eine wahre Aussage ist. \_\_\_\_\_



$$\begin{aligned}C_0 &= \frac{(2 \cdot 0)!}{(0+1)! \cdot 0!} \\&= \frac{0!}{1! \cdot 0!} \\&= \frac{1}{1 \cdot 1} \\&= \frac{1}{1} \\&= 1\end{aligned}$$

**Induktionsvoraussetzung**

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . \_\_\_\_\_

$$C_n = \frac{(2n)!}{(n+1)! \cdot n!}$$

**Induktionsschritt**

— Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss.

**Vom Code ausgehend**

$$\begin{aligned}
 C_{n+1} &= \frac{(4 \cdot (n+1-1) + 2) \cdot \text{cn}(n+1-1)}{n+1+1} && \text{Java nach Mathe} \\
 &= \frac{(4n+2) \cdot \text{cn}(n)}{n+2} && \text{addiert, subtrahiert} \\
 &= \frac{(4n+2) \cdot (2n)!}{(n+2) \cdot (n+1)! \cdot n!} && \text{für cn(n) Formel eingesetzt} \\
 &= \frac{(4n+2) \cdot (2n)! \cdot (n+1)}{(n+2) \cdot (n+1)! \cdot n! \cdot (n+1)} && (n+1) \text{ multipliziert} \\
 &= \frac{(4n+2) \cdot (n+1) \cdot (2n)!}{(n+2) \cdot (n+1)! \cdot (n+1) \cdot n!} && \text{umsortiert} \\
 &= \frac{(2(n+1))!}{(n+2)! \cdot (n+1)!} && \text{Hilfsgleichungen verwendet} \\
 &= \frac{(2(n+1))!}{((n+1)+1)! \cdot (n+1)!} && (n+1) \text{ verdeutlicht}
 \end{aligned}$$

**Mathematische Herangehensweise**

$$\begin{aligned}
 C_{n+1} &= \frac{(2(n+1))!}{((n+1)+1)! \cdot (n+1)!} && n+1 \text{ in } C_n \text{ eingesetzt} \\
 &= \frac{(2(n+1))!}{(n+2)! \cdot (n+1)!} && \text{addiert} \\
 &= \frac{(4n+2) \cdot (n+1) \cdot (2n)!}{(n+2) \cdot (n+1) \cdot (n+1)! \cdot n!} && \text{Hilfsgleichungen verwendet} \\
 &= \frac{(4n+2) \cdot (2n)!}{(n+2) \cdot (n+1)! \cdot n!} && (n+1) \text{ gekürzt} \\
 &= \frac{4n+2}{n+2} \cdot C_n && \text{Catalan-Formel ersetzt} \\
 &= \frac{4((n+1)-1)+2}{(n+1)+1} \cdot C_{(n+1)-1} && (n+1) \text{ verdeutlicht}
 \end{aligned}$$

- (b) Geben Sie eine geeignete Terminierungsfunktion an und begründen Sie, warum der Methodenaufruf  $cn(n)$  für jedes  $n \geq 0$  terminiert.

Lösungsvorschlag

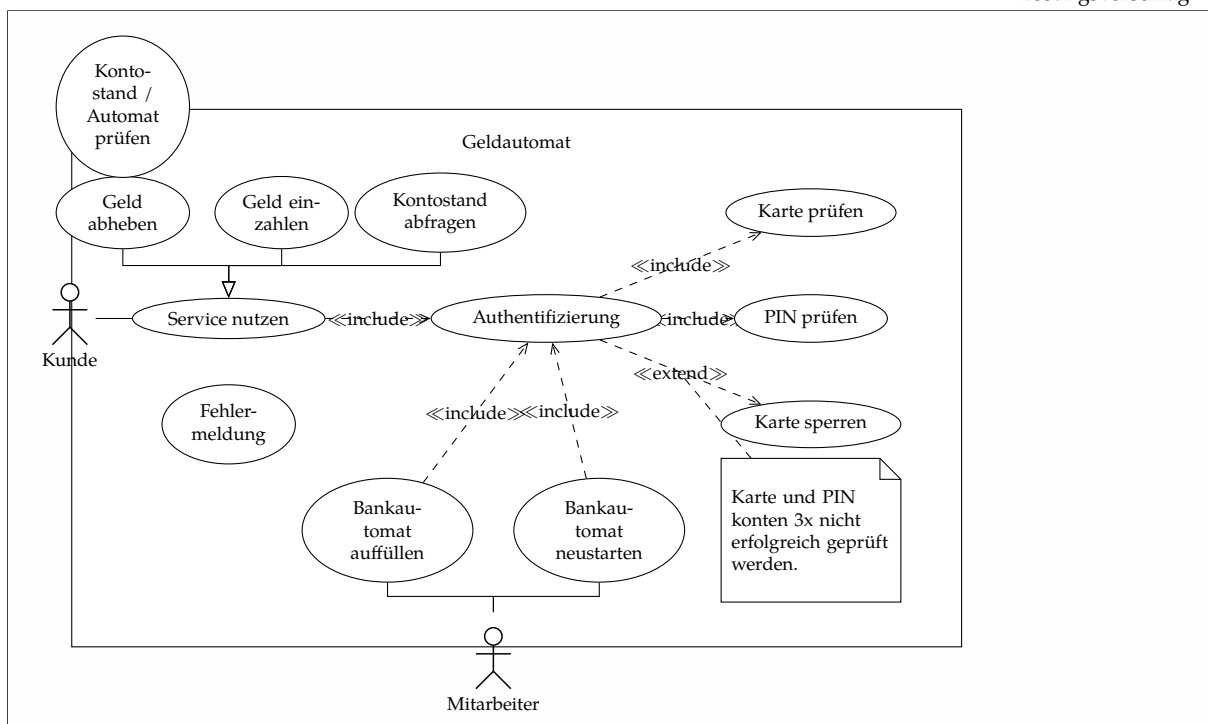
$T(n) = n$ . Diese Funktion verringert sich bei jedem Rekursionsschritt um eins. Sie ist monoton fallend und für  $T(0) = 0$  definiert. Damit ist sie eine Terminierungsfunktion für  $cn(n)$ .

## 46116 / 2017 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Im Folgenden sehen Sie ein fehlerhaftes Use-Case-Diagramm für das System „Geldautomat“. Geben Sie ein korrektes Use-Case-Diagramm (Beziehungen und Beschriftungen) entsprechend der folgenden Beschreibung an. Sollte es mehrere Möglichkeiten geben, begründen Sie Ihre Entscheidung kurz.

Kunden können am Geldautomat verschiedene Services nutzen, es kann Geld abgehoben und eingezahlt sowie der Kontostand abgefragt werden. Für jeden dieser Services ist eine Authentifizierung notwendig. Diese besteht aus der Prüfung der Bankkarte und des eingegebenen PINs. Manche Bankautomaten können Karten bei zu vielen Fehlversuchen (3) bei der Anmeldung sperren, andere geben Fehlermeldungen aus, falls die Karte gesperrt wurde oder nicht mehr genügend Geld auf dem Konto oder im Bankautomaten vorhanden ist. Mitarbeiter bzw. Mitarbeiterinnen der Bank können sich ebenfalls über PIN und Karte authentifizieren, um dann den Bankautomaten neu zu starten oder mit Geld aufzufüllen. Bevor Geld abgehoben werden kann, ist sicherzustellen, dass auf dem Konto und im Bankautomaten genügend Geld vorhanden ist.

Lösungsvorschlag



**46116 / 2017 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3**

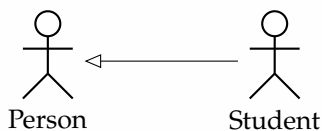
Betrachten Sie die folgenden UML-Diagramme. Sind diese korrekt? Falls nein, begründen Sie warum nicht. Geben Sie in diesem Fall außerdem eine korrigierte Version an. Falls ja, erklären Sie die inhaltliche Bedeutung des Diagramms.

(a)

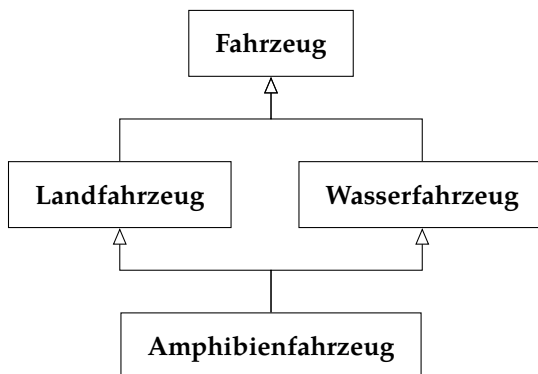


Lösungsvorschlag

*Falsch*, den verwendeten „extends“-Pfeil gibt es nur zwischen Anwendungsfällen.



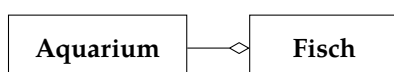
(b)



Lösungsvorschlag

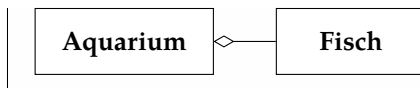
Die dargestellte Modellierung ist *korrekt*. Sowohl Land- als auch Wasserfahrzeuge sind Fahrzeuge und erben somit von dieser Klasse. Da ein Amphibienfahrzeug eine „Mischung“ aus beidem ist, erbt diese Klasse auch von beiden Klassen. Diese Mehrfachvererbung kann allerdings nicht in jeder Programmiersprache (z. B. nicht in Java) umgesetzt werden.

(c)



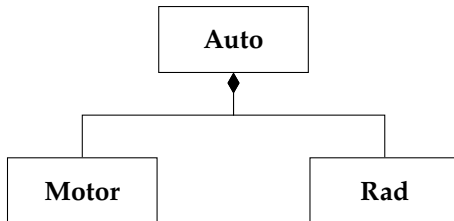
Lösungsvorschlag

Bei der dargestellten Aggregation befindet sich die Raute an der *falschen* Seite der Beziehung. Das Diagramm würde bedeuten, dass ein Fisch mehrere Aquarien enthält. Die Umkehrung ist aber korrekt:



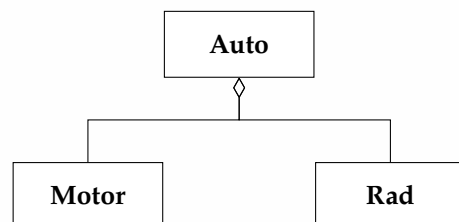
Gantt-Diagramm

(d)



Lösungsvorschlag

Hier wurde für die Modellierung eine Komposition gewählt. Dies bedeutet, dass die Existenz der Teile vom Ganzen abhängt. Einen Raum kann es z. B. ohne ein Gebäude nicht geben. In diesem Fall ist die Darstellung *falsch*, da Motor und Rad auch ohne Auto existieren können. Die Modellierung muss also mittels Aggregation erfolgen:



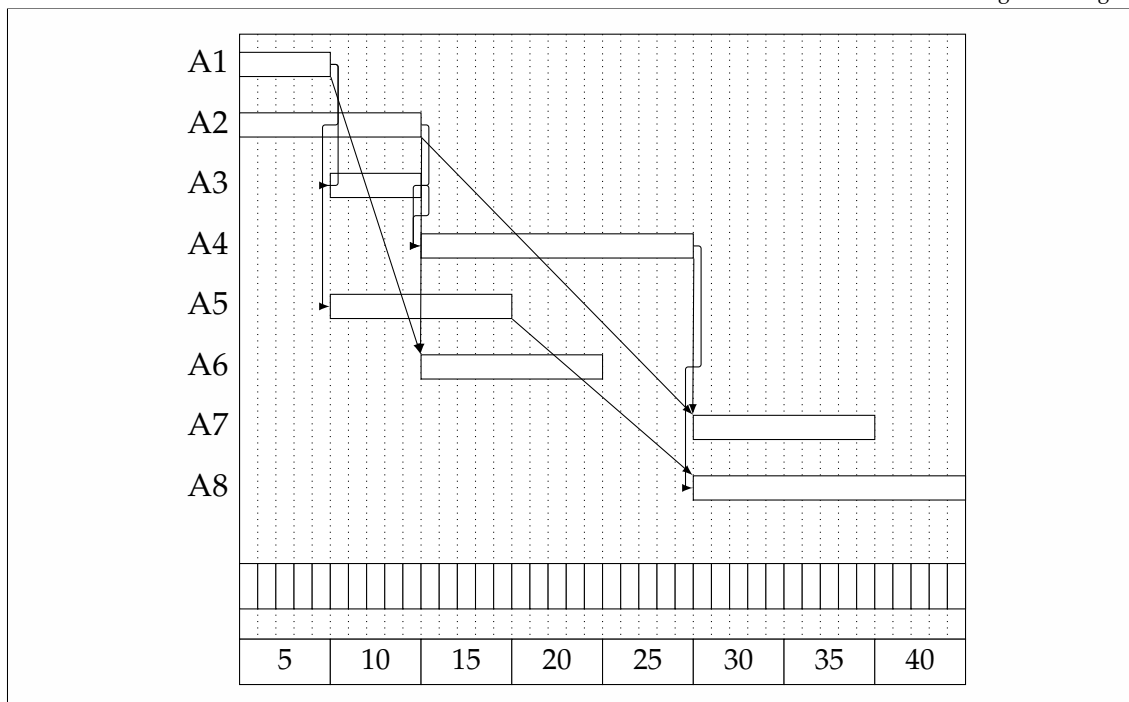
## 46116 / 2017 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 5

Betrachten Sie die folgende Tabelle zum Projektmanagement:

| Name | Dauer (Tage) | Abhängig von |
|------|--------------|--------------|
| A1   | 5            |              |
| A2   | 10           |              |
| A3   | 5            | A1           |
| AA   | 15           | A2, A3       |
| AS   | 10           | A1           |
| A6   | 10           | A1, A2       |
| A7   | 10           | A2, A4       |
| A8   | 15           | A4, A5       |

Tabelle 1: Übersicht Arbeitspakete

- (a) Erstellen Sie ein Gantt-Diagramm, das die in der Tabelle angegebenen Abhängigkeiten berücksichtigt.



(b) Wie lange dauert das Projekt mindestens?

Lösungsvorschlag

40 Tage

(c) Geben Sie den oder die kritischen Pfad(e) an.

Lösungsvorschlag

A2 A4 A8

A1 A3 A4 A8

(d) Konstruieren Sie ein PERT-Chart zum obigen Problem.

## 46116 / 2017 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 4

(a) Erläutern Sie in ein bis zwei Sätzen, aus welchen zwei Teilen sich ein TID (Tupel-identifikator) zusammensetzt.

Lösungsvorschlag

Seitennummer (Seiten bzw. Blöcke sind größere Speichereinheiten auf der Platte) Relative Indexposition innerhalb der Seite

(b) Erläutern Sie in ein bis zwei Sätzen das Vorgehen, wenn ein durch einen TID adressierter Satz innerhalb einer Seite verschoben werden muss.

Satzverschiebung innerhalb einer Seite bleibt ohne Auswirkungen auf TID,

- (c) Erläutern Sie in ein bis zwei Sätzen das Vorgehen, wenn ein durch einen TID adressierter Satz erstmalig in eine andere Seite verschoben werden muss.

Lösungsvorschlag

wird ein Satz auf eine andere Seite migriert, wird eine „Stellvertreter-TID“ zum Verweis auf den neuen Speicherort verwendet. Die eigentliche TID-Adresse bleibt stabil

- (d) Erläutern Sie in zwei bis drei Sätzen das Vorgehen, wenn ein durch einen TID adressierter und bereits einmal über Seitengrenzen hinweg verschobener Satz erneut in eine andere Seite verschoben werden muss.

Lösungsvorschlag

Es wird eine neue stellvertreter TID aktualisiert.

## 46116 / 2017 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4

Ein gängiger Ansatz zur Messung der Qualität von Software ist das automatisierte Testen von Programmen. Im Folgenden werden praktische Testmethoden anhand des nachstehend angegebenen Sortieralgorithmus diskutiert.

### Algorithmus 1 Bubble Sort

```
public class BubbleSort {
 void bubblesort(int[] array, int len) {
 for (int i = 0; i < len - 1; i++) { // 1
 for (int j = 0; j < len - 1; j++) { // 2
 if (array[j] > array[j + 1]) { // 3
 int temp = array[j]; // 4
 array[j] = array[j + 1]; // 5
 array[j + 1] = temp; // 6
 }
 }
 }
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2017/herbst/BubbleSort.java](https://github.com/bschlangaul/examen/examen_46116/jahr_2017/herbst/BubbleSort.java)

- (a) Nennen Sie eine Art des Black-Box-Testens und beschreiben Sie deren Durchführung anhand des vorgegebenen Algorithmus.

Lösungsvorschlag

Beim Black-Box-Testen sind die Testfälle von Daten getrieben (Data-Driven) und beziehen sich auf die Anforderungen und das spezifizierte Verhalten.)

⇒ Aufruf der Methoden mit verschiedenen Eingangsparametern und Vergleich der erhaltenen Ergebnisse mit den erwarteten Ergebnissen.

Das Ziel ist dabei eine möglichst hohe Anforderungsüberdeckung, wobei man

eine minimale Anzahl von Testfällen durch Äquivalenzklassenzerlegung (1) und Grenzwertanalyse (2) erhält.

**zu (1):** Man identifiziert Bereiche von Eingabewerten, die jeweils dieselben Ergebnisse liefern. Dies sind die sog. Äquivalenzklassen. Aus diesen wählt man nun je einen Repräsentanten und nutzt diesen für den Testfall.

**zu (2):** Bei der Grenzwertanalyse identifiziert man die Grenzbereiche der Eingabedaten und wählt Daten aus dem nahen Umfeld dieser für seine Testfälle.

Angewendet auf den gegebenen Bubblesort-Algorithmus würde die Grenzwertanalyse bedeuten, dass man ein bereits aufsteigend sortiertes Array und ein absteigend sortiertes Array übergibt.

- (b) Zeichnen Sie ein mit Zeilennummern beschriftetes Kontrollflussdiagramm für den oben angegebenen Sortieralgorithmus.

Lösungsvorschlag

Zur Erinnerung: Eine im Code enthaltene Wiederholung mit `for` muss wie folgt im Kontrollflussgraphen „zerlegt“ werden:

- (c) Erklären Sie, ob eine vollständige Pfadüberdeckung für die gegebene Funktion möglich und sinnvoll ist.

Lösungsvorschlag

Eine vollständige Pfadüberdeckung ( $C_1$ -Test) kann nicht erreicht werden, da die Bedingung der inneren Wiederholung immer wahr ist, wenn die Bedingung der äußeren Wiederholung wahr ist. D. h., der Pfad S-1-1-2-2-1“ kann nie gegangen werden. Dies wäre aber auch nicht sinnvoll, weil jeder Eintrag mit jedem anderen verglichen werden soll und im Fall `true` → `false` ein Durchgang ausgelassen.

## 46116 / 2017 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3

Verwenden Sie geeignete **Entwurfsmuster**, um die folgenden Sachverhalte mit Hilfe von **UML-Klassendiagrammen** zu beschreiben. Nennen Sie das zu verwendende Entwurfsmuster namentlich, wenden Sie es zur Lösung der jeweiligen *Fragestellung* an und erstellen Sie damit das problemspezifische UML-Klassendiagramm. Beschränken Sie sich dabei auf die statische Sicht, definieren Sie keinerlei Verhalten mit Ausnahme der Definition geeigneter Operationen.

- (a) Es gibt unterschiedliche Arten von Bankkonten: Girokonto, Bausparkonto und Kreditkarte. Bei allen Konten ist der Name des Inhabers hinterlegt. Girokonten haben eine IBAN. Kreditkarten sind immer mit einem Girokonto verknüpft. Bei Bausparkonten werden ein Sparzins sowie ein Darlehenszins festgelegt. Es gibt eine *zentrale* Klasse, die die *Erzeugung* unterschiedlicher Typen von Bankkonten steuert.



- (b) Beim Ticker für ein Hockeyspiel können sich verschiedene Geräte registrieren und wieder abmelden, um auf *Veränderungen* des Spielstands zu *reagieren*. Hierzu werden im Ticker die Tore der Heim- vmd Gastmannschaft sowie die aktuelle Spielminute vermerkt. Als konkrete Geräte sind eine Smartphone-App sowie eine Stadionuhr bekannt.
- (c) Dateisysteme sind *baumartig* strukturiert. Verzeichnisse können wiederum selbst Verzeichnisse und/oder Dateien beinhalten. Sowohl Dateien als auch Verzeichnisse haben einen Namen. Das jeweilige Elternverzeichnis ist eindeutig. Bei Dateien wird die Art (Binär, Text oder andere) sowie die Größe in Byte, bei Verzeichnissen die Anzahl enthaltener Dateien hinterlegt.

## 46116 / 2017 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 4

Für die bayerische Meisterschaft im Turmspringen ist folgendes Datenbankschema angelegt:

Springer : {[ Startnummer, Nachname, Vorname, Geburtsdatum, Körpergröße ]}

Sprung : {[ SID, Beschreibung, Schwierigkeit ]}

springt : {[ SID, Startnummer, Durchgang ]}

FK (SID) referenziert Sprung (SID)

FK (Startnummer) referenziert Springer (Startnummer)

Das Attribut Schwierigkeit kann die Werte 1 bis 10 annehmen, das Attribut Durchgang ist positiv und ganzzahlig. Die Körpergröße der Springer ist in Zentimeter angegeben.

- (a) Welche Springer sind größer als 1,80 m? Schreiben Sie eine SQL-Anweisung, welche in der Ausgabe mit dem größten Springer beginnt.

Lösungsvorschlag

```
SELECT Vorname, Nachname, Körpergröße
FROM Springer
WHERE Körpergröße > 180
ORDER BY Körpergröße DESC;
```

- (b) Welche Springer haben im ersten Durchgang einen Sprung mit einer Schwierigkeit von unter 6 gezeigt? Schreiben Sie eine SQL-Anweisung, welche Startnummer und Nachname dieser Springer ausgibt.

Lösungsvorschlag

```
SELECT Springer.Startnummer, Springer.Nachname
FROM Springer, Sprung, springt
WHERE
 Sprung.SID = springt.SID AND
 Springer.Startnummer = springt.Startnummer AND
 springt.Durchgang = 1 AND
 Sprung.Schwierigkeit < 6;
```

- (c) Formulieren Sie in Umgangssprache, aber trotzdem möglichst präzise, wonach mit folgender Abfrage gesucht wird:

```
SELECT springt.Startnummer, s.Nachname, s.Vorname, MAX(springt.Durchgang)
FROM springt, Springer s
WHERE springt.Startnummer = s.Startnummer
GROUP BY springt.Startnummer, s.Nachname, s.Vorname
```

Lösungsvorschlag

Die Abfrage gibt die Startnummer, den Nachnamen, den Vornamen und die Anzahl der Sprünge, d. h. die Anzahl der Durchgänge der einzelnen Springer an.

- (d) Gesucht ist die „durchschnittliche Körpergröße“ all der Springer, die vor dem 01.01.2000 geboren wurden. Formulieren Sie eine SQL-Anweisung, wobei die Spalte mit der durchschnittlichen Körpergröße genau diesen Namen „durchschnittliche Körpergröße“ haben soll.

Lösungsvorschlag

Umlaute und Leerzeichen sind bei Spaltenbeschriftungen nicht erlaubt.

```
SELECT AVG(Körpergröße) AS durchschnittliche_Koerpergroesse
FROM SPRINGER
WHERE Geburtsdatum < DATE('2000-01-01');
```

oder

Lösungsvorschlag

```
SELECT AVG(Körpergröße) AS durchschnittliche_Koerpergroesse
FROM SPRINGER
WHERE Geburtsdatum < '01.01.2000';
```

## 46116 / 2018 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Eine Digitaluhr kann alternativ entweder die Zeit (Stunden und Minuten) oder das Datum (Tag, Monat und Jahr) anzeigen. Zu Beginn zeigt die Uhr die Zeit an. Sie besitzt drei Druckknöpfe **A**, **B** und **C**. Mit Knopf **A** kann zwischen Zeit- und Datumsanzeige hin und her gewechselt werden.

Wird die Zeit angezeigt, dann kann mit Knopf **B** der Reihe nach erst in einen Stundenmodus, dann in einen Minutenmodus und schließlich zurück zur Zeitanzeige gewechselt werden. Im Stundenmodus blinkt die Stundenanzeige. Mit Drücken des Knopfes **C** können dann die Stunden schrittweise inkrementiert werden. Im Minutenmodus blinkt die Minutenanzeige und es können mit Hilfe des Knopfes **C** die Minuten schrittweise inkrementiert werden.

Die Datumsfunktionen sind analog. Wird das Datum angezeigt, dann kann mit Knopf **B** der Reihe nach in einen Tagesmodus, Monatsmodus, Jahresmodus und schließlich zurück zur Datumsanzeige gewechselt werden. Im Tagesmodus blinkt die Tagesanzeige. Mit Drücken des Knopfes **C** können dann die Tage schrittweise inkrementiert

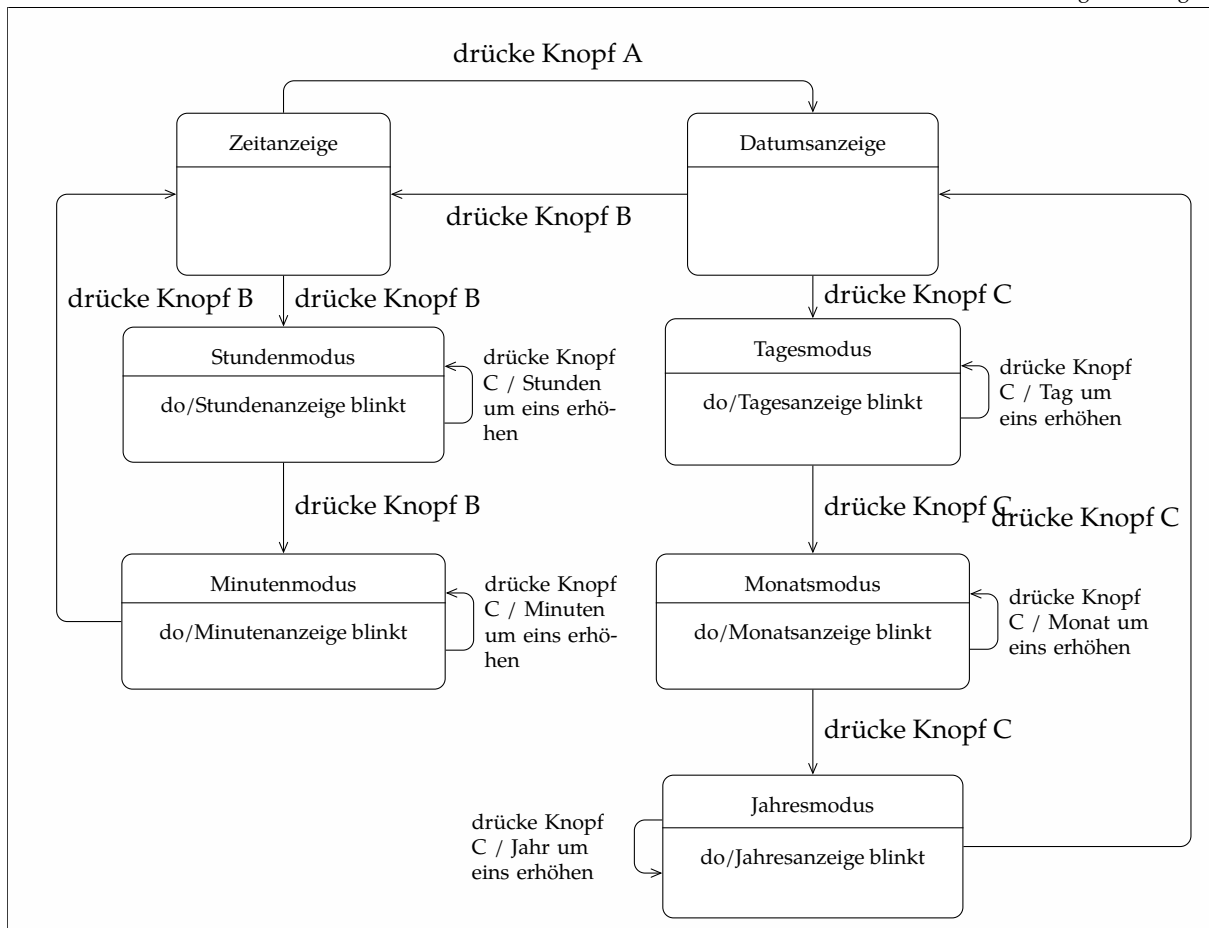
werden. Analog blinken mit Eintritt in den entsprechenden Einstellmodus der Monat oder das Jahr, die dann mit Knopf C schrittweise inkrementiert werden können.

Wenn sich die Uhr in einem Einstellmodus befindet, hat das Betätigen des Knopfes A keine Wirkung. Ebenso wirkungslos ist Knopf C, wenn gerade Zeit oder Datum angezeigt wird.

Beschreiben Sie das Verhalten der Digitaluhr durch ein UML-Zustandsdiagramm. Dabei muss - gemäß der UML-Notation - unterscheidbar sein, was Ereignisse und was Aktionen sind. Deren Bedeutung soll durch die Verwendung von sprechenden Namen klar sein. Für die Inkrementierung von Stunden, Minuten, Tagen etc. brauchen keine konkreten Berechnungen angegeben werden. Der kontinuierliche Zeitfortschritt des Uhrwerks ist nicht zu modellieren.

Zustände sind, wie in der UML üblich, durch abgerundete Rechtecke darzustellen. Sie können unterteilt werden in eine obere und eine untere Hälfte, wobei der Name des Zustands in den oberen Teil und eine in dem Zustand auszuführende Aktivität in den unteren Teil einzutragen ist.

Lösungsvorschlag



## 46116 / 2018 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4

Gegeben sind folgende Relationen aus einem Kundenverwaltungssystem:

Kunde : {[ ID, Vorname, Nachname, PLZ ]}

Produkt : {[ GTIN, Bezeichnung, Bruttopreis, MWStSatz) ]}

Kauf : {[ ID[Kunde], GTIN[Produkt], Datum, Menge ]}

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz, also eine Entity, nicht mehrfach aus.

- (a) Schreiben Sie eine SQL-Anweisung, die die Tabelle „Kauf“ anlegt. Gehen Sie davon aus, dass die Tabellen „Kunde“ und „Produkt“ bereits existieren.

Lösungsvorschlag

```
CREATE TABLE IF NOT EXISTS Kauf (
 ID INTEGER REFERENCES Kunde(ID),
 GTIN INTEGER REFERENCES Produkt(GTIN),
 Datum DATE,
 Menge INTEGER,
 PRIMARY KEY (ID, GTIN, Datum)
);
```

- (b) Schreiben Sie eine SQL-Anweisung, die *Vorname* und *Nachname* aller *Kunden* mit der *Postleitzahl* 20251 ausgibt, absteigend sortiert nach *Nachname* und bei gleichen *Nachnamen*, absteigend nach *Vorname*.

Lösungsvorschlag

```
SELECT Vorname, Nachname
FROM Kunde
WHERE PLZ = 20251
ORDER BY Nachname DESC, Vorname DESC;
```

```
vorname | nachname
-----+-----
Hanna | Winter
Jakob | Sommer
Bert | Sommer
(3 rows)
```

- (c) Schreiben Sie eine SQL-Anweisung, die zu jedem Einkauf mit mehr als 10 unterschiedlichen Produkten den *Nachnamen* des *Kunden* und den *Bruttogesamtpreis* des Einkaufs ausgibt. Ein Einkauf ist definiert als Menge aller Produkte, die ein bestimmter Kunde an einem bestimmten Datum kauft.

Lösungsvorschlag

```
SELECT Nachname, SUM(Bruttopreis * Menge)
FROM Kunde k, Produkt p, Kauf x
WHERE k.ID = x.ID AND p.GTIN = x.GTIN
GROUP BY Datum, Nachname, k.ID
HAVING COUNT (*) > 10;
```

```
nachname | sum
-----+-----
```

Mustermann | 713.86  
(1 row)

HAVING  
EXCEPT

- (d) Schreiben Sie eine SQL-Anweisung, die die *GTINs* aller Produkte ausgibt, die an mindestens einen in der Datenbank enthaltenen PLZ-Bereich noch nie verkauft worden sind. Als in der Datenbank enthaltener PLZ-Bereich gelten alle in der Tabelle „Kunde“ enthaltenen PLZs. Ein Produkt gilt als an einen PLZ-Bereich verkauft, sobald es von mindestens einem Kunden aus diesem PLZ-Bereich gekauft wurde. Produkte, die bisher noch gar nicht verkauft worden sind, müssen nicht berücksichtigt werden.

Lösungsvorschlag

Die beiden Lösungswege liefern leider unterschiedliche Ergebnisse.

```
WITH tmp AS (
 SELECT x.GTIN, k.PLZ
 FROM Kunde k, Kauf x
 WHERE x.ID = k.ID
 GROUP BY x.GTIN, k.PLZ
)

SELECT DISTINCT GTIN
FROM tmp
WHERE EXISTS (
 SELECT Kunde.PLZ
 FROM Kunde LEFT OUTER JOIN tmp
 ON Kunde.PLZ = tmp.PLZ
 WHERE tmp.PLZ IS NULL
)
ORDER BY GTIN;
```

```
gtin

 4
 23
112
113
123
124
125
155
189
324
453
765
(12 rows)
```

oder

```

SELECT DISTINCT GTIN FROM (
 (
 SELECT GTIN, PLZ
 FROM Kunde, Produkt
)
 EXCEPT
 (
 SELECT x.GTIN, k.PLZ
 FROM Kunde k, Kauf x
 WHERE x.ID = k.ID
 GROUP BY x.GTIN, k.PLZ
)
) as tmp
ORDER BY GTIN;

```

```

gtin

 4
 23
112
113
123
124
125
155
189
324
453
765
889
(13 rows)

```

- (e) Schreiben Sie eine SQL-Anweisung, die die Top-Ten der am meisten verkauften Produkte ausgibt. Ausgegeben werden sollen der Rang (1 bis 10) und die Bezeichnung des Produkts. Gehen Sie davon aus, dass es keine zwei Produkte mit gleicher Verkaufszahl gibt und verwenden Sie keine produktspezifischen Anweisungen wie beispielsweise ROWNUM, TOP oder LIMIT.

Lösungsvorschlag

```

WITH Gesamtverkauf AS (
 SELECT k.GTIN, Bezeichnung, SUM(Menge) AS Gesamtmenge
 FROM Produkt p, Kauf k
 WHERE p.GTIN = k.GTIN
 GROUP BY k.GTIN, Bezeichnung
)

SELECT g1.Bezeichnung, COUNT (*) AS Rang
FROM Gesamtverkauf g1, Gesamtverkauf g2
WHERE g1.Gesamtmenge <= g2.Gesamtmenge
GROUP BY g1.GTIN, g1.Bezeichnung

```

```
HAVING COUNT (*) <= 10
ORDER BY Rang;
```

| bezeichnung  | rang |
|--------------|------|
| Topf         | 1    |
| Kaffee       | 2    |
| Sonnenbrille | 3    |
| T-Shirt      | 4    |
| Klopapier    | 5    |
| Duschgel     | 6    |
| Hammer       | 7    |
| Heft         | 8    |

(8 rows)

DELETE  
Entity-Relation-Modell

- (f) Schreiben Sie eine SQL-Anweisung, die alle Produkte löscht, die noch nie gekauft wurden.

Lösungsvorschlag

```
count

 13
(1 row)

SELECT COUNT(*) FROM Produkt;

DELETE FROM Produkt
WHERE GTIN NOT IN
(
 SELECT DISTINCT GTIN
 FROM Kauf
);

SELECT COUNT(*) FROM Produkt;

count

 12
(1 row)
```

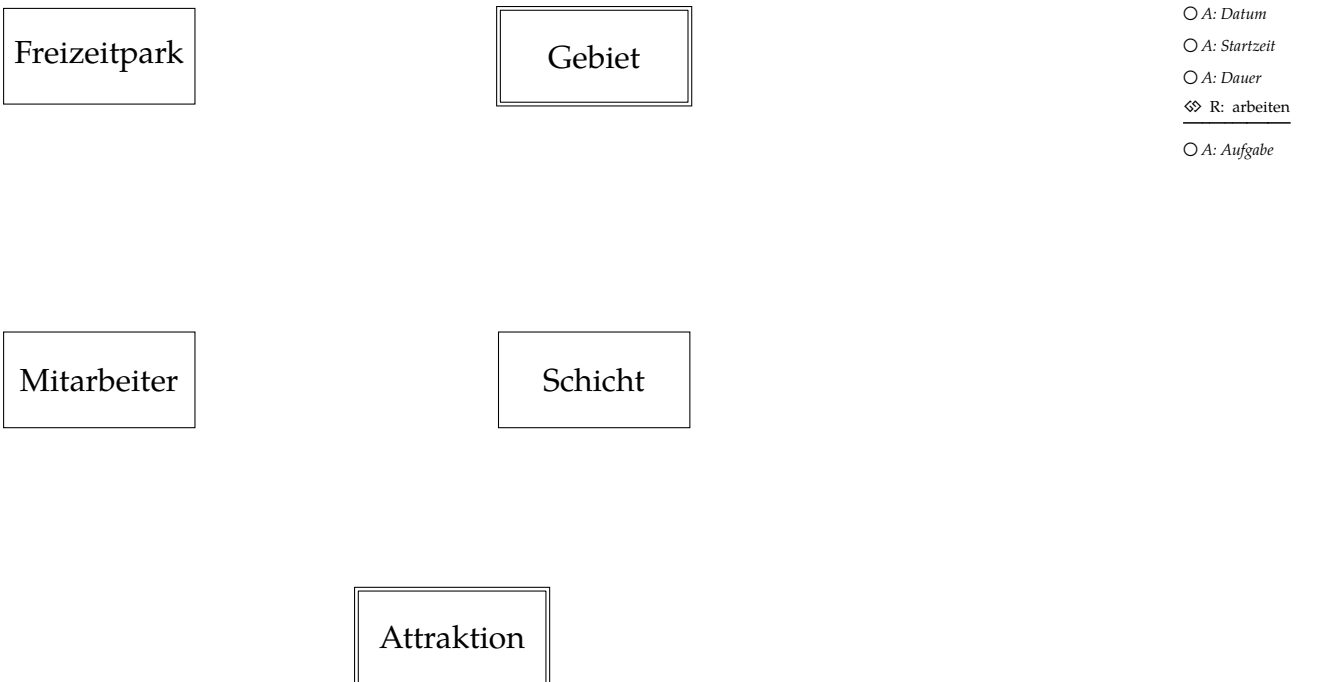
## 46116 / 2018 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 2

Im Folgenden finden Sie die Beschreibung eines Systems zur Verwaltung von Freizeitparks. Erstellen Sie zu dieser Beschreibung ein erweitertes ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Kennzeichnen Sie auch die totale Teilnahme (= Existenzabhängigkeit, Partizipität) von Entitytypen.

- Der **Freizeitpark** ist in mehrere Gebiete eingeteilt.

□ E: Freizeitpark  
⋈ R: eingeteilt

- Ein **Gebiet** hat einen eindeutigen *Namen* und eine *Beschreibung*.
- In jedem Gebiet gibt es eine oder mehrere **Attraktionen**. Diese verfügen über eine innerhalb ihres Gebiets eindeutige *Nummer*. Außerdem gibt es zu jeder Attraktion einen *Namen*, eine *Beschreibung* und ein oder mehrere Fotos.
- Der Freizeitpark hat **Mitarbeiter**. Zu diesen werden jeweils eine eindeutige *ID*, der *Vorname* und der *Nachname* gespeichert. Weiterhin hat jeder Mitarbeiter ein *Geburtsdatum*, das sich aus *Tag*, *Monat* und *Jahr* zusammensetzt.
- Die Arbeit im Freizeitpark ist in **Schichten** organisiert. Eine Schicht kann eindeutig durch das *Datum* und die *Startzeit* identifiziert werden. Jede Schicht hat weiterhin eine *Dauer*.
- Mitarbeiter können in Schichten an Attraktionen arbeiten. Dabei wird die *Aufgabe* gespeichert, die der Mitarbeiter übernimmt. Pro Schicht kann der selbe Mitarbeiter nur an maximal einer Attraktion arbeiten.



- ☐ E: Gebiet
- ☐ A: Namen
- ☐ A: Beschreibung
- ☐ R: gibt
- ☐ E: Attraktionen
- ☐ A: Nummer
- ☐ A: Namen
- ☐ A: Beschreibung
- ☐ R: hat
- ☐ E: Mitarbeiter
- ☐ A: ID
- ☐ A: Vorname
- ☐ A: Nachname
- ☐ A: Geburtsdatum
- ☐ A: Tag
- ☐ A: Monat
- ☐ A: Jahr
- ☐ E: Schichten
- ☐ A: Datum
- ☐ A: Startzeit
- ☐ A: Dauer
- ☐ R: arbeiten
- ☐ A: Aufgabe

## 46116 / 2018 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 4

Gegeben sind folgende Relationen aus einem Kundenverwaltungssystem:

Kunde (ID, Vorname, Nachname, PLZ)  
 Produkt (GTIN, Bezeichnung, Bruttopreis, MWStSatz)  
 Kauf (ID[Kunde], GTIN[Produkt], Datum, Menge)

- (a) Schreiben Sie eine SQL-Anweisung, die *Vorname* und *Nachname* aller Kunden mit der Postleitzahl 20251 ausgibt, *absteigend* sortiert nach *Nachname* und bei gleichen Nachnamen absteigend nach *Vorname*.

ASC (ascending) = aufsteigend DESC (descending) = absteigend



Lösungsvorschlag

CREATE TABLE  
Relationale Algebra

```

SELECT Vorname, Nachname
FROM Kunde
WHERE PLZ = 20251
ORDER BY Nachname, Vorname DESC;

```

- (b) Schreiben Sie eine SQL-Anweisung, die die Bezeichnung aller Produkte ausgibt, deren Bruttopreis größer ist als 10 €.

Lösungsvorschlag

```

SELECT Bezeichnung
FROM Produkt
WHERE Bruttopreis > 10;

```

- (c) Schreiben Sie eine SQL-Anweisung, die die Tabelle „Kauf“ anlegt. Gehen Sie davon aus, dass die Tabellen „Kunde“ und „Produkt“ bereits existieren.

Lösungsvorschlag

```

CREATE TABLE Kauf (
 ID INTEGER REFERENCES Kunde(ID),
 GTIN INTEGER REFERENCES Produkt(GTIN),
 Datum DATE,
 Menge INTEGER,
 PRIMARY KEY (ID, GTIN, Datum)
);

```

## 46116 / 2018 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Geben Sie die Ergebnisrelation folgender Ausdrücke der relationalen Algebra als Tabellen an. Begründen Sie Ihr Ergebnis, gegebenenfalls durch Zwischenschritte. Gegeben seien folgende Relationen:

| R |   |   |   |   |   | S |   |   |   | T |   |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | A | C | X | Z | X | Y |
| 6 | 8 | 1 | 7 | 3 | 7 | 7 | 8 | 6 | 1 | 5 | 3 |
| 5 | 3 | 4 | 4 | 5 | 7 | 0 | 3 | 0 | 0 | 0 | 5 |
| 0 | 6 | 3 | 0 | 1 | 7 | 2 | 3 | 0 | 5 | 8 | 6 |
|   |   |   |   |   |   | 0 | 6 | 1 | 6 | 3 | 6 |
|   |   |   |   |   |   | 6 | 7 | 1 | 7 | 5 | 7 |
|   |   |   |   |   |   | 7 | 1 | 2 | 2 | 2 | 8 |
|   |   |   |   |   |   | 1 | 8 | 8 | 0 |   |   |
|   |   |   |   |   |   | 5 | 1 | 5 | 5 |   |   |
|   |   |   |   |   |   | 7 | 3 | 0 | 2 |   |   |
|   |   |   |   |   |   | 4 | 8 | 2 | 7 |   |   |

(a)  $\sigma_{A>6}(S) \bowtie_{S.X=T.Y} \pi_Y(T)$

Lösungsvorschlag

| A | C | X | Z | Y |
|---|---|---|---|---|
| 7 | 8 | 6 | 1 | 6 |

(b)  $\pi_{A,C}(S) - (\pi_A(R) \times \pi_C(\sigma_{x=1}(S)))$

Lösungsvorschlag

| $\sigma_{x=1}(S):$                                                                                                                                                                                                         | $\pi_C(\sigma_{x=1}(S)):$ | $\pi_A(R):$ |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|-------------|---|---|---|---|---|---|---|---|---|---|--------------------------------------------------------------------------|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---------------------------------------------------------------------------------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|
| <table><tr><th>A</th><th>C</th><th>X</th><th>Z</th></tr><tr><td>0</td><td>6</td><td>1</td><td>6</td></tr><tr><td>6</td><td>7</td><td>1</td><td>7</td></tr></table>                                                         | A                         | C           | X | Z | 0 | 6 | 1 | 6 | 6 | 7 | 1 | 7 | <table><tr><th>C</th></tr><tr><td>6</td></tr><tr><td>7</td></tr></table> | C | 6                                                                                                                                                                                                                                                                                                                                              | 7 | <table><tr><th>A</th></tr><tr><td>6</td></tr><tr><td>5</td></tr><tr><td>0</td></tr></table> | A | 6 | 5 | 0 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| A                                                                                                                                                                                                                          | C                         | X           | Z |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 0                                                                                                                                                                                                                          | 6                         | 1           | 6 |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 6                                                                                                                                                                                                                          | 7                         | 1           | 7 |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| C                                                                                                                                                                                                                          |                           |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 6                                                                                                                                                                                                                          |                           |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 7                                                                                                                                                                                                                          |                           |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| A                                                                                                                                                                                                                          |                           |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 6                                                                                                                                                                                                                          |                           |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 5                                                                                                                                                                                                                          |                           |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 0                                                                                                                                                                                                                          |                           |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| $(\pi_A(R) \times \pi_C(\sigma_{x=1}(S)))$                                                                                                                                                                                 | $\pi_{A,C}(S)$            |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| <table><tr><th>A</th><th>C</th></tr><tr><td>6</td><td>6</td></tr><tr><td>5</td><td>6</td></tr><tr><td>0</td><td>6</td></tr><tr><td>6</td><td>7</td></tr><tr><td>5</td><td>7</td></tr><tr><td>0</td><td>7</td></tr></table> | A                         | C           | 6 | 6 | 5 | 6 | 0 | 6 | 6 | 7 | 5 | 7 | 0                                                                        | 7 | <table><tr><th>A</th><th>C</th></tr><tr><td>7</td><td>8</td></tr><tr><td>0</td><td>3</td></tr><tr><td>2</td><td>3</td></tr><tr><td>0</td><td>6</td></tr><tr><td>6</td><td>7</td></tr><tr><td>7</td><td>1</td></tr><tr><td>1</td><td>8</td></tr><tr><td>5</td><td>1</td></tr><tr><td>7</td><td>3</td></tr><tr><td>4</td><td>8</td></tr></table> | A | C                                                                                           | 7 | 8 | 0 | 3 | 2 | 3 | 0 | 6 | 6 | 7 | 7 | 1 | 1 | 8 | 5 | 1 | 7 | 3 | 4 | 8 |  |
| A                                                                                                                                                                                                                          | C                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 6                                                                                                                                                                                                                          | 6                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 5                                                                                                                                                                                                                          | 6                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 0                                                                                                                                                                                                                          | 6                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 6                                                                                                                                                                                                                          | 7                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 5                                                                                                                                                                                                                          | 7                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 0                                                                                                                                                                                                                          | 7                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| A                                                                                                                                                                                                                          | C                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 7                                                                                                                                                                                                                          | 8                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 0                                                                                                                                                                                                                          | 3                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 2                                                                                                                                                                                                                          | 3                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 0                                                                                                                                                                                                                          | 6                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 6                                                                                                                                                                                                                          | 7                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 7                                                                                                                                                                                                                          | 1                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 1                                                                                                                                                                                                                          | 8                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 5                                                                                                                                                                                                                          | 1                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 7                                                                                                                                                                                                                          | 3                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| 4                                                                                                                                                                                                                          | 8                         |             |   |   |   |   |   |   |   |   |   |   |                                                                          |   |                                                                                                                                                                                                                                                                                                                                                |   |                                                                                             |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |

| A | C |
|---|---|
| 7 | 8 |
| 0 | 3 |
| 2 | 3 |
| 7 | 1 |
| 1 | 8 |
| 5 | 1 |
| 7 | 3 |
| 4 | 8 |

SQL

$$(c) (\pi_D(R) \times \pi_E(R)) \div \pi_E(R)$$

Lösungsvorschlag

| $\pi_D(R) \times \pi_E(R)$ |   | $\pi_E(R)$ | $(\pi_D(R) \times \pi_E(R)) \div \pi_E(R)$ |
|----------------------------|---|------------|--------------------------------------------|
| A                          | E | E          | D                                          |
| 7                          | 3 | 3          | 7                                          |
| 4                          | 3 | 5          | 4                                          |
| 0                          | 3 | 1          | 0                                          |
| 7                          | 5 |            |                                            |
| 4                          | 5 |            |                                            |
| 0                          | 5 |            |                                            |
| 7                          | 1 |            |                                            |
| 4                          | 1 |            |                                            |
| 0                          | 1 |            |                                            |

### 46116 / 2018 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 3

Gegeben sei das folgende Datenbank-Schema, das für die Speicherung der Daten einer Schule entworfen wurde, zusammen mit einem Teil seiner Ausprägung. Die Primärschlüssel-Attribute sind jeweils unterstrichen.

Die Relation *Schüler* enthält allgemeine Daten zu den Schülerinnen und Schülern. Schülerinnen und Schüler nehmen an Prüfungen in verschiedenen Unterrichtsfächern teil und erhalten dadurch Noten. Diese werden in der Relation *Noten* abgespeichert. Prüfungen haben ein unterschiedliches Gewicht. Beispielsweise hat ein mündliches Ausfragen oder eine Extemporale das Gewicht 1, während eine Schulaufgabe das Gewicht 2 hat.

#### Schüler:

CREATE TABLE  
INSERT

| <u>SchülerID</u> | Vorname | Nachname  | Klasse |
|------------------|---------|-----------|--------|
| 1                | Laura   | Müller    | 4A     |
| 2                | Linus   | Schmidt   | 4A     |
| 3                | Jonas   | Schneider | 4A     |
| 4                | Liam    | Fischer   | 4B     |
| 5                | Tim     | Weber     | 4B     |
| 6                | Lea     | Becker    | 4B     |
| 7                | Emilia  | Klein     | 4C     |
| 8                | Julia   | Wolf      | 4C     |

**Noten:**

| <u>SchülerID</u> [Schüler] | Schulfach  | Note | Gewicht | Datum      |
|----------------------------|------------|------|---------|------------|
| 1                          | Mathematik | 3    | 2       | 23.09.2017 |
| 1                          | Mathematik | 1    | 1       | 03.10.2017 |
| 1                          | Mathematik | 2    | 2       | 15.10.2017 |
| 1                          | Mathematik | 4    | 1       | 11.11.2017 |

- (a) Geben Sie die SQL-Befehle an, die notwendig sind, um die oben dargestellten Tabellen in einer SQL-Datenbank anzulegen.

Lösungsvorschlag

```

CREATE TABLE IF NOT EXISTS Schüler (
 SchülerID INTEGER PRIMARY KEY NOT NULL,
 Vorname VARCHAR(20),
 Nachname VARCHAR(20),
 Klasse VARCHAR(5)
);

CREATE TABLE IF NOT EXISTS Noten (
 SchülerID INTEGER NOT NULL,
 Schulfach VARCHAR(20),
 Note INTEGER,
 Gewicht INTEGER,
 Datum DATE,
 PRIMARY KEY (SchülerID, Schulfach, Datum),
 FOREIGN KEY (SchülerID) REFERENCES Schüler(SchülerID)
);

```

- (b) Entscheiden Sie jeweils, ob folgende Einfügeoperationen vom gegebenen Datenbanksystem (mit der angegebenen Ausprägungen) erfolgreich verarbeitet werden können und begründen Sie Ihre Antwort kurz.

```
INSERT INTO Schüler
(SchülerID, Vorname, Nachname, Klasse)
VALUES
(6, 'Johannes', 'Schmied', '4C');
```

Lösungsvorschlag

Nein. Ein/e Schüler/in mit der ID 6 existiert bereits. Primärschlüssel müssen eindeutig sein.

```
INSERT INTO Noten VALUES (6, 'Chemie', 1, 2, '1.4.2020');
```

Lösungsvorschlag

Nein. Ein *Datum* ist zwingend notwendig. Da *Datum* im Primärschlüssel enthalten ist, darf es nicht NULL sein. Es gibt auch keine/n Schüler/in mit der ID 9. Der/die Schüler/in müsste vorher angelegt werden, da die Spalte *SchülerID* von der Tabelle *Noten* auf den Fremdschlüssel *SchülerId* aus der Schülertabelle verweist.

- (c) Geben Sie die Befehle für die folgenden Aktionen in SQL an. Beachten Sie dabei, dass die Befehle auch noch bei Änderungen des oben gegebenen Datenbankzustandes korrekte Ergebnisse zurückliefern müssen.

- Die Schule möchte verhindern, dass in die Datenbank mehrere Kinder mit dem selben Vornamen in die gleiche Klasse kommen. Dies soll bereits auf Datenbankebene verhindert werden. Dabei sollen die Primärschlüssel nicht verändert werden. Geben Sie den Befehl an, der diese Änderung durchführt.

Lösungsvorschlag

```
ALTER TABLE Schüler
ADD CONSTRAINT eindeutiger_Vorname UNIQUE (Vorname, Klasse);
```

- Der Schüler *Tim Weber* (SchülerID: 5) wechselt die Klasse. Geben Sie den SQL-Befehl an, der den genannten Schüler in die Klasse „4C“ überführt.

Lösungsvorschlag

```
UPDATE Schüler
SET Klasse = '4C'
WHERE
 Vorname = 'Tim' AND
 Nachname = 'Weber' AND
 SchülerID = 5;
```

- Die Schülerin *Laura Müller* (SchülerID: 1) zieht um und wechselt die Schule. Löschen Sie die Schülerin aus der Datenbank. Nennen Sie einen möglichen Effekt, welcher bei der Verwendung von Primär- und Fremdschlüsseln auftreten kann.

Alle Noten von *Laura Müller* werden gelöscht, falls **ON DELETE CASCADE** gesetzt ist. Oder es müssen erst alle Fremdschlüsselverweise auf diese *SchülerID* in der Tabelle *Noten* gelöscht werden

```
DELETE FROM Noten
WHERE SchülerID = 1;
```

- Erstellen Sie eine View „DurchschnittsNoten“, die die folgenden Spalten beinhaltet: *Klasse*, *Schulfach*, *Durchschnittsnote*

Hinweis: Beachten Sie die Gewichte der Noten.

```
CREATE VIEW DurchschnittsNoten AS (
 (SELECT s.Klasse, n.Schulfach, (SUM(n.Note * n.Gewicht) /
 ↪ SUM(n.Gewicht)) AS Durchschnittsnote
 FROM Noten n, Schüler s
 WHERE s.SchülerID = n.SchülerID
 GROUP BY s.Klasse, n.Schulfach)
);
```

```
SELECT * FROM DurchschnittsNoten;
```

| klasse | schulfach  | durchschnittsnote |
|--------|------------|-------------------|
| 4A     | Mathematik | 2                 |

(1 row)

- Geben Sie den Befehl an, der die komplette Tabelle „Noten“ löscht.

```
DROP TABLE Noten;
```

- (d) Formulieren Sie die folgenden Anfragen in SQL. Beachten Sie dabei, dass sie SQL-Befehle auch noch bei Änderungen der Ausprägung die korrekten Anfrageergebnisse zurückgeben sollen.

- Gesucht ist die durchschnittliche Note, die im Fach Mathematik vergeben wird.

Hinweis: Das Gewicht ist bei dieser Anfrage nicht relevant

```
SELECT AVG(Note)
FROM Noten
WHERE Schulfach = 'Mathematik';
```

- Berechnen Sie die Anzahl der Schüler, die im Fach Mathematik am 23.09.2017 eine Schulaufgabe (öGewicht=2) geschrieben haben.

Lösungsvorschlag

```
SELECT COUNT(*) AS Anzahl_Schüler
FROM Noten
WHERE Datum = '23.09.2017' AND Gewicht = 2 AND Schulfach = 'Mathematik';

anzahl_schüler

1
(1 row)
```

- Geben Sie die *SchülerID* aller Schüler zurück, die im Fach Mathematik mindestens drei mal die Schulnote 6 geschrieben haben.

Lösungsvorschlag

```
SELECT SchülerID
FROM Noten
WHERE Schulfach = 'Mathematik' AND Note = 6
GROUP BY SchülerID
HAVING COUNT(*) >= 3;

schülerid

(0 rows)
```

- Gesucht ist der Notendurchschnitt bezüglich jedes Fachs der Klasse „4A“.

Lösungsvorschlag

```
SELECT n.Schulfach, AVG(n.Note)
FROM Schüler s, Noten n
WHERE s.SchülerID = n.SchülerID AND s.Klasse = '4A'
GROUP BY n.Schulfach;

schulfach | avg
-----+-----
Mathematik | 2.5000000000000000
(1 row)
```

- (e) Geben Sie jeweils an, welchen Ergebniswert die folgenden SQL-Befehle für die gegebene Ausprägung zurückliefern.

```
SELECT COUNT(DISTINCT Klasse)
FROM
Schüler NATURAL JOIN Noten;
```

Lösungsvorschlag

```
count

1
(1 row)
```

4A von Laura Müller. Ohne **DISTINCT** wäre das Ergebnis 4.

```
SELECT COUNT(ALL Klasse)
FROM
Noten, Schüler;
```

```
count

 32
(1 row)
```

Es entsteht das Kreuzprodukt ( $8 \cdot 4 = 32$ ).

```
SELECT COUNT(Note)
FROM
Schüler NATURAL LEFT OUTER JOIN Noten;
```

```
SELECT * FROM Schüler NATURAL LEFT OUTER JOIN Noten;
```

ergibt:

| schülerid | vorname | nachname  | klasse | schulfach  | note | gewicht | datum      |
|-----------|---------|-----------|--------|------------|------|---------|------------|
| 1         | Laura   | Müller    | 4A     | Mathematik | 3    | 2       | 2017-09-23 |
| 1         | Laura   | Müller    | 4A     | Mathematik | 1    | 1       | 2017-10-03 |
| 1         | Laura   | Müller    | 4A     | Mathematik | 2    | 2       | 2017-10-15 |
| 1         | Laura   | Müller    | 4A     | Mathematik | 4    | 1       | 2017-11-11 |
| 2         | Linus   | Schmidt   | 4A     |            |      |         |            |
| 5         | Tim     | Weber     | 4B     |            |      |         |            |
| 8         | Julia   | Wolf      | 4C     |            |      |         |            |
| 6         | Lea     | Becker    | 4B     |            |      |         |            |
| 4         | Liam    | Fischer   | 4B     |            |      |         |            |
| 3         | Jonas   | Schneider | 4A     |            |      |         |            |
| 7         | Emilia  | Klein     | 4C     |            |      |         |            |

(11 rows)

```
count

 4
(1 row)
```

COUNT zählt die **NULL**-Werte nicht mit. Die *Laura Müller* hat 4 Noten.

```
SELECT COUNT(*)
FROM
Schüler NATURAL LEFT OUTER JOIN Noten;
```

```
count

 11
(1 row)
```

Siehe Zwischenergebnistabelle in der obenstehenden Antwort. Alle Schüler und die Laura 4-mal, weil sie 4 Noten hat.

## 46116 / 2018 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 5

Überführen Sie das Datenbankschema in ein ER-Diagramm. Verwenden Sie hierfür die bereits eingezeichneten Entity-Typen und Relationship-Typen. Weisen Sie die Relatio-



nen zu und schreiben Sie deren Namen in die dazugehörigen Felder. Fügen Sie, falls erforderlich, Attribute hinzu und beschriften Sie die Beziehungen. Markieren Sie Schlüsselattribute durch unterstreichen.

Gegeben sei das folgende Datenbankschema, wobei Primärschlüssel unterstrichen und Fremdschlüssel überstrichen sind. Die von einem Fremdschlüssel referenzierte Relation ist in eckigen Klammern nach dem Fremdschlüsselattribut angegeben.

Schüler (SchülerID, SVorname, SNachname, KlassenID[Klassen], Geburtsdatum)

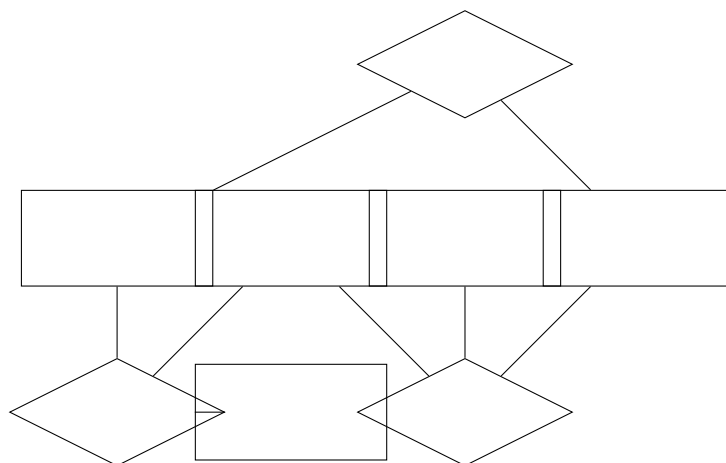
Lehrer (LehrerID, LVorname, LNachname)

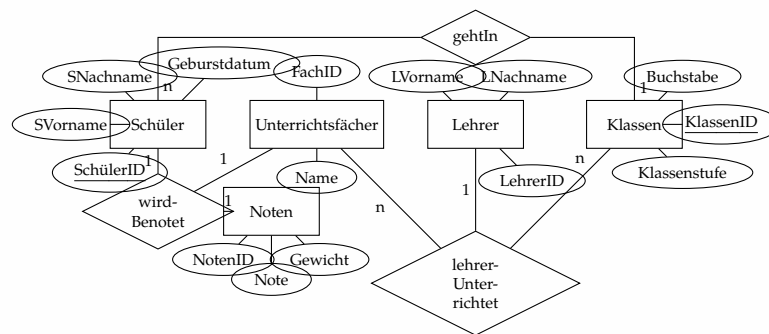
Klassen (KlassenID, Klassenstufe, Buchstabe)

Unterrichtsfächer (FachID, Name)

Noten (NotenID, SchülerID[Schüler], FachID[Unterrichtsfächer], Note, Gewicht)

LehrerUnterrichtet (LehrerID[Lehrer], KlassenID[Klassen], FachID[Unterrichtsfächer])





## Funktionalitäten

**gehtIn** n:1 Eine Klasse hat n Schüler, einE SchülerIn geht in eine Klasse

**wird Benotet** 1:1:1: Das ist anderes nicht möglich, da nur NotenID Primärschlüssel ist, dann muss aber die Kombination aus Schüler und Unterrichtsfach auch einmalig sein,  $\delta$ UNIQUE und es gilt Note und Unterrichtsfach bestimmt Schüler, Note und Schüler bestimmt Unterrichtsfach und Schüler und Unterrichtsfach bestimmt Noten.

**LehrerUnterrichtet** 1(Lehrer):n:n

## 66112 (Automatentheorie / Komplexität / Algorithmen (vertieft))

### 66112 / 2003 / Herbst / Thema 2 / Aufgabe 5

Zeigen Sie mit Hilfe vollständiger Induktion, dass das folgende Programm bzgl. der Vorbedingung  $x > 0$  und der Nachbedingung  $\text{drei\_hoch } x = 3^x$  partiell korrekt ist!

```
(define (drei_hoch x)
 (cond ((= x 0) 1)
 (else (* 3 (drei_hoch (- x 1)))))
)
```

Lösungsvorschlag

#### Induktionsanfang

— Beweise, dass  $A(1)$  eine wahre Aussage ist. \_\_\_\_\_  
 $\text{drei\_hoch } 1 = 3 \cdot (\text{drei\_hoch } 0) = 3 \cdot 1 = 3$

#### Induktionsvoraussetzung

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . \_\_\_\_\_  
 für alle  $x < x_0$  gilt  $\text{drei\_hoch } x = 3^x$

#### Induktionsschritt

— Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss. \_\_\_\_\_  
 $x \rightarrow x+1$

$$\begin{aligned} \text{drei\_hoch } (x + 1) &= 3 \cdot \text{drei\_hoch } (-(x + 1)1)) \\ &= 3 \cdot (\text{drei\_hoch } x) \\ &= 3 \cdot 3^x \\ &= 3^{x+1} \end{aligned}$$

## 66112 / 2003 / Herbst / Thema 2 / Aufgabe 8

- (a) Implementieren Sie in einer objektorientierten Sprache einen binären Suchbaum für ganze Zahlen! Dazu gehören Methoden zum Setzen und Ausgeben der Attribute `zahl`, `linker_teilbaum` und `rechter_teilbaum`. Design: eine Klasse `Knoten` und eine Klasse `BinBaum`. Ein Knoten hat einen linken und einen rechten Nachfolger. Ein Baum verwaltet die Wurzel. Er hängt neue Knoten an und löscht Knoten.

```
public class BinBaum {

 private Knoten wurzel = null;

 public void setzeWurzel(Knoten knoten) {
 wurzel = knoten;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2003/herbst/BinBaum.java](https://github.com/orgs/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java)

```
public class Knoten {
 private int zahl;
 private Knoten links = null;
 private Knoten rechts = null;

 public Knoten() {
 }

 public Knoten(int zahl) {
 this.zahl = zahl;
 }

 public void setzeZahl(int zahl) {
 this.zahl = zahl;
 }

 public int gibZahl() {
 return zahl;
 }

 public void setzeLinks(Knoten k) {
 links = k;
 }

 public Knoten gibLinks() {
 return links;
 }

 public void setzeRechts(Knoten k) {
 rechts = k;
 }

 public Knoten gibRechts() {
 return rechts;
 }
}
```

```

 }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2003/herbst/Knoten.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2003/herbst/Knoten.java)

(b) Schreiben Sie eine Methode `fügeEin(...)`, die eine Zahl in den Baum einfügt!

Lösungsvorschlag

```

public void fügeEin(int zahl) {
 Knoten aktueller = wurzel;
 Knoten neuerKnoten = new Knoten(zahl);
 if (wurzel == null) {
 wurzel = neuerKnoten;
 return;
 }
 while (aktueller != null) {
 // suche links
 if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() != null) {
 aktueller = aktueller.gibLinks();
 // fuege ein
 } else if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() == null)
 ↪ {
 aktueller.setzeLinks(neuerKnoten);
 break;
 }
 // suche rechts
 if (zahl > aktueller.gibZahl() && aktueller.gibRechts() != null) {
 aktueller = aktueller.gibRechts();
 // fuege ein
 } else if (zahl > aktueller.gibZahl() && aktueller.gibRechts() == null)
 ↪ {
 aktueller.setzeRechts(neuerKnoten);
 break;
 }
 }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2003/herbst/BinBaum.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java)

(c) Schreiben Sie eine Methode `void besuchePostOrder(...)`, die die Zahlen in der Reihenfolge postorder ausgibt!

Lösungsvorschlag

```

public static void besuchePostOrder(Knoten knoten) {
 // Sonderfall leerer (Teil-)Baum
 if (knoten == null) {
 System.out.println("Leerer Baum");
 } else {
 // Linker
 if (knoten.gibLinks() != null) {
 besuchePostOrder(knoten.gibLinks());
 }
 // Rechter
 if (knoten.gibRechts() != null) {

```

```
 besuchePostOrder(knoten.gibRechts());
 }
 System.out.println(knoten.gibZahl());
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2003/herbst/BinBaum.java](https://github.com/orgs/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java)

- (d) Ergänzen Sie Ihr Programm um die rekursiv implementierte Methode `int berechneSumme(...)`, die die Summe der Zahlen des Unterbaums, dessen Wurzel der Knoten ist, zurückgibt! Falls der Unterbaum leer ist, ist der Rückgabewert 0!

```
int summe (Knoten x)...
```

Lösungsvorschlag

```
public int berechneSumme(Knoten knoten) {
 int ergebnis = 0;

 // Sonderfall: leerer Unterbaum
 if (knoten == null) {
 return 0;
 }
 // linker
 if (knoten.gibLinks() != null) {
 ergebnis = ergebnis + berechneSumme(knoten.gibLinks());
 }
 // rechter
 if (knoten.gibRechts() != null) {
 ergebnis = ergebnis + berechneSumme(knoten.gibRechts());
 }
 // Wurzel
 ergebnis = ergebnis + knoten.gibZahl();
 return ergebnis;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2003/herbst/BinBaum.java](https://github.com/orgs/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java)

- (e) Schreiben Sie eine Folge von Anweisungen, die einen Baum mit Namen BinBaum erzeugt und nacheinander die Zahlen 5 und 7 einfügt! In den binären Suchbaum werden noch die Zahlen 4, 11, 6 und 2 eingefügt. Zeichnen Sie den Baum, den Sie danach erhalten haben, und schreiben Sie die eingefügten Zahlen in der Reihenfolge der Traversierungsmöglichkeit postorder auf!
- (f) Implementieren Sie eine Operation `isSorted(...)`, die für einen (Teil-)baum feststellt, ob er sortiert ist.

```
public boolean istSortiert(Knoten knoten) {
 // Baum leer
 if (knoten == null) {
 return true;
 }
}
```



```
// linker Nachfolger nicht okay
if (knoten.gibLinks() != null && knoten.gibLinks().gibZahl() >
 ↪ knoten.gibZahl()) {
 return false;
}

// rechter Nachfolger nicht okay
if (knoten.gibRechts() != null && knoten.gibRechts().gibZahl() <=
 ↪ knoten.gibZahl()) {
 return false;
}
// sonst prüfe Teilbaeume
return (istSortiert(knoten.gibRechts()) &&
 ↪ istSortiert(knoten.gibLinks()));
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2003/herbst/BinBaum.java](https://github.com/orgs/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java)

```
public class BinBaum {

 private Knoten wurzel = null;

 public void setzeWurzel(Knoten knoten) {
 wurzel = knoten;
 }

 public void fügeEin(int zahl) {
 Knoten aktueller = wurzel;
 Knoten neuerKnoten = new Knoten(zahl);
 if (wurzel == null) {
 wurzel = neuerKnoten;
 return;
 }
 while (aktueller != null) {
 // suche links
 if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() != null) {
 aktueller = aktueller.gibLinks();
 // fuege ein
 } else if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() == null) {
 aktueller.setzeLinks(neuerKnoten);
 break;
 }
 // suche rechts
 if (zahl > aktueller.gibZahl() && aktueller.gibRechts() != null) {
 aktueller = aktueller.gibRechts();
 // fuege ein
 } else if (zahl > aktueller.gibZahl() && aktueller.gibRechts() == null) {
 aktueller.setzeRechts(neuerKnoten);
 break;
 }
 }
 }
}
```

```
public static void besuchePostOrder(Knoten knoten) {
 // Sonderfall leerer (Teil-)Baum
 if (knoten == null) {
 System.out.println("Leerer Baum");
 } else {
 // Linker
 if (knoten.gibLinks() != null) {
 besuchePostOrder(knoten.gibLinks());
 }
 // Rechter
 if (knoten.gibRechts() != null) {
 besuchePostOrder(knoten.gibRechts());
 }
 System.out.println(knoten.gibZahl());
 }
}

public int berechneSumme(Knoten knoten) {
 int ergebnis = 0;

 // Sonderfall: leerer Unterbaum
 if (knoten == null) {
 return 0;
 }
 // linker
 if (knoten.gibLinks() != null) {
 ergebnis = ergebnis + berechneSumme(knoten.gibLinks());
 }
 // rechter
 if (knoten.gibRechts() != null) {
 ergebnis = ergebnis + berechneSumme(knoten.gibRechts());
 }
 // Wurzel
 ergebnis = ergebnis + knoten.gibZahl();
 return ergebnis;
}

public boolean istSortiert(Knoten knoten) {
 // Baum leer
 if (knoten == null) {
 return true;
 }

 // linker Nachfolger nicht okay
 if (knoten.gibLinks() != null && knoten.gibLinks().gibZahl() >
 ↪ knoten.gibZahl()) {
 return false;
 }

 // rechter Nachfolger nicht okay
 if (knoten.gibRechts() != null && knoten.gibRechts().gibZahl() <=
 ↪ knoten.gibZahl()) {
 return false;
 }
}
```

```
// sonst prüfe Teilbaeume
return (istSortiert(knoten.gibRechts()) && istSortiert(knoten.gibLinks()));
}

public static void main(String[] args) {
 BinBaum baum = new BinBaum();

 baum.fügeEin(5);
 baum.fügeEin(7);
 baum.fügeEin(4);
 baum.fügeEin(11);
 baum.fügeEin(6);
 baum.fügeEin(2);

 besuchePostOrder(baum.wurzel);
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2003/herbst/BinBaum.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java)

```
public class Knoten {
 private int zahl;
 private Knoten links = null;
 private Knoten rechts = null;

 public Knoten() {
 }

 public Knoten(int zahl) {
 this.zahl = zahl;
 }

 public void setzeZahl(int zahl) {
 this.zahl = zahl;
 }

 public int gibZahl() {
 return zahl;
 }

 public void setzeLinks(Knoten k) {
 links = k;
 }

 public Knoten gibLinks() {
 return links;
 }

 public void setzeRechts(Knoten k) {
 rechts = k;
 }

 public Knoten gibRechts() {
 return rechts;
 }
}
```

```

 }
}

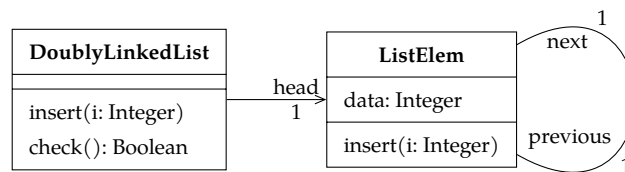
```

Klassendiagramm

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2003/herbst/Knoten.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2003/herbst/Knoten.java)

## 66112 / 2005 / Frühjahr / Thema 1 / Aufgabe 1

Betrachten Sie folgendes Klassendiagramm, das doppelt-verkettete Listen spezifiziert. Die Assoziation `head` zeigt auf das erste Element der Liste. Die Assoziationen `previous` und `next` zeigen auf das vorherige bzw. folgende Element.



Implementieren Sie die doppelt-verketteten Listen in einer geeigneten objektorientierten Sprache (z. B. Java oder C++), das heißt:

- (a) Implementieren Sie die Klasse `ListElem`. Die Methode `insert` ordnet eine ganze Zahl `i` in eine aufsteigend geordnete doppelt-verkettete Liste `l` an die korrekte Stelle ein. Sei z. B. das Objekt `l` eine Repräsentation der Liste `[0, 2, 2, 6, 8]` dann liefert `l.insert(3)` eine Repräsentation der Liste `[0, 2, 2, 3, 6, 8]`.

Lösungsvorschlag

```

public class ListElem {
 private int data;
 private ListElem previous;
 private ListElem next;

 public ListElem(int i) {
 data = i;
 }

 public ListElem() {
 }

 public void insert(int i) {
 ListElem newElement = new ListElem(i);
 if (i <= data) {
 if (previous != null) {
 newElement.next = this;
 newElement.previous = previous;
 previous.next = newElement;
 previous = newElement;
 } else {
 newElement.next = this;
 previous = newElement;
 }
 }
 }
}

```

```

 }
 } else {
 if (next != null) {
 next.insert(i);
 } else {
 newElement.previous = this;
 next = newElement;
 }
 }
}

public ListElem getPrevious() {
 return previous;
}

public ListElem getNext() {
 return next;
}

public int getData() {
 return data;
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2005/fruehjahr/ListElem.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/ListElem.java)

- (b) Implementieren Sie die Klasse `DoublyLinkedList`, wobei die Methode `insert` eine Zahl `i` in eine aufsteigend geordnete Liste einordnet. Die Methode `check` überprüft, ob eine Liste korrekt verkettet ist, d.h. für jedes `ListElem`-Objekt `o`, das über den `head` der Liste erreichbar ist, der Vorgänger des Nachfolgers von `o` gleich `o` ist.

Lösungsvorschlag

```

public class DoublyLinkedList {
 private ListElem head;

 public DoublyLinkedList() {
 }

 public void insert(int i) {
 if (head != null) {
 // Immer einen neue Zahl einfügen, nicht nur wenn die Zahl kleiner ist als
 // → head.
 head.insert(i);
 // Es muss kleiner gleich heißen, sonst können mehrer gleiche Zahlen am
 // → Anfang
 // nicht eingefügt werden.
 if (i <= head.getData()) {
 head = head.getPrevious();
 }
 } else {
 head = new ListElem(i);
 }
 }
}

```

```

}

public boolean check() {
 ListElem current = head;
 while (current.getNext() != null) {
 if (current.getNext().getPrevious() != current) {
 return false;
 } else {
 current = current.getNext();
 }
 }
 return true;
}

public ListElem getHead() {
 return head;
}

public static void main(String[] args) {
 DoublyLinkedList list = new DoublyLinkedList();
 // int[] numbers = new int[] { 1 };
 // int[] numbers = new int[] { 1, 1, 1, 1, };
 // int[] numbers = new int[] { 1, 1, 1, 2, };
 // int[] numbers = new int[] { 2, 1, 1, 1, };
 // int[] numbers = new int[] { 2, 1 };
 int[] numbers = new int[] { 0, 2, 2, 6, 8, 4 };
 for (int number : numbers) {
 list.insert(number);
 }
 list.insert(3);

 ListElem current = list.getHead();
 while (current.getNext() != null) {
 System.out.println(current.getData());
 current = current.getNext();
 }
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2005/fruehjahr/DoublyLinkedList.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/DoublyLinkedList.java)

## 66112 / 2005 / Frühjahr / Thema 2 / Aufgabe 8

Gegeben seien die folgenden Zahlen: 7, 4, 3, 5, 0, 1

- (a) Zeichnen Sie eine Hash-Tabelle mit 8 Zellen und tragen Sie diese Zahlen genau in der oben gegebenen Reihenfolge in Ihre Hash-Tabelle ein. Verwenden Sie dabei die Streufunktion  $f(n) = n^2 \bmod 7$  und eine Kollisionsauflösung durch lineares Sondieren.

Lösungsvorschlag

$$f(7) = 7^2 \bmod 7 = 49 \bmod 7 = 0$$

$$f(4) = 4^2 \bmod 7 = 16 \bmod 7 = 2$$

$f(3) = 3^2 \bmod 7 = 9 \bmod 7 = 2$  lineares Sondieren:  $+1 = 3$   
 $f(5) = 5^2 \bmod 7 = 25 \bmod 7 = 4$   
 $f(0) = 0^2 \bmod 7 = 0 \bmod 7 = 0$  lineares Sondieren:  $+1 = 1$   
 $f(1) = 1^2 \bmod 7 = 1 \bmod 7 = 1$  lineares Sondieren:  $-1 = 0, -1 = 7$

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 7 | 0 | 4 | 3 | 5 |   |   | 1 |

- (b) Welcher Belegungsfaktor ist für die Streutabelle und die Streufunktion aus Teilaufgabe a zu erwarten, wenn sehr viele Zahlen eingeordnet werden und eine Kollisionsauflösung durch Verkettung (verzeigerte Listen) verwendet wird? Begründen Sie Ihre Antwort kurz.

Lösungsvorschlag

Der Belegungsfaktor berechnet sich aus der Formel:

$$\text{Belegungsfaktor} = \frac{\text{Anzahl tatsächlich eingetragenen Schlüssel}}{\text{Anzahl Hashwerte}}$$

Der Belegungsfaktor steigt kontinuierlich, je mehr Zahlen in die Streutabelle gespeichert werden.

Die Streufunktion legt die Zahlen nur in die Buckets 0, 1, 2, 4.

## 66113 (Rechnerarchitektur / Datenbanken / Betriebssysteme (vertieft))

### 66113 / 2003 / Frühjahr / Thema 1 / Aufgabe 5

Gegeben seien die folgenden drei Relationen. Diese Relationen erfassen die Mitarbeiterverwaltung eines Unternehmens. Schlüssel sind fett dargestellt und Fremdschlüssel sind kursiv dargestellt. So werden Mitarbeiter, Abteilungen und Unternehmen jeweils durch ihre Nummer identifiziert. AbtNr ist die Nummer der Abteilung, in der ein Mitarbeiter arbeitet. Manager ist die Nummer des Mitarbeiters, der die Abteilung leitet. UntNr ist die Nummer des Unternehmens, dem eine Abteilung zugeordnet ist.

Mitarbeiter(Nummer, Name, Alter, Gehalt, AbtNr)  
 Abteilung(Nummer, Name, Budget, Manager, UntNr)  
 Unternehmen(Nummer, Name, Adresse)

```

CREATE TABLE unternehmen (
 Nummer integer NOT NULL PRIMARY KEY,
 Name VARCHAR(20) DEFAULT NULL,
 Adresse VARCHAR(50) DEFAULT NULL
);

```

```

CREATE TABLE abteilung (

```



```
 Nummer integer NOT NULL PRIMARY KEY,
 Name VARCHAR(20) DEFAULT NULL,
 Budget float DEFAULT NULL,
 Manager VARCHAR(20) NOT NULL,
 UntNr integer DEFAULT NULL REFERENCES unternehmen (Nummer)
);

CREATE TABLE mitarbeiter (
 Nummer integer NOT NULL PRIMARY KEY,
 Name VARCHAR(20) NOT NULL,
 Alter integer NOT NULL,
 Gehalt float NOT NULL,
 AbtNr integer NOT NULL REFERENCES abteilung (Nummer)
);

INSERT INTO unternehmen (Nummer, Name, Adresse) VALUES
(1, 'Test.com', 'Alter Hafen 11'),
(2, 'Party.de', 'Technostraße 3'),
(3, 'IT.ch', 'Sequelweg 1');

INSERT INTO abteilung (Nummer, Name, Budget, Manager, UntNr) VALUES
(1, 'Personal_Care', 20000, 'Huber', 1),
(11, 'Tequilla_Mix', 50000, 'Taylor', 2),
(21, 'Nerds', 500, 'Gates', 3);

INSERT INTO mitarbeiter (Nummer, Name, Alter, Gehalt, AbtNr) VALUES
(1, 'Müller', 30, 30000, 1),
(2, 'Huber', 45, 80000, 1),
(3, 'Habermeier', 62, 40000, 1),
(4, 'Leifsson', 27, 50000, 1),
(5, 'Taylor', 37, 85000, 11),
(6, 'Smith', 61, 34000, 11),
(7, 'Pitt', 36, 40000, 11),
(8, 'Thompson', 54, 52000, 11),
(9, 'Gates', 69, 15000000, 21),
(10, 'Zuckerberg', 36, 10000000, 21),
(11, 'Jobs', 99, 14000000, 21),
(12, 'Nakamoto', 66, 5000000, 21);
```

- (a) Wie hoch ist das Durchschnittsalter der Abteilung „Personal Care“ im Unternehmen „Test.com“?

Lösungsvorschlag

GROUP BY nicht nötig, AS nicht vergessen.

```
SELECT AVG(m.Alter) AS Durchschnittsalter
FROM Unternehmen u, Abteilung a, Mitarbeiter m
WHERE
 a.Name = 'Personal Care' AND
 u.Name = 'Test.com' AND
 u.Nummer = a.UntNr AND
 m.AbtNr = a.Nummer;
```

- (b) Geben Sie für jedes Unternehmen das Durchschnittsalter der Mitarbeiter an!

Statt a.UntNr kann u.Nummer verwendet werden. a.UntNr nur deshalb, weil man dann eventuell den Join über die Unternehmenstabelle sparen kann.  
 Alles was ausgegeben werden soll, muss auch in GROUP BY enthalten sein.

```
SELECT a.UntNr, u.Name, AVG(m.Alter) as Durchschnittsalter
FROM Unternehmen u, Abteilung a, Mitarbeiter m
WHERE
 u.Nummer = a.UntNr AND
 m.AbtNr = a.Nummer
GROUP BY a.UntNr, u.Name;
```

- (c) Wie viele Mitarbeiter im Unternehmen „Test.com“ sind älter als ihr Chef? (D.h. sind älter als der Manager der Abteilung, in der sie arbeiten.)

```
SELECT COUNT(*)
FROM Mitarbeiter m, Abteilung a, Unternehmen u
WHERE
 m.AbtNr = a.Nummer AND
 a.UntNr = u.Nummer AND
 u.Name = 'Test.com'
AND m.Alter > (
 SELECT ma.Alter
 FROM Mitarbeiter ma, Abteilung ab
 WHERE
 ma.Nummer = ab.Manager AND
 a.Nummer = ab.Nummer
);
```

oder einfacher:

```
SELECT COUNT(*)
FROM Mitarbeiter m, Abteilung a, Unternehmen u
WHERE
 m.AbtNr = a.Nummer AND
 a.UntNr = u.Nummer AND
 u.Name = 'Test.com'
AND m.Alter > (
 SELECT ma.Alter
 FROM Mitarbeiter ma
 WHERE ma.Nummer = a.Manager
);
```

Alternativ Lösung ohne Unterabfragen, mit Self join:

```
SELECT COUNT(*)
FROM Mitarbeiter m, Abteilung a, Unternehmen u, Mitarbeiter m2
WHERE
 m.AbtNr = a.Nummer AND
 a.UntNr = u.Nummer AND
 u.Name = 'Test.com' AND
```

```
a.Manager = m2.Nummer AND
m.Alter > m2.Alter;
```

- (d) Welche Abteilungen haben ein geringeres Budget als die Summe der Gehälter der Mitarbeiter, die in der Abteilung arbeiten?

Lösungsvorschlag

```
SELECT a.Name, a.Nummer
FROM Abteilung a
WHERE a.Budget < (
 SELECT SUM(m.Gehalt)
 FROM Mitarbeiter m
 WHERE a.Nummer = m.AbtNr
);
```

Ohne Unterabfrage

```
SELECT a.Name, a.Nummer
FROM Abteilung a, Mitarbeiter m
WHERE a.Nummer = m.AbtNr
GROUP BY a.Nummer, a.Name, a.Budget
HAVING a.Budget < SUM(m.Gehalt);
```

- (e) Versetzen Sie den Mitarbeiter „Wagner“ in die Abteilung „Personal Care“!

Lösungsvorschlag

```
UPDATE Mitarbeiter m
SET AbtNr = (
 SELECT a.Nummer FROM
 Abteilung a
 WHERE a.Name = 'Personal Care'
)
WHERE m.Name = 'Wagner';
```

- (f) Löschen Sie die Abteilung „Personal Care“ mit allen ihren Mitarbeitern!

Lösungsvorschlag

```
DELETE FROM Mitarbeiter
WHERE AbtNr = (
 SELECT a.Nummer
 FROM Abteilung a
 WHERE a.Name = 'Personal Care'
);

DELETE FROM Abteilung
WHERE Name = 'Personal Care';
```

- (g) Geben Sie den Managern aller Abteilungen, die ihr Budget nicht überziehen, eine 10 Prozent Gehaltserhöhung. (Das Budget ist überzogen, wenn die Gehälter der Mitarbeiter höher sind als das Budget der Abteilung.) Zusatzfrage: Was passiert mit Mitarbeitern, die Manager von mehreren Abteilungen sind?

```

CREATE VIEW LowBudget AS (
 SELECT Nummer
 FROM Abteilung
 WHERE Nummer NOT IN (
 SELECT a.Nummer
 FROM Abteilung a
 WHERE
 a.Budget < (
 SELECT SUM(Gehalt)
 FROM Mitarbeiter m Abteilung A
 WHERE m.AbtNr = A.Nummer AND
 a.Nummer = A.Nummer
)
)
)

UPDATE Mitarbeiter
SET Gehalt = 1.1 * Gehalt
WHERE Nummer IN (
 SELECT Manager
 FROM LowBudget, Abteilung
 WHERE LowBudget.Manager = Abteilung.Nummer
)

```

### 66113 / 2003 / Frühjahr / Thema 2 / Aufgabe 3

Gegeben sei folgendes relationales Schema, das eine Universitätsverwaltung modelliert:

```

Studenten {[MatrNr:integer, Name:string, Semester:integer]}
Vorlesungen {[VorlNr:integer, Titel:string, SWS:integer, gelesenVon:integer]}
Professoren {[PersNr:integer, Name:string, Rang:string, Raum:integer]}
 hoeren {[MatrNr:integer, VorlNr:integer]}
 voraussetzen {[VorgaengerVorlNr:integer, NachfolgerVorlNr:integer]}
 pruefen {[MatrNr:integer, VorlNr:integer, PrueferPersNr:integer, Note:decimal]}

```

Formulieren Sie die folgenden Anfragen in SQL:

- (a) Alle Studenten, die den Professor *Kant* aus einer Vorlesung kennen.

Lösungsvorschlag

```

SELECT DISTINCT s.Name
FROM Studenten s, Professoren p, hoeren h, Vorlesungen v
WHERE
 p.Name = 'Kant' AND
 v.gelesenVon = p.PersNr AND
 s.MatrNr = h.MatrNr AND
 h.VorlNr = v.VorlNr

```

- (b) Geben Sie eine Liste der Professoren (Name, PersNr) mit ihrem Lehrdeputat (Summe der SWS der gelesenen Vorlesungen) aus. Ordnen Sie diese Liste so, dass sie absteigend nach Lehrdeputat sortiert ist! Bei gleicher Lehrtätigkeit dann noch aufsteigend nach dem Namen des Professors/der Professorin.

```
SELECT p.Name, p.PersNr, SUM(v.SWS) AS Lehrdeputat
FROM Vorlesung v, Professoren p
WHERE v.gelesenVon = p.PersNr
GROUP BY p.Name, p.PersNr
ORDER BY Lehrdeputat DESC, p.Name ASC;
```

- (c) Geben Sie eine Liste der Studenten (Name, MatrNr, Semester) aus, die mindestens zwei Vorlesungen bei *Kant* gehört haben.

### Mit einer VIEW

```
CREATE VIEW hoertKant AS
SELECT s.Name, s.MatrNr, s.Semester, v.VorlNr
FROM Studenten s, hoeren h, Vorlesungen v, Professoren p
WHERE
 s.MatrNr = h.MatrNr AND
 h.VorlNr = v.VorlNr AND
 v.gelesenVon = p.PersNr AND
 p.Name = 'Kant';

SELECT DISTINCT h1.Name, h2.MatrNr, h1.Semester
FROM hoertKant h1, hoertKant h2
WHERE h1.MatrNr = h2.MatrNr AND h1.VorlNr <> h2.VorlNr;
```

oder:

```
SELECT DISTINCT Name, MatrNr, Semester
FROM hoertKant
GROUP BY Name, MatrNr, Semester
HAVING COUNT(VorlNr) > 1;
```

### In einer Abfrage

```
SELECT s.Name, s.MatrNr, s.Semester
FROM Studenten s, hoeren h, Vorlesungen v, Professoren p
WHERE
 s.MatrNr = h.MatrNr AND
 h.VorlNr = v.VorlNr AND
 v.gelesenVon = p.PersNr AND
 p.Name = 'Kant'
GROUP BY s.MatrNr, s.Name, s.Semester
HAVING COUNT(s.MatrNr) > 1;
```

- (d) Geben Sie eine Liste der Semesterbesten (MatrNr und Notendurchschnitt) aus.

```
CREATE VIEW Notenschnitte AS (
 SELECT p.MatrNr, s.Name, s.Semester, AVG(Note) AS Durchschnitt
 FROM Studenten s, pruefen p
 WHERE s.MatrNr = p.MatrNr
 GROUP BY p.MatrNr, s.Name, s.Semester
);

SELECT a.Durchschnitt, a.MatrNr, a.Semester
FROM Notenschnitte a, Notenschnitte b
WHERE
 a.Durchschnitt >= b.Durchschnitt
 a.Semster = b.Semster
GROUP BY a.Durchschnitt, a.MatrNr, a.Semester
HAVING COUNT(*) < 2;
```

## 66115 (Theoretische Informatik / Algorithmen (vertieft))

### 66115 / 2007 / Frühjahr / Thema 1 / Aufgabe 7

Implementieren Sie die angegebenen Methoden einer Klasse `Queue` für Warteschlangen. Eine Warteschlange soll eine unbeschränkte Anzahl von Elementen aufnehmen können. Elemente sollen am Ende der Warteschlange angefügt und am Anfang aus ihr entfernt werden. Sie können davon ausgehen, dass ein Klasse `QueueElement` mit der folgenden Schnittstelle bereits implementiert ist .

```
class QueueElement {

 private QueueElement next;
 private Object contents;

 QueueElement(Object contents) {
 this.contents = contents;
 }

 Object getContents() {
 return contents;
 }

 QueueElement getNext() {
 return next;
 }

 void setNext(QueueElement next) {
 this.next = next;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2007/fruehjahr/queue/QueueElement.java](src/main/java/org/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/QueueElement.java)

Von der Klasse `Queue` ist folgendes gegeben:

```
class Queue {
 QueueElement first;
 QueueElement last;
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2007/fruehjahr/queue/Queue.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/Queue.java)

- (a) Schreiben Sie eine Methode `void append (Object contents)`, die ein neues Objekt in der Warteschlange einfügt.

Lösungsvorschlag

```
public void append(Object contents) {
 QueueElement newElement = new QueueElement(contents);
 if (first == null) {
 first = newElement;
 last = newElement;
 } else {
 // neues Element hinten anhängen
 last.setNext(newElement);
 // angehängtes Element ist Letztes
 last = last.getNext();
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2007/fruehjahr/queue/Queue.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/Queue.java)

- (b) Schreiben Sie eine Methode `Object remove()`, die ein Element aus der Warteschlange entfernt und dessen Inhalt zurückliefert. Berücksichtigen Sie, dass die Warteschlange leer sein könnte.

Lösungsvorschlag

```
public Object remove() {
 Object tmp = null;
 if (first != null) {
 // Den Inhalt des ersten Elements temporär speichern
 tmp = first.getContents();
 // Das erste Element aus der Schlange nehmen
 first = first.getNext();
 }
 // Den Inhalt des gelöschten Elements ausgeben bzw . null
 return tmp;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2007/fruehjahr/queue/Queue.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/Queue.java)

- (c) Schreiben Sie eine Methode `boolean isEmpty()`, die überprüft, ob die Warteschlange leer ist.

Lösungsvorschlag

```
public boolean isEmpty() {
 return (first == null);
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2007/fruehjahr/queue/Queue.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/Queue.java)

## Klasse Queue

```
class Queue {
 QueueElement first;
 QueueElement last;

 public void append(Object contents) {
 QueueElement newElement = new QueueElement(contents);
 if (first == null) {
 first = newElement;
 last = newElement;
 } else {
 // neues Element hinten anhängen
 last.setNext(newElement);
 // angehängtes Element ist Letztes
 last = last.getNext();
 }
 }

 public Object remove() {
 Object tmp = null;
 if (first != null) {
 // Dein Inhalt des ersten Elements temporär speichern
 tmp = first.getContents();
 // Das erste Element aus der Schlange nehmen
 first = first.getNext();
 }
 // Den Inhalt des gelöschten Elements ausgeben bzw . null
 return tmp;
 }

 public boolean isEmpty() {
 return (first == null);
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2007/fruehjahr/queue/Queue.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/Queue.java)

## Tests

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class QueueTest {

 @Test
 public void methodAppend() {
 Queue queue = new Queue();
 assertEquals(true, queue.isEmpty());
 queue.append(1);
 assertEquals(false, queue.isEmpty());
 }
}
```



```

@Test
public void methodRemove() {
 Queue queue = new Queue();
 queue.append(1);
 queue.append(2);
 queue.append(3);

 assertEquals(1, queue.remove());
 assertEquals(2, queue.remove());
 assertEquals(3, queue.remove());
 assertEquals(null, queue.remove());
}

@Test
public void methodIsEmpty() {
 Queue queue = new Queue();
 assertEquals(true, queue.isEmpty());
}
}

```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2007/fruehjahr/queue/QueueTest.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2007/fruehjahr/queue/QueueTest.java)

## 66115 / 2007 / Frühjahr / Thema 2 / Aufgabe 1

- Beschreiben Sie in Pseudocode oder einer Programmiersprache Ihrer Wahl einen Greedy-Algorithmus, der einen Betrag von  $n$  Cents mit möglichst wenigen Cent-Münzen herausgibt. Bei  $n = 29$  wäre die erwartete Antwort etwa  $1 \times 20\text{ct}$ ,  $1 \times 5\text{ct}$ ,  $2 \times 2\text{ct}$ .
- Beweisen Sie die Korrektheit Ihres Verfahrens, also dass tatsächlich die Anzahl der Münzen minimiert wird.
- Nehmen wir an, Bayern führe eine Sondermünze im Wert von 7ct ein. Dann liefert der naheliegende Greedy-Algorithmus nicht immer die minimale Zahl von Münzen. Geben Sie für dieses Phänomen ein konkretes Beispiel an und führen Sie aus, warum Ihr Beweis aus Aufgabenteil a) in dieser Situation nicht funktioniert.

## 66115 / 2010 / Herbst / Thema 2 / Aufgabe 3

Gegeben sei ein Array der Größe 10, z. B. `int[] hashfeld = new int [10]`. Die Hashfunktion sei der Wert modulo 10,  $h(x) = x \% 10$ . Kollisionen werden mit linearer Verschiebung um 1 (modulo 10) gelöst.

`in(x)` bedeutet, dass die Zahl  $x$  eingefügt wird, `search(x)`, dass nach  $x$  gesucht wird mit den Antworten „ja“ bzw. „nein“ und `out(x)`, dass  $x$  gelöscht wird, sofern  $x$  gespeichert ist.

Es wird folgende Sequenz von Operationen auf ein anfangs leeres Array ausgeführt:

`in(19)`, `in(29)`, `in(39)`, `in(10)`, `out(29)`, `out(39)`, `search(29)`, `in(11)`, `in(17)`, `out(10)`, `in(2)`, `in(22)`

Geben Sie den Inhalt von `hashfeld` an

nach `search(29)`  
nach `out(10)`  
und nach `in(22)`.

## 66115 / 2010 / Herbst / Thema 2 / Aufgabe 7

Konstruieren Sie eine Turingmaschine  $M$  mit  $L(M) = L$ , wobei  $p, q \geq 1$ . Beschreiben Sie zusätzlich, wie  $M$  arbeitet (Stil:  $M$  liest das Zeichen  $a$  und speichert ....)

$$L = \{ w \mid w \in \{a, b\}^*, w \text{ besteht aus } p \text{ Zeichen } a \text{ und aus } q \text{ Zeichen } b \}.$$

## 66115 / 2012 / Frühjahr / Thema 1 / Aufgabe 3

Beweisen Sie, dass folgende Sprache kontextfrei, aber nicht regulär ist.

$$C = \{ a^n b^m \mid n \geq m \geq 1 \}$$

Lösungsvorschlag

### Nachweis Kontextfrei über Grammatik

$$G = (\{S\}, \{a, b\}, P, S)$$

$$P = \left\{ \right.$$

$$S \rightarrow aSb \mid aS \mid ab$$

$$\left. \right\}$$

- Regel 1:  $aSb$

- Regel 2:  $aS$

- Regel 3:  $ab$

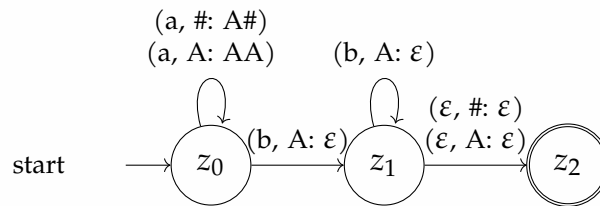
$$ab: S \xrightarrow{3} ab$$

$$a^n b: S \xrightarrow[n-1]{2} a^{n-1} S \xrightarrow{3} a^{n-1} ab$$

$$a^n b^m: S \xrightarrow[m-1]{1} a^{m-1} S b^{m-1} \xrightarrow[n-(m-1)]{2} a^{n-1} S b^{m-1} \xrightarrow{3} a^n b^m$$

$$\Rightarrow L(G) = C$$

### Nachweis Kontextfrei über Kellerautomat



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aji151myg

### Nachweis: C nicht regulär

C sei regulär

⇒ Pumping-Lemma für C erfüllt

$j$  sei die Pumping-Zahl ( $j \in \mathbb{N}$ )

$\omega \in C: \omega = a^j b^j$

$\omega = uvw$

Dann gilt:

- $|v| \geq 1$
- $|uv| \leq j$
- $uv^i w \in C$  für alle  $i \in \mathbb{N}_0$

In  $uv$  können nur  $a$ 's vorkommen

⇒ In  $v$  muss mindestens ein  $a$  vorkommen

⇒  $uv^0 w = a^l (a^{j-l})^0 b^j ((a^{j-l})^0 = \varepsilon)$

⇒ In  $\omega'$  sind nur  $l$  viele  $a$ 's, Da  $l < j$ ,  $\omega' \notin C$ ,

⇒ Widerspruch zur Annahme

⇒ C nicht regulär

### 66115 / 2012 / Frühjahr / Thema 1 / Aufgabe 4

Gegeben ist die kontextfreie Grammatik  $G = (V, \Sigma, P, S)$  mit  $\Sigma = \{a, b\}$ ,  $N = \{S, A, B\}$  und

$P = \{$

$S \rightarrow A$

$S \rightarrow B$

$A \rightarrow aAb$

$B \rightarrow AA$

$B \rightarrow bBa$

$A \rightarrow a$

$\}$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gr3rgt2vg

Geben Sie eine äquivalente Grammatik in Chomsky-Normalform an.

Lösungsvorschlag

Kann auch so geschrieben werden:

$$P = \left\{ \begin{array}{l} S \rightarrow A \mid B \\ A \rightarrow aAb \mid a \\ B \rightarrow AA \mid bBa \end{array} \right\}$$

(a) **Elimination der  $\varepsilon$ -Regeln**

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen. —

$\emptyset$  Nichts zu tun

(b) **Elimination von Kettenregeln**

— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren. —

$$P = \left\{ \begin{array}{l} S \rightarrow aAb \mid a \mid AA \mid bBa \\ A \rightarrow aAb \mid a \\ B \rightarrow AA \mid bBa \end{array} \right\}$$

(c) **Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt. —

$$P = \left\{ \begin{array}{l} S \rightarrow T_a A T_b \mid a \mid AA \mid T_b B T_a \\ A \rightarrow T_a A T_b \mid a \\ B \rightarrow AA \mid T_b B T_a \\ T_a \rightarrow a \\ T_b \rightarrow b \end{array} \right\}$$

**(d) Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1 B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1} B_n, A_{n-1} \rightarrow A_{n-2} B_{n-1}, \dots, A_2 \rightarrow B_1 B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht.

$$P = \{$$

$$S \rightarrow T_a C \mid a \mid AA \mid T_b D$$

$$A \rightarrow T_a C \mid a$$

$$B \rightarrow AA \mid T_b D$$

$$T_a \rightarrow a$$

$$T_b \rightarrow b$$

$$C \rightarrow AT_b$$

$$D \rightarrow BT_a$$

$$\}$$
**66115 / 2012 / Frühjahr / Thema 1 / Aufgabe 7**

Mit der Länge eines Pfads oder eines Kreises bezeichnen wir die Anzahl der Kanten, aus denen der Pfad bzw. der Kreis besteht. Bekanntlich kann man Breitensuche verwenden, um für zwei gegebene Knoten  $s$  und  $t$  die Länge eines kürzesten  $s$ - $t$ -Wegs zu berechnen. Im folgenden geht es um die Berechnung kürzester Kreise.

- (a) Für einen Graphen  $G$  und einen Knoten  $v$  von  $G$  berechnet  $KK(G, v)$  (siehe Abbildung 1) die Länge des kürzesten Kreises in  $G$ , der durch  $v$  geht.

Analysieren Sie die Laufzeit von  $KK$  in Abhängigkeit von der Anzahl  $n$  der Knoten von  $G$ , von der Anzahl  $m$  der Kanten von  $G$  und vom Grad  $\deg(v)$  des übergebenen Knotens  $v$ .

- (b) Wenn man den Algorithmus  $KK$  für jeden Knoten eines Graphen  $G$  aufruft, kann man die Länge eines kürzesten Kreises in  $G$  berechnen. Welche Laufzeit hat der resultierende Algorithmus in in Abhängigkeit von  $n$  und  $m$ ?
- (c) Geben Sie einen Algorithmus  $KK_{\text{schnell}}(G, v)$  an, der in  $O(n + m)$  Zeit die Länge des kürzesten Kreises in  $G$  berechnet, der durch  $v$  geht. Argumentieren Sie, warum ihr Algorithmus korrekt ist.

Abbildung 1

vum pam aba ee aan mn a sr lee

$KK(\text{ungerichteter UNBEWICHHLELEN rap1}$

$1L=2 \text{ Adi}): =weV \mid v, w \in E \text{ \& } \text{foreach } w \in \text{Adj}lvj \text{ do}$

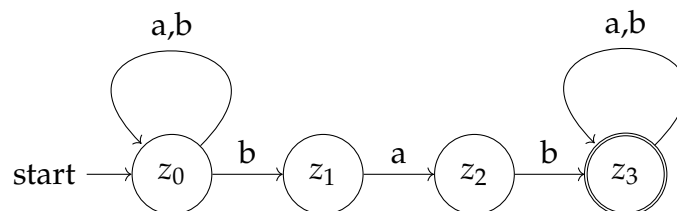
4 | Sei  $G'$  der Graph  $G$  ohne die Kante  $v, w$ .  
 5 Sei  $\ell$  die Länge eines kürzesten  $v$ - $w$ -Wegs in  $G'$ . 6 if  $2 < \ell$  then  
 7 | Lex  
 8 return  $\ell$

## 66115 / 2012 / Herbst / Thema 1 / Aufgabe 1

Wir fixieren das Alphabet  $\Sigma = \{a, b\}$  und definieren  $L \subseteq \Sigma^*$  durch

$$L = \{w \mid \text{in } w \text{ kommt das Teilwort } bab \text{ vor}\}$$

z. B. ist  $babaabb \in L$ , aber  $baabaabb \notin L$ . Der folgende nichtdeterministische Automat  $A$  erkennt  $L$ :

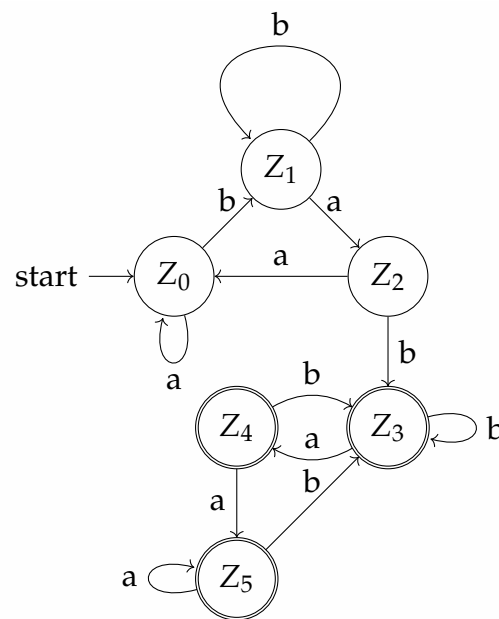


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Af75jwj3r

- (a) Wenden Sie die Potenzmengenkonstruktion auf den Automaten an und geben Sie den resultierenden deterministischen Automaten an. Nicht erreichbare Zustände sollen nicht dargestellt werden.

Lösungsvorschlag

| Zustandsmenge           | Eingabe a               | Eingabe b               |
|-------------------------|-------------------------|-------------------------|
| $Z_0 \{z_0\}$           | $Z_0 \{z_0\}$           | $Z_1 \{z_0, z_1\}$      |
| $Z_1 \{z_0, z_1\}$      | $Z_2 \{z_0, z_2\}$      | $Z_1 \{z_0, z_1\}$      |
| $Z_2 \{z_0, z_2\}$      | $Z_0 \{z_0\}$           | $Z_3 \{z_0, z_1, z_3\}$ |
| $Z_3 \{z_0, z_1, z_3\}$ | $Z_4 \{z_0, z_2, z_3\}$ | $Z_3 \{z_0, z_1, z_3\}$ |
| $Z_4 \{z_0, z_2, z_3\}$ | $Z_5 \{z_0, z_3\}$      | $Z_3 \{z_0, z_1, z_3\}$ |
| $Z_5 \{z_0, z_3\}$      | $Z_5 \{z_0, z_3\}$      | $Z_3 \{z_0, z_1, z_3\}$ |



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aro483e89

- (b) Konstruieren Sie aus dem so erhaltenen deterministischen Automaten den Minimalautomaten für  $L$ . Beschreiben Sie dabei die Arbeitsschritte des verwendeten Algorithmus in nachvollziehbarer Weise.

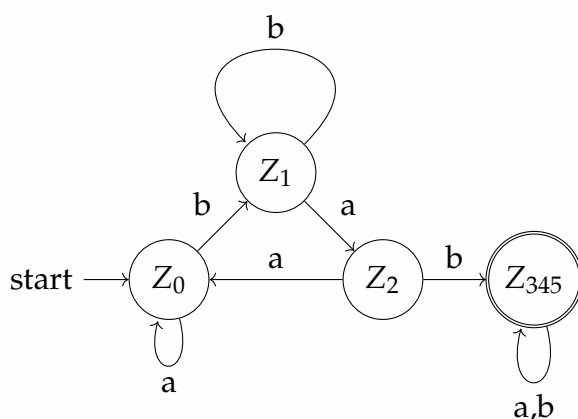
Lösungsvorschlag

|       |             |             |             |             |             |             |
|-------|-------------|-------------|-------------|-------------|-------------|-------------|
| $z_0$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $z_1$ | $x_3$       | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $z_2$ | $x_2$       | $x_2$       | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $z_3$ | $x_1$       | $x_1$       | $x_1$       | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $z_4$ | $x_1$       | $x_1$       | $x_1$       |             | $\emptyset$ | $\emptyset$ |
| $z_5$ | $x_1$       | $x_1$       | $x_1$       |             |             | $\emptyset$ |
|       | $z_0$       | $z_1$       | $z_2$       | $z_3$       | $z_4$       | $z_5$       |

- $x_1$  Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.  
 $x_2$  Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.  
 $x_3$  In weiteren Iterationen markierte Zustände.  
 $x_4$  ...

### Übergangstabelle

| Zustandspaar | a                  | b                  |
|--------------|--------------------|--------------------|
| $(z_0, z_1)$ | $(z_0, z_2) \ x_3$ | $(z_1, z_1)$       |
| $(z_0, z_2)$ | $(z_0, z_0)$       | $(z_1, z_3) \ x_2$ |
| $(z_1, z_2)$ | $(z_2, z_0) \ x_3$ | $(z_1, z_3) \ x_2$ |
| $(z_3, z_4)$ | $(z_4, z_5)$       | $(z_3, z_3)$       |
| $(z_3, z_5)$ | $(z_4, z_5)$       | $(z_3, z_3)$       |
| $(z_4, z_5)$ | $(z_5, z_5)$       | $(z_3, z_3)$       |



Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ar3joif5z](http://flaci.com/Ar3joif5z)

## 66115 / 2012 / Herbst / Thema 1 / Aufgabe 4

Gegeben ist ein Array  $a$  von ganzen Zahlen der Länge  $n$ , z. B. :

| $i$   | 0 | 1  | 2 | 3 | 4  | 5 | 6  | 7  | 8 | 9 |
|-------|---|----|---|---|----|---|----|----|---|---|
| $a_i$ | 5 | -6 | 4 | 2 | -5 | 7 | -2 | -7 | 3 | 5 |

Im Beispiel ist also  $n = 10$ . Es soll die maximale Teilsumme berechnet werden, also der Wert des Ausdrucks

$$\max_{i,j \leq n} \sum_{k=1}^{j-1} a_k$$

Im Beispiel ist dieser Wert 8 und wird für  $i = 8, j = 10$  erreicht. Entwerfen Sie ein Divide-And-Conquer Verfahren, welches diese Aufgabenstellung in Zeit  $\mathcal{O}(n \log n)$  löst. Skizzieren Sie Ihre Lösung hinreichend detailliert.

Tipp: Sie sollten ein geringfügig allgemeineres Problem lösen, welches neben der maximalen Teilsumme auch noch die beiden „maximalen Randsummen“ berechnet. Die werden dann bei der Endausgabe verworfen.



```
/**
 * Klasse zur Berechnung der maximalen Teilsumme einer Zahlenfolge.
 *
 * nach Teilsumme.java Klasse mit Algorithmen für die Berechnung des größten
 * gemeinsamen Teilers zweier Ganzzahlen Algorithmen und Datenstrukturen,
 * Auflage 4, Kapitel 2.1
 *
 * nach Prof. Grude, Prof. Solymosi, (c) 2000-2008: 22. April 2008
 * <a href="http://public.beuth-hochschule.de/oo-
↳ plug/A&D/prog/kap21/Teilsumme.java">Teilsumme.java
 */
public class Teilsumme {

 /**
 * Berechne die maximale Teilsumme an der rechten Grenze. Die Eingabeparameter
 * müssen diese Werte aufweisen: 0 <= links <= rechts <= folge.length.
 *
 * @param folge Die Zahlenfolge, in der die maximale Zahlensumme gerechnet
 * werden soll.
 * @param links Die Index-Nummer der linken Grenze.
 * @param rechts Die Index-Nummer der rechten Grenze.
 *
 * @return Die maximale Teilsumme.
 */
 private static int berechneRandRechts(int[] folge, int links, int rechts) {
 int max = 0;
 int sum = 0;
 for (int i = rechts; i >= links; i--) {
 sum += folge[i];
 max = Math.max(max, sum);
 }
 return max;
 }

 /**
 * Berechne die maximale Teilsumme an der linken Grenze. Die Eingabeparameter
 * müssen diese Werte aufweisen: 0 <= links <= rechts <= folge.length.
 *
 * @param folge Die Zahlenfolge, in der die maximale Zahlensumme gerechnet
 * werden soll.
 * @param links Die Index-Nummer der linken Grenze.
 * @param rechts Die Index-Nummer der rechten Grenze.
 *
 * @return Die maximale Teilsumme.
 */
 private static int berechneRandLinks(int[] folge, int links, int rechts) {
 int max = 0;
 int sum = 0;
 for (int i = links; i <= rechts; i++) {
 sum += folge[i];
 max = Math.max(max, sum);
 }
 return max;
 }
}
```

```

/**
 * Berechne die maximale Teilsumme in der Zahlenfolge zwischen einer gegebenen
 * linken und rechten Grenze. Die Eingabeparameter müssen diese Werte aufweisen:
 * 0 <= links <= rechts <= folge.length.
 *
 * @param folge Die Zahlenfolge, in der die maximale Zahlensumme gerechnet
 * werden soll.
 * @param links Die Index-Nummer der linken Grenze.
 * @param rechts Die Index-Nummer der rechten Grenze.
 *
 * @return Die maximale Teilsumme.
 */
private static int berechne(int[] folge, int links, int rechts) {
 if (links == rechts) // nur ein Element
 return Math.max(0, folge[links]);
 else {
 final int mitte = (rechts + links) / 2;
 final int maxLinks = berechne(folge, links, mitte);
 final int maxRechts = berechne(folge, mitte + 1, rechts);
 final int maxGrenzeRechts = berechneRandRechts(folge, links, mitte);
 // linke Hälfte
 final int maxGrenzeLinks = berechneRandLinks(folge, mitte + 1, rechts);
 // rechte Hälfte
 return Math.max(maxRechts, Math.max(maxLinks, maxGrenzeRechts +
 ↪ maxGrenzeLinks));
 }
}

/**
 * Berechne die maximale Teilsumme einer Zahlenfolge rekursiv mit
 * logarithmischer Zeitkomplexität.
 *
 * @param folge Die Zahlenfolge, in der die maximale Zahlensumme gerechnet
 * werden soll.
 *
 * @return Die maximale Teilsumme.
 */
public static int berechne(int[] folge) {
 return berechne(folge, 0, folge.length - 1);
}

public static void main(String[] args) {
 int[] folge = { 5, -6, 4, 2, -5, 7, -2, -7, 3, 5 };
 int ergebnis = berechne(folge);
 System.out.println(ergebnis);
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2012/herbst/Teilsumme.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66115/jahr_2012/herbst/Teilsumme.java)

## 66115 / 2012 / Herbst / Thema 2 / Aufgabe 6

Gegeben seien die Funktionen  $f: \mathbb{N} \rightarrow \mathbb{N}$  und  $g: \mathbb{N} \rightarrow \mathbb{N}$ , wobei  $f(n) = (n - 1)^3$  und  $g(n) = (2n + 3)(3n + 2)$ . Geben Sie an, welche der folgenden Aussagen gelten. Beweisen Sie Ihre Angaben.

(a)  $f(n) \in \mathcal{O}(g(n))$

(b)  $g(n) \in \mathcal{O}(f(n))$

### Exkurs: Regel von L'Hospital

Die Regel von de L'Hospital ist ein Hilfsmittel zum Berechnen von Grenzwerten bei Brüchen  $\frac{f}{g}$  von Funktionen  $f$  und  $g$ , wenn Zähler und Nenner entweder beide gegen 0 oder beide gegen (+ oder -) unendlich gehen. Wenn in einem solchen Fall auch der Grenzwert des Bruches der Ableitungen existiert, so hat dieser denselben Wert wie der ursprüngliche Grenzwert: <sup>a</sup>

$$\lim_{x \rightarrow x_0} \frac{f(x)}{g(x)} = \lim_{x \rightarrow x_0} \frac{f'(x)}{g'(x)}$$

<sup>a</sup><https://de.serlo.org/mathe/funktionen/grenzwerte-stetigkeit-differenzierbarkeit/grenzwert/regel-l-hospital>

Lösungsvorschlag

Es gilt Aussage (b), da  $f(n) \in \mathcal{O}(n^3)$  und  $g(n) \in \mathcal{O}(n^2)$  und der Grenzwert  $\lim$  bei größer werdendem  $n$  gegen  $\infty$  geht. Damit wächst  $f(n)$  stärker als  $g(n)$ , sodass nur Aussage (b) gilt und nicht (a). Dafür nutzen wir die formale Definition des  $\mathcal{O}$ -Kalküls, indem wir den Grenzwert  $\frac{f}{g}$  bzw.  $\frac{g}{f}$  bilden:

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{(n-1)^3}{(2n+3)(3n+2)} = \lim_{n \rightarrow \infty} \frac{3(n-1)^2}{(2n+3) \cdot 3 + 2 \cdot (3n+2)} = \lim_{n \rightarrow \infty} \frac{6(n-1)}{12} = \infty$$

$$\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = \lim_{n \rightarrow \infty} \frac{(2n+3)(3n+2)}{(n-1)^3} = \lim_{n \rightarrow \infty} \frac{(2n+3) \cdot 3 + 2 \cdot (3n+2)}{3(n-1)^2} = \lim_{n \rightarrow \infty} \frac{12}{6(n-1)} = 0$$

Hinweis: Hierbei haben wir die Regel von L'Hospital angewendet.

## 66115 / 2012 / Herbst / Thema 2 / Aufgabe 7

(a) Fügen Sie nacheinander die Zahlen 3, 5, 1, 2, 4

(i) in einen leeren binären Suchbaum ein

Nach dem Einfügen von „3“:

3

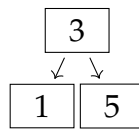
Nach dem Einfügen von „5“:

3

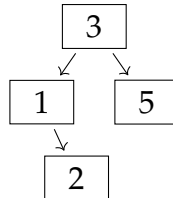
↓

5

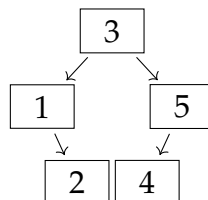
Nach dem Einfügen von „1“:



Nach dem Einfügen von „2“:

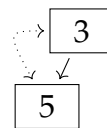


Nach dem Einfügen von „4“:

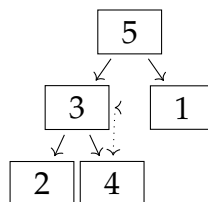


(ii) in einen leeren Heap ein

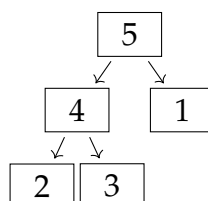
Erstellen einer Max.-Halde, einfügen von 3 und 5, Versickern notwendig:



Einfügen von 1 und 2 ohne Änderungen, Einfügen von 4, versickern notwendig:

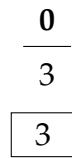


Fertiger Heap:

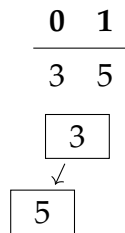


## Ausführlicher als Max-Halde

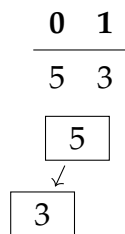
Nach dem Einfügen von „3“:



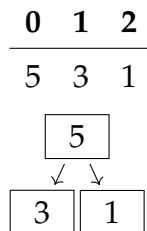
Nach dem Einfügen von „5“:



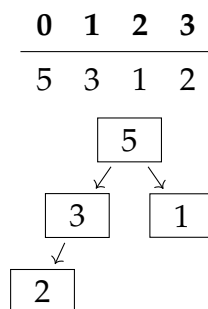
Nach dem Vertauschen von „5“ und „3“:



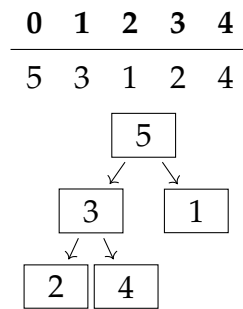
Nach dem Einfügen von „1“:



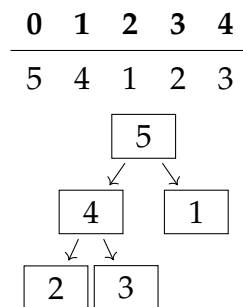
Nach dem Einfügen von „2“:



Nach dem Einfügen von „4“:

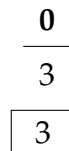


Nach dem Vertauschen von „4“ und „3“:

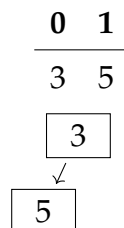


### Ausführlicher als Min-Halde

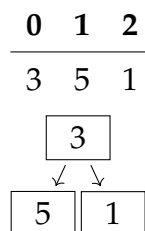
Nach dem Einfügen von „3“:



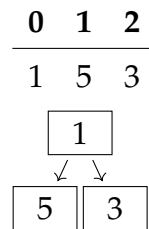
Nach dem Einfügen von „5“:



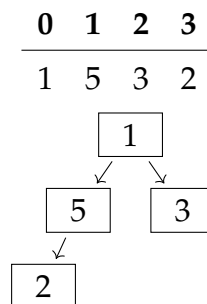
Nach dem Einfügen von „1“:



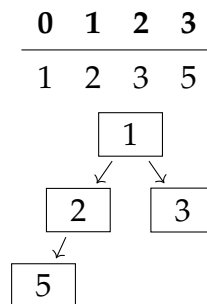
Nach dem Vertauschen von „1“ und „3“:



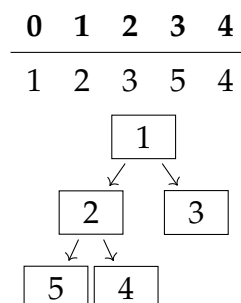
Nach dem Einfügen von „2“:



Nach dem Vertauschen von „2“ und „5“:



Nach dem Einfügen von „4“:



Geben Sie die Ergebnisse an (Zeichnung)

- (b) Geben Sie zwei Merkmale an, bei denen sich Heaps und binäre Suchbäume wesentlich unterscheiden. Ein wesentlicher Unterschied zwischen Bubblesort und Mergesort ist z. B. die *worst case* Laufzeit mit  $\mathcal{O}(n^2)$  für Bubblesort und  $\mathcal{O}(n \log n)$  für Mergesort.

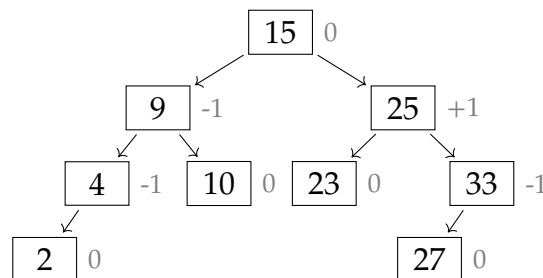
|                                     | Binärer Suchbaum       | Heap             |
|-------------------------------------|------------------------|------------------|
| Suchen beliebiger Wert (worst case) | $\mathcal{O}(\log(n))$ | $\mathcal{O}(n)$ |
| Suchen Min-Max (average case)       | $\mathcal{O}(\log(n))$ | $\mathcal{O}(1)$ |

<sup>a</sup>

<sup>a</sup><https://cs.stackexchange.com/q/27860>

## 66115 / 2012 / Herbst / Thema 2 / Aufgabe 8

Gegeben sei der folgende AVL-Baum  $T$ . Führen Sie auf  $T$  folgende Operationen durch.

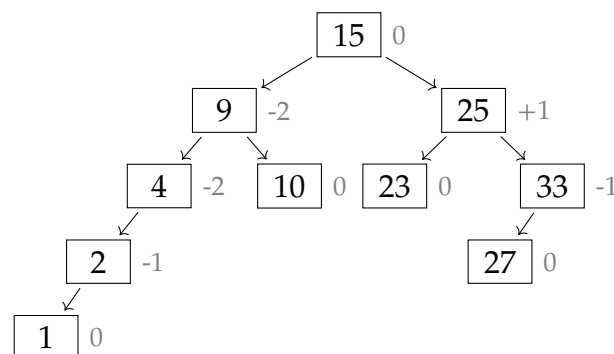


Wir führen alle Operationen am Ursprungsbaum  $T$  durch und nicht am veränderten Baum.

(a) Einfüge-Operationen:

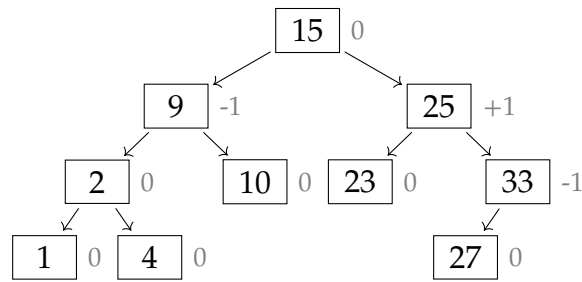
- (i) Fügen Sie den Wert 1 in  $T$  ein. Balancieren Sie falls nötig und geben Sie den entstandenen Baum (als Zeichnung) an.

Nach dem Einfügen von „1“:



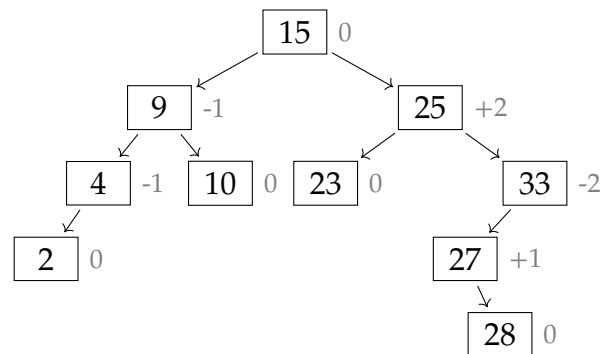
Nach der Rechtsrotation:



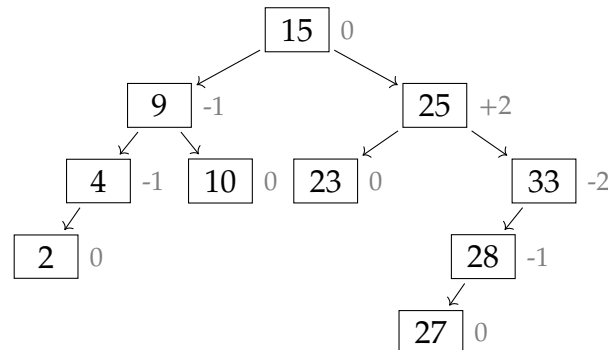


- (ii) Fügen Sie nun den Wert 28 in  $T$  ein. Balancieren Sie falls nötig und geben Sie den entstandenen Baum (als Zeichnung) an.

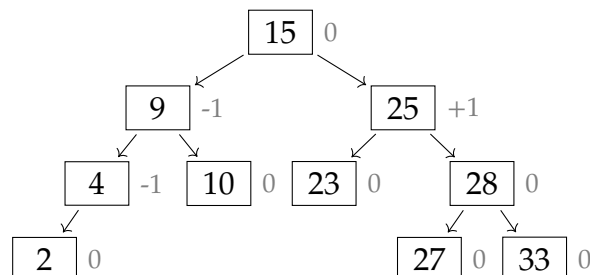
Nach dem Einfügen von „28“:



Nach der Linksrotation:

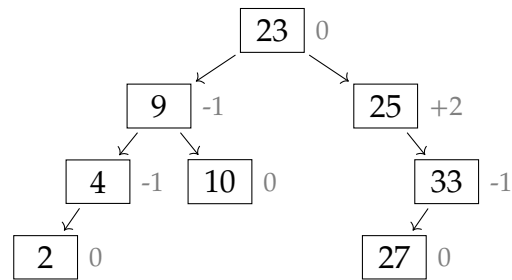


Nach der Rechtsrotation:

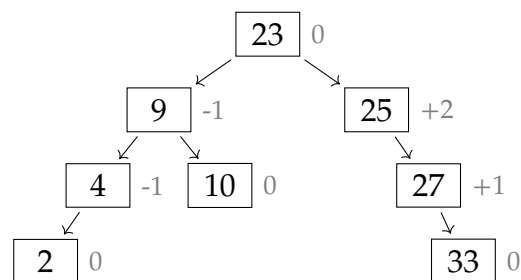


- (b) Löschen Sie aus  $T$  den Wert 15. Balancieren Sie falls nötig und geben Sie den entstandenen Baum (als Zeichnung) an.

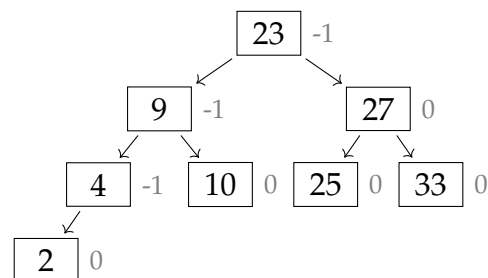
Nach dem Löschen von „15“:



*Nach der Rechtsrotation:*



*Nach der Linksrotation:*



## 66115 / 2013 / Frühjahr / Thema 1 / Aufgabe 2

Gegeben sei die Grammatik  $G = (\{S, A, B, C\}, \{a, b\}, P, S)$  und

$P = \{$

$S \rightarrow AB$

$S \rightarrow CS$

$A \rightarrow BC$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AC$

$B \rightarrow b$

$C \rightarrow AA$

$C \rightarrow BA$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gr46a6j0a

$L = L(G)$  ist die von  $G$  erzeugte Sprache.

(a) Zeigen Sie, dass  $G$  mehrdeutig ist.

Lösungsvorschlag

Das Wort  $baab$  kann in zwei verschiedenen Ableitungen hergeleitet werden:

(i)  $S \vdash AB \vdash BCB \vdash bCB \vdash bAAB \vdash baAB \vdash baaB \vdash baab$

(ii)  $S \vdash CS \vdash BAS \vdash bAS \vdash baS \vdash baAB \vdash baaB \vdash baab$

(b) Entscheiden Sie mithilfe des Algorithmus von Cocke, Younger und Kasami (CYK), ob das Wort  $w = babaaa$  zur Sprache  $L$  gehört. Begründen Sie Ihre Entscheidung.

Lösungsvorschlag

|       |       |     |   |   |   |
|-------|-------|-----|---|---|---|
| b     | a     | b   | a | a | a |
| B     | A     | B   | A | A | A |
| C     | S     | C   | C | C |   |
| -     | B     | A   | B |   |   |
| A     | C     | A,C |   |   |   |
| A,C   | B,C,A |     |   |   |   |
| A,C,B |       |     |   |   |   |

$\Rightarrow babaaa \notin L(G)$

Das Startsymbol  $S$  ist nicht in der Zelle  $V(1,5) = \{A,C,B\}$  enthalten.

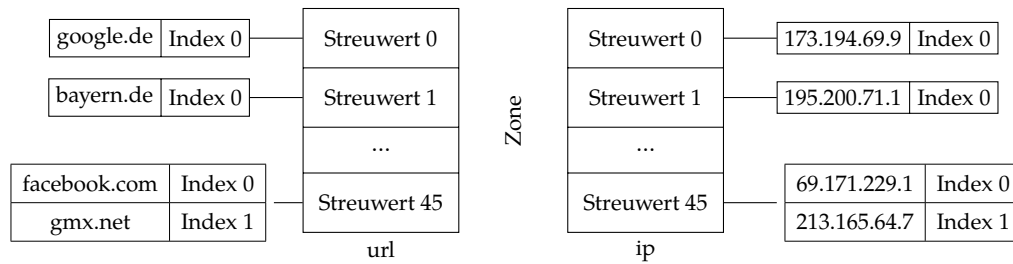
(c) Geben Sie eine Ableitung für  $w = babaaa$  an.

Lösungsvorschlag

$A \vdash BB \vdash bB \vdash bAC \vdash baC \vdash baAA \vdash baBCA \vdash babCA \vdash babAAA \vdash babaAA \vdash babaaA \vdash babaaa$

## 66115 / 2013 / Frühjahr / Thema 1 / Aufgabe 6

Um die URL (zum Beispiel google.de) und die zugehörige IP des Servers (hier 173.194.69.9) zu verwalten, werden Streutabellen verwendet, die eine bestimmte Zone von Adressen abbilden. Die Streutabellen werden als zwei dynamische Arrays (in Java: ArrayLists) realisiert. Kollisionen innerhalb einer Zone werden ebenfalls in dynamischen Arrays verwaltet.



Um zu einer URL die IP zu finden, berechnet man zunächst mittels der Funktion `hash()` den entsprechenden Streuwert, entnimmt dann den Index der Tabelle URL und sucht schließlich an entsprechender Stelle in der Tabelle IP die IP-Adresse.

- Erläutern Sie am vorgestellten Beispiel, wie ein Hash-Verfahren zum Speichern großer Datenmengen prinzipiell funktioniert und welche Voraussetzungen und Bedingungen daran geknüpft sind.
- Nun implementieren Sie Teile dieser IP- und URL-Verwaltung in einer objektorientierten Sprache Ihrer Wahl. Verwenden Sie dabei die folgende Klasse (die Vorgaben sind in der Sprache Java gehalten):

```
class Zone {
 private ArrayList<ArrayList<String>> urlList =
 new ArrayList<ArrayList<String>>();
 private ArrayList<ArrayList<String>> ipList =
 new ArrayList<ArrayList<String>>();
 public int hash(String url) { /* calculates hash-value h, >=0 */
 }
}
```

- Prüfen Sie in einer Methode `boolean exists(int h)` der Klasse `Zone`, ob bereits mindestens ein Eintrag für einen gegebenen Streuwert vorhanden ist. Falls `h` größer ist als die derzeitige Größe der Streutabelle, existiert der Eintrag nicht.

Lösungsvorschlag

```
boolean exists(int h) {
 if (urlList.size() - 1 < h || ipList.size() - 1 < h)
 return false;

 ArrayList<String> urlCollisionList = urlList.get(h);
 ArrayList<String> ipCollisionList = ipList.get(h);
 if (urlCollisionList.size() == 0 || ipCollisionList.size() == 0)
 return false;

 return true;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2013/fruehjahr/Zone.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java)

- Die Methode `int getIndex (string url, ArrayList<String> urlList)` soll den Index einer URL in der Kollisionsliste berechnen. Ist die URL in der Kollisionsliste nicht vorhanden, soll `-1` zurückgeliefert werden.

```
int getIndex(String url, ArrayList<String> urlList) {
 for (int i = 0; i < urlList.size(); i++) {
 if (urlList.get(i).equals(url))
 return i;
 }
 return -1;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2013/fruehjahr/Zone.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java)

- (iii) Ergänzen Sie die Klasse Zone um eine Methode `String lookup (String url)`, die in der Streutabelle die IP-Adresse zur `url` zurückgibt. Wird eine nicht vorhandene Adresse abgerufen, wird eine Fehlermeldung zurückgegeben.

```
String lookup(String url) {
 int h = hash(url);
 int collisionIndex = getIndex(url, urlList.get(h));
 if (collisionIndex == -1)
 return "Die URL konnte nicht in der Tabelle gefunden werden";
 return ipList.get(h).get(collisionIndex);
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2013/fruehjahr/Zone.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2013/fruehjahr/Zone.java)

## 66115 / 2013 / Frühjahr / Thema 2 / Aufgabe 5

Drei Missionare und drei Kannibalen befinden sich am Ufer eines Flusses und möchten diesen überqueren. Dazu steht ihnen ein Boot zur Verfügung, das ein oder zwei Personen befördern kann. Verbleiben an einem Ufer mehr Kannibalen als Missionare, so werden die Missionare verspeist. Die Frage besteht nun darin, wie alle Personen unversehr auf die andere Seite des Flusses gelangen.

Im Anfangszustand befinden sich alle Personen und das Boot auf einer Seite des Flusses. Nehmen Sie an, es sei die linke Seite des Flusses. Im Zielzustand befinden sich alle Personen und das Boot auf der anderen Seite des Flusses. Jeden Zustand kann man durch folgendes Fünftupel beschreiben:

Missionareyngs X Kannibalen;;,xs X Missionareyeens X Kannibalenyechus X Bool-position

Dabei werden die Elemente für Missionare und Kannibalen durch natürliche Zahlen und die Bootsposition durch einen der Strings „links“ oder „rechts“ modelliert. Der Anfangszustand wäre somit also (3, 3, 0, 0, » links“) und der Zielzustand (0, 0, 3, 3, „rechts“).

Schreiben Sie Algorithmen zur Lösung dieses Suchproblems und retten Sie die Missionare! Es ist empfehlenswert (aber nicht zwingend), hierzu eine funktionale Programmiersprache zu verwenden. Gehen Sie im Einzelnen vor, wie folgt:

- (a) a.) Schreiben Sie eine Funktion, die alle möglichen Überfahrten von links nach rechts oder umgekehrt modelliert, d. h. die zu einem gegebenen Zustand die Liste der möglichen Folgezustände im Sinne der o. g. Regeln berechnet. Gehen Sie

dazu von allen möglichen Überfahrten aus und überprüfen Sie für jede konkrete Überfahrt mittels einer geeigneter Funktion, ob diese zu einem zulässigen Zustand führt. Zustände, die die Missionare nicht überleben, gelten im Sinne des Rettungsvorhabens ebenfalls als nicht zulässig. Nicht zulässige Zustände werden nicht in die Liste der möglichen Folgezustände eingefügt.

- (b) b.) Geben Sie eine Funktion an, die feststellt, ob ein Zyklus vorliegt, d. h. ob ein Zustand in einer Liste bereits besuchter Zustände schon enthalten ist.
- (c) c.) Verwenden Sie Ihre Ergebnisse aus a.) und b.), um eine Funktion anzugeben, die dieses Suchproblem mittels Breitensuche löst. (Sie können die Funktionen aus a.) und b.) hier auch dann verwenden, wenn Sie diese Teilaufgaben nicht vollständig gelöst haben.) Die Funktion erhält als Eingabe einen Start- und einen Zielzustand und liefert als Ergebnis die erste gefundene Liste von Zuständen, die das Problem löst.

### 66115 / 2013 / Herbst / Thema 1 / Aufgabe 3

Wir betrachten das wie folgt definierte Problem DOPP:

**GEGEBEN:** Eine deterministische Turingmaschine  $M$ , eine Eingabe  $x$  (für  $M$ ), ein Zustand  $q$  (von  $M$ ).

**GEFRAGT:** Wird der Zustand  $q$  bei der Berechnung von  $M$  auf  $x$  mindestens zweimal besucht?

- (a) Zeigen Sie durch Angabe einer Reduktion vom Halteproblem, dass DOPP unentscheidbar.
- (b) Begründen Sie, dass DOPP rekursiv aufzählbar (semi-entscheidbar) ist.

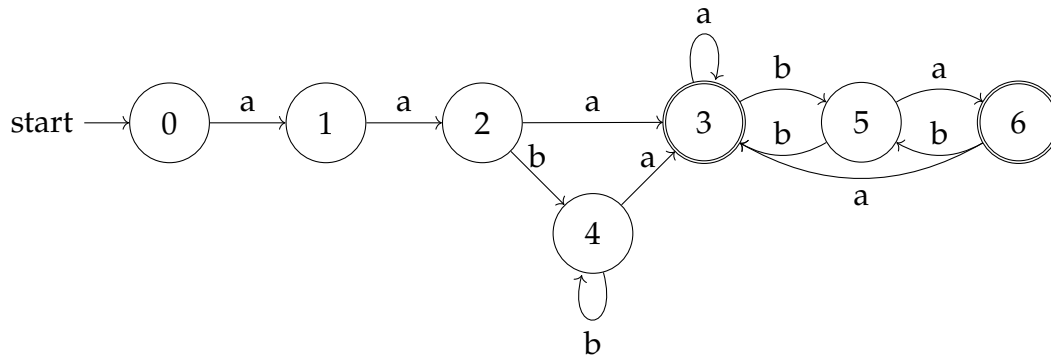
Die Reduktion  $f: \text{HALTE} \leq_p \text{DOPP}$ , bildet  $c(M), w$  auf  $c(M'), xw, q$  ab, wobei  $M'$  eine Turingmaschine mit folgendem Verhalten ist:

- Sie verfügt über den neuen Startzustand  $q$
- Sie erweitert das Wort  $w$  vorne um ein Zeichen  $x \notin \Gamma_M$  (dies ist nicht unbedingt notwendig, aber schöner, damit die neue Maschine auch noch terminiert)
- Für  $q$  wird die Regel  $(q, x) \rightarrow (z_0, \square, R)$ , wobei  $z_0$  der Startzustand von  $M$  ist
- Alle Endzustände  $z$  von  $M$  erhalten eine neue Regel  $(z, \square) \rightarrow (q, \square, N)$

Hierbei wird davon ausgegangen, dass das Halteproblem eine Turingmaschine mit Endzuständen als Eingabe hat und Aufgrund der Akzeptanz eines Wortes in diesem Fall das Zeichen gelöscht wird.

## 66115 / 2013 / Herbst / Thema 2 / Aufgabe 3

Minimieren Sie den folgenden deterministischen Automaten mit den Zuständen  $\{0, 1, 2, 3, 4, 5, 6\}$ , dem Startzustand 0 und den Endzuständen  $\{3, 6\}$ . Geben Sie z. B. durch die Bezeichnung an, welche Zustände zusammengefasst wurden.



Lösungsvorschlag

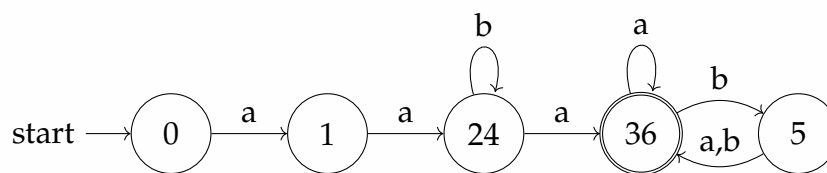
|   |             |             |             |             |             |             |             |
|---|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0 | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| 1 | $x_3$       | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| 2 | $x_2$       | $x_2$       | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| 3 | $x_1$       | $x_1$       | $x_1$       | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| 4 | $x_2$       | $x_2$       |             | $x_1$       | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| 5 | $x_2$       | $x_2$       | $x_2$       | $x_1$       | $x_2$       | $\emptyset$ | $\emptyset$ |
| 6 | $x_1$       | $x_1$       | $x_1$       |             | $x_1$       | $x_1$       | $\emptyset$ |
|   | 0           | 1           | 2           | 3           | 4           | 5           | 6           |

- $x_1$  Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.  
 $x_2$  Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.  
 $x_3$  In weiteren Iterationen markierte Zustände.  
 $x_4$  ...

## Übergangstabelle

| Zustandspaar | a            | b            |
|--------------|--------------|--------------|
| (0, 1)       | (1, 2) $x_3$ | (T, T)       |
| (0, 2)       | (1, 3) $x_2$ | (T, 4)       |
| (0, 4)       | (1, 3) $x_2$ | (T, 4)       |
| (0, 5)       | (1, 6) $x_2$ | (T, 3)       |
| (1, 2)       | (2, 3) $x_2$ | (T, 4)       |
| (1, 4)       | (2, 3) $x_2$ | (T, 4)       |
| (1, 5)       | (2, 6) $x_2$ | (T, 3)       |
| (2, 4)       | (3, 3)       | (4, 4)       |
| (2, 5)       | (3, 6)       | (3, 4) $x_2$ |
| (3, 6)       | (3, 3)       | (5, 5)       |
| (4, 5)       | (3, 6)       | (3, 4) $x_2$ |

T = Trap-Zustand = Falle

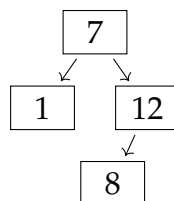


## 66115 / 2013 / Herbst / Thema 2 / Aufgabe 7

(a)

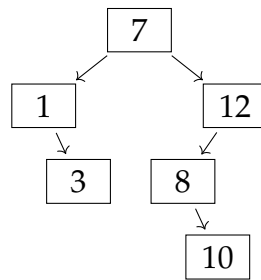
- (i) Fügen Sie nacheinander die Zahlen 7, 1, 12, 8, 10, 3, 5 in einen leeren binären Suchbaum ein und zeichnen Sie den Suchbaum nach „8“ und nach „3“.

Nach dem Einfügen von „8“:

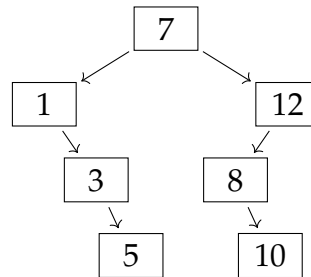


Nach dem Einfügen von „3“:



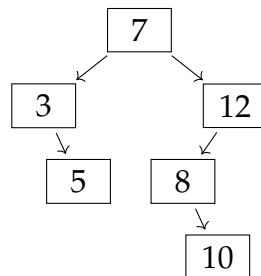


Nach dem Einfügen von „5“:



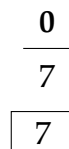
- (ii) Löschen Sie die „1“ aus dem in (i) erstellten Suchbaum und zeichnen Sie den Suchbaum.

Nach dem Löschen von „1“:

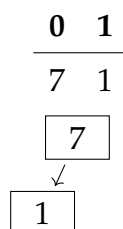


- (iii) Fügen Sie 7, 1, 12, 8, 10, 3, 5 in einen leeren MIN-Heap ein, der bzgl. „ $\leq$ “ angeordnet ist. Geben Sie den Heap nach jedem Element an.

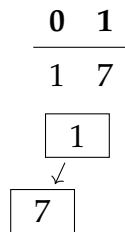
Nach dem Einfügen von „7“:



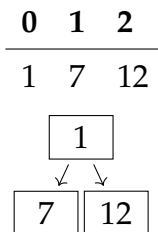
Nach dem Einfügen von „1“:



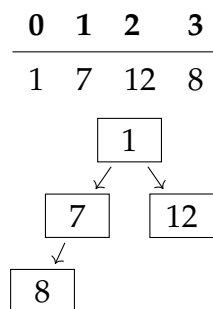
Nach dem Vertauschen von „1“ und „7“:



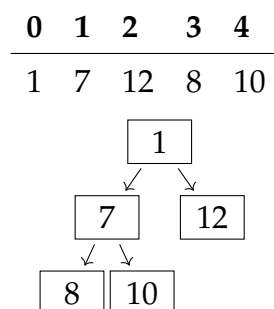
Nach dem Einfügen von „12“:



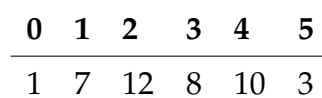
Nach dem Einfügen von „8“:

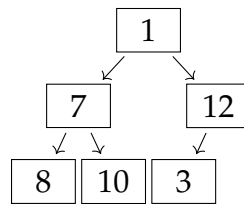


Nach dem Einfügen von „10“:

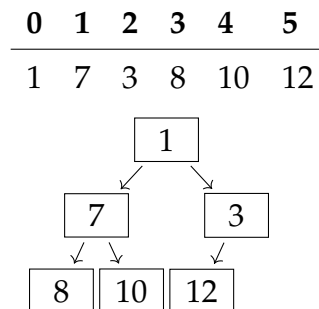


Nach dem Einfügen von „3“:

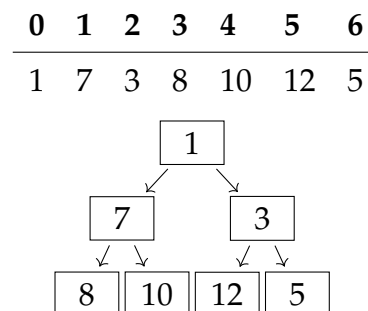




Nach dem Vertauschen von „3“ und „12“:



Nach dem Einfügen von „5“:



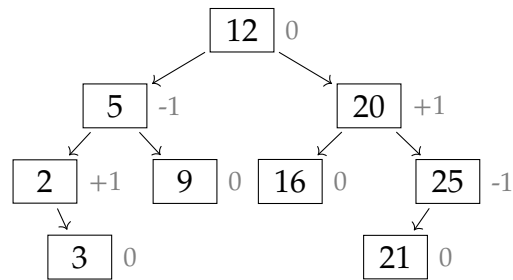
- (b) Was ist die worst-case Laufzeit in O-Notation für das Einfügen eines Elements in einen Heap der Größe  $n$ ? Begründen Sie ihre Antwort.

Lösungsvorschlag

Die worst-case Laufzeit berechnet sich aus dem Aufwand für das Durchsickern eines eingefügten Elementes. Da das Durchsickern entlang eines Pfades im Baum erfolgt, entspricht der Aufwand im ungünstigsten Fall der Höhe des Baumes,  $\log_2 n$ . Insgesamt ergibt sich somit eine worst-case Laufzeit von  $\mathcal{O}(\log n)$ .

## 66115 / 2013 / Herbst / Thema 2 / Aufgabe 8

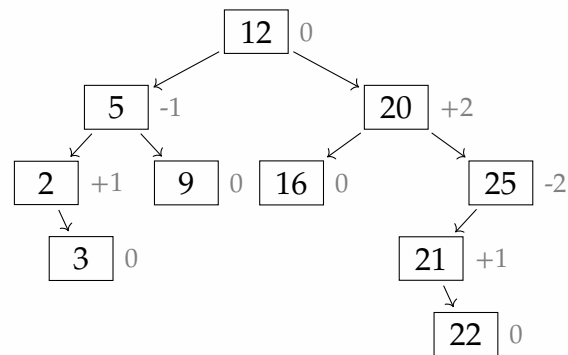
Gegeben sei der folgende AVL-Baum  $T$ . Führen Sie auf  $T$  folgende Operationen durch.



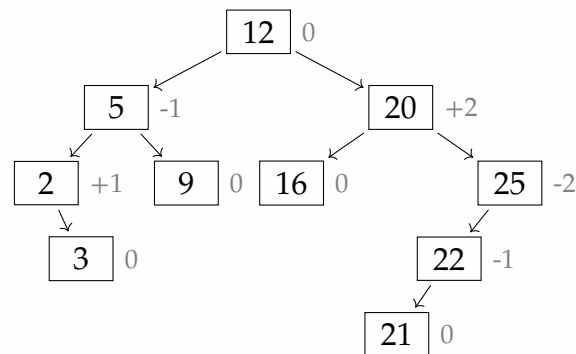
- (a) Fügen Sie den Wert 22 in  $T$  ein. Balancieren Sie falls nötig und geben Sie den entstandenen Baum (als Zeichnung) an.

Lösungsvorschlag

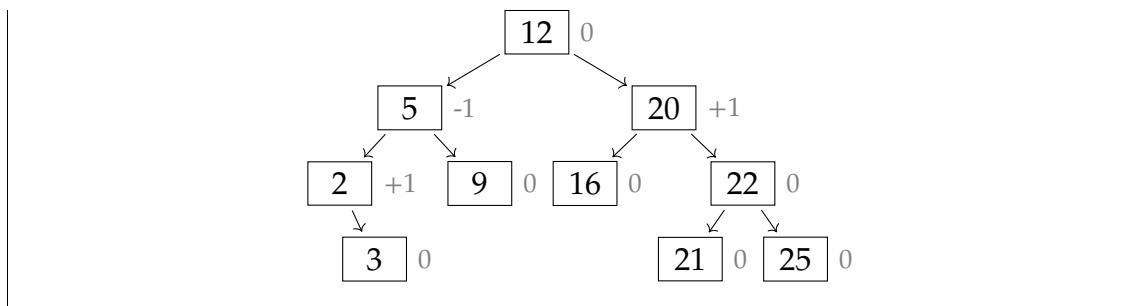
Nach dem Einfügen von „22“:



Nach der Linksrotation:



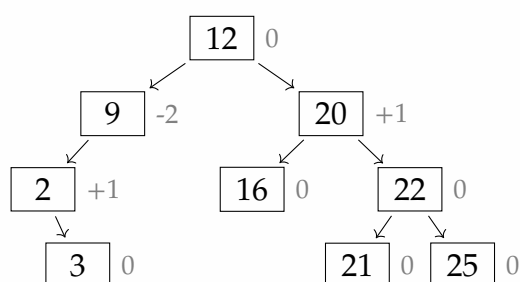
Nach der Rechtsrotation:



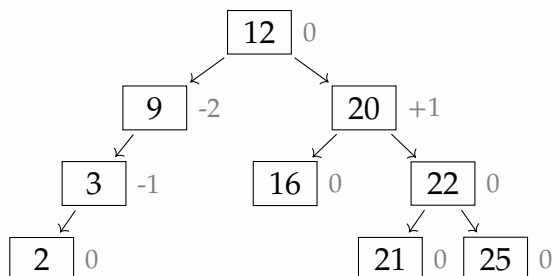
- (b) Löschen Sie danach die 5. Balancieren Sie  $T$  falls nötig und geben Sie den entstandenen Baum (als Zeichnung) an.

Lösungsvorschlag

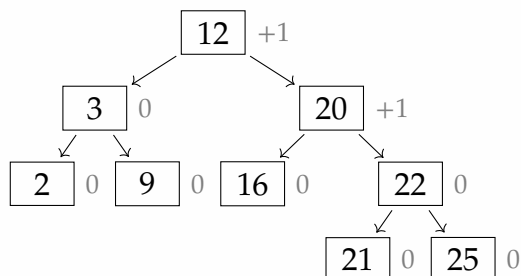
Nach dem Löschen von „5“:



Nach der Linksrotation:



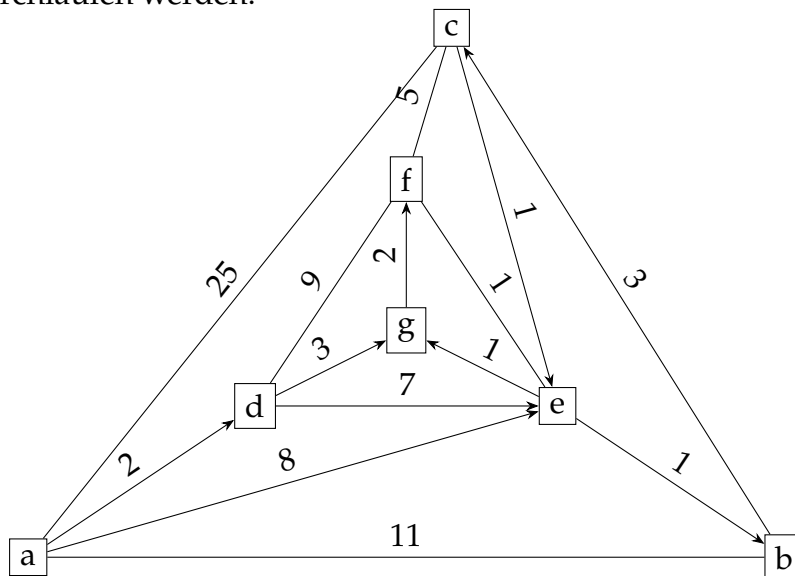
Nach der Rechtsrotation:



## 66115 / 2013 / Herbst / Thema 2 / Aufgabe 9

Gegeben sei der unten stehende gerichtete Graph  $G = (V, E)$  mit positiven Kantenlängen  $l(e)$  für jede Kante  $e \in E$ . Kanten mit Doppelspitzen können in beide Richtungen

durchlaufen werden.



- (a) In welcher Reihenfolge werden die Knoten von  $G$  ab dem Knoten  $a$  durch den Dijkstra-Algorithmus bei der Berechnung der kürzesten Wege endgültig bearbeitet?

Lösungsvorschlag

| Nr. | besucht | a        | b        | c         | d        | e        | f        | g        |
|-----|---------|----------|----------|-----------|----------|----------|----------|----------|
| 0   |         | 0        | $\infty$ | $\infty$  | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| 1   | a       | <b>0</b> | 11       | 25        | 2        | 8        | $\infty$ | $\infty$ |
| 2   | d       |          | 11       | 25        | <b>2</b> | 8        | 11       | 5        |
| 3   | g       |          | 11       | 25        |          | 8        | 7        | <b>5</b> |
| 4   | f       |          | 11       | 12        |          | 8        | <b>7</b> |          |
| 5   | e       |          | 9        | 12        |          | <b>8</b> |          |          |
| 6   | b       |          | <b>9</b> | 12        |          |          |          |          |
| 7   | c       |          |          | <b>12</b> |          |          |          |          |

- (b) Berechnen Sie die Länge des kürzesten Weges von  $a$  zu jedem Knoten.

Lösungsvorschlag

siehe oben

- (c) Geben Sie einen kürzesten Weg von  $a$  nach  $c$  an.

Lösungsvorschlag

$a \rightarrow d \rightarrow g \rightarrow f \rightarrow c$

## 66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 1

- (a) Gegeben sei die Methode `BigInteger lfBig(int n)` zur Berechnung der eingeschränkten Linksfakultät:

$$!n := \begin{cases} n!(n-1) - (n-1)!(n-2) & \text{falls } 1 < n < 32767 \\ 1 & \text{falls } n = 1 \\ 0 & \text{sonst} \end{cases}$$

```
import java.math.BigInteger;
import static java.math.BigInteger.ZERO;
import static java.math.BigInteger.ONE;

public class LeftFactorial {

 BigInteger sub(BigInteger a, BigInteger b) {
 return a.subtract(b);
 }

 BigInteger mul(BigInteger a, BigInteger b) {
 return a.multiply(b);
 }

 BigInteger mul(int a, BigInteger b) {
 return mul(BigInteger.valueOf(a), b);
 }

 // returns the left factorial !n
 BigInteger lfBig(int n) {
 if (n <= 0 || n >= Short.MAX_VALUE) {
 return ZERO;
 } else if (n == 1) {
 return ONE;
 } else {
 return sub(mul(n, lfBig(n - 1)), mul(n - 1, lfBig(n - 2)));
 }
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/fruehjahr/LeftFactorial.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/LeftFactorial.java)

Implementieren Sie unter Verwendung des Konzeptes der *dynamischen Programmierung* die Methode `BigInteger dp(int n)`, die jede  $!n$  auch bei mehrfachem Aufrufen mit dem gleichen Parameter höchstens einmal rekursiv berechnet. Sie dürfen der Klasse `LeftFactorial` genau ein Attribut beliebigen Datentyps hinzufügen und die in `lfBig(int)` verwendeten Methoden und Konstanten ebenfalls nutzen.

Lösungsvorschlag

Wir führen ein Attribut mit dem Namen `store` ein und erzeugen ein Feld vom Typ `BigInteger` mit der Länge  $n + 1$ . Die Länge des Feld  $n + 1$  hat den Vorteil, dass nicht ständig  $n - 1$  verwendet werden muss, um den gewünschten Wert zu erhalten.

In der untenstehenden Implementation gibt es zwei Methoden mit dem Namen `dp`. Die untenstehende Methode ist nur eine Hüllmethode, mit der nach außen hin die Berechnung gestartet und das `store`-Feld neu gesetzt wird. So ist es möglich `dp()` mehrmals hintereinander mit verschiedenen Werten aufzurufen (siehe `main()`-Methode).

```
BigInteger[] store;

BigInteger dp(int n, BigInteger[] store) {
 if (n > 1 && store[n] != null) {
 return store[n];
 }
 if (n <= 0 || n >= Short.MAX_VALUE) {
 return ZERO;
 } else if (n == 1) {
 return ONE;
 } else {
 BigInteger result = sub(mul(n, dp(n - 1, store)), mul(n - 1, dp(n - 2,
 ↪ store)));
 store[n] = result;
 return result;
 }
}

BigInteger dp(int n) {
 store = new BigInteger[n + 1];
 return dp(n, store);
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/fruehjahr/LeftFactorial.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/LeftFactorial.java)

- (b) Betrachten Sie nun die Methode `lfLong(int)` zur Berechnung der vorangehend definierten Linksfakultät ohne obere Schranke. Nehmen Sie im Folgenden an, dass der Datentyp `long` unbeschränkt ist und daher kein Überlauf auftritt.

```
long lfLong(int n) {
 if (n <= 0) {
 return 0;
 } else if (n == 1) {
 return 1;
 } else {
 return n * lfLong(n - 1) - (n - 1) * lfLong(n - 2);
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/fruehjahr/LeftFactorial.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/LeftFactorial.java)

Beweisen Sie *formal* mittels *vollständiger Induktion*:

$$\forall n \geq 0 : \text{lfLong}(n) \equiv \sum_{k=0}^{n-1} k!$$



**Induktionsanfang**

— Beweise, dass  $A(1)$  eine wahre Aussage ist. \_\_\_\_\_

$$n = 1 \Rightarrow \text{lfLong}(1) = 1 = \sum_{k=0}^{n-1} k! = 0! = 1$$

$$\begin{aligned} n = 2 &\Rightarrow \text{lfLong}(2) \\ &= (n+1) \sum_{k=0}^{n-1} k! - n \sum_{k=0}^{n-2} k! \\ &= 2 * \text{lfLong}(1) - 1 * \text{lfLong}(0) \\ &= 2 \\ &= \sum_{k=0}^1 k! \\ &= 1! + 0! \\ &= 1 + 1 \\ &= 2 \end{aligned}$$

**Induktionsvoraussetzung**

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . \_\_\_\_\_

$$\text{lfLong}(n) = \sum_{k=0}^{n-1} k!$$

**Induktionsschritt**

— Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss.

$$\begin{aligned}
A(n+1) &= \text{lfLong}(n+1) \\
&= (n+1) * \text{lfLong}(n) - n * \text{lfLong}(n-1) \\
&= (n+1) \sum_{k=0}^{n-1} k! - n \sum_{k=0}^{(n-1)-1} k! && \text{Formel eingesetzt} \\
&= (n+1) \sum_{k=0}^{n-1} k! - n \sum_{k=0}^{n-2} k! && \text{subtrahiert} \\
&= n \sum_{k=0}^{n-1} k! + \sum_{k=0}^{n-1} k! - n \sum_{k=0}^{n-2} k! && \text{ausmultipliziert mit } (n+1) \\
&= \sum_{k=0}^{n-1} k! + n \sum_{k=0}^{n-1} k! - n \sum_{k=0}^{n-2} k! && \text{Reihenfolge der Terme geändert} \\
&= \sum_{k=0}^{n-1} k! + n \left( (n-1)! + \sum_{k=0}^{n-2} k! \right) - n \sum_{k=0}^{n-2} k! && (n-1)! \text{ aus Summenzeichen entfernt} \\
&= \sum_{k=0}^{n-1} k! + n \left( (n-1)! + \sum_{k=0}^{n-2} k! - \sum_{k=0}^{n-2} k! \right) && \text{Distributivgesetz } ac - bc = (a-b)c \\
&= \sum_{k=0}^{n-1} k! + n(n-1)! && +\Sigma - \Sigma = 0 \\
&= \sum_{k=0}^{n-1} k! + n! && \text{Fakultät erhöht} \\
&= \sum_{k=0}^n k! && \text{Element zum Summenzeichen hinzugefügt} \\
&= \sum_{k=0}^{(n+1)-1} k! && \text{mit } (n+1) \text{ an der Stelle von } n
\end{aligned}$$

## 66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 2

Implementieren Sie in einer objekt-orientierten Sprache Ihrer Wahl eine Klasse namens `BinBaum`, deren Instanzen binäre Bäume mit ganzzahligen Datenknoten darstellen, nach folgender Spezifikation:

(a) Beginnen Sie zunächst mit der Datenstruktur selbst:

- Mit Hilfe des Konstruktors soll ein neuer Baum erstellt werden, der aus einem einzelnen Knoten besteht, in dem der dem Konstruktor als Parameter übergebene Wert (ganzzahlig, in Java z.B. `int`) gespeichert ist.

Lösungsvorschlag

```
class Knoten {
 int value;

 Knoten left;
 Knoten right;

 public Knoten(int value) {
 this.value = value;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/fruehjahr/BinBaum.java](#)

- Die Methoden `setLeft(int value)` bzw. `setRight(int value)` sollen den linken bzw. rechten Teilbaum des aktuellen Knotens durch einen jeweils neuen Teilbaum ersetzen, der seinerseits aus einem einzelnen Knoten besteht, in dem der übergebene Wert `value` gespeichert ist. Sie haben keinen Rückgabewert.

Lösungsvorschlag

```
public void setLeft(Knoten left) {
 this.left = left;
}

public void setRight(Knoten right) {
 this.right = right;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/fruehjahr/BinBaum.java](#)

- Die Methoden `getLeft()` bzw. `getRight()` sollen den linken bzw. rechten Teilbaum zurückgeben (bzw. `null`, wenn keiner vorhanden ist)

Lösungsvorschlag

```
public Knoten getLeft() {
 return left;
}

public Knoten getRight() {
 return right;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/fruehjahr/BinBaum.java](#)

- Die Methode `int getValue()` soll den Wert zurückgeben, der im aktuellen Wurzelknoten gespeichert ist.

Lösungsvorschlag

```
public int getValue() {
 return value;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/fruehjahr/BinBaum.java](#)

- (b) Implementieren Sie nun die Methoden `preOrder()` bzw. `postOrder()`. Sie sollen die Knoten des Baumes mit Tiefensuche traversieren, die Werte dabei in pre-order bzw. post-order Reihenfolge in eine Liste (z.B. `List<Integer>`) speichern und diese Ergebnisliste zurückgeben. Die Tiefensuche soll dabei zuerst in den linken und dann in den rechten Teilbaum absteigen.

Lösungsvorschlag

```

void preOrder(Knoten knoten, List<Integer> list) {
 if (knoten != null) {
 list.add(knoten.getValue());
 preOrder(knoten.getLeft(), list);
 preOrder(knoten.getRight(), list);
 }
}

List<Integer> preOrder() {
 List<Integer> list = new ArrayList<>();
 preOrder(head, list);
 return list;
}

void postOrder(Knoten knoten, List<Integer> list) {
 if (knoten != null) {
 postOrder(knoten.getLeft(), list);
 postOrder(knoten.getRight(), list);
 list.add(knoten.getValue());
 }
}

List<Integer> postOrder() {
 List<Integer> list = new ArrayList<>();
 postOrder(head, list);
 return list;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/fruehjahr/BinBaum.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java)

- (c) Ergänzen Sie schließlich die Methode `isSearchTree()`. Sie soll überprüfen, ob der Binärbaum die Eigenschaften eines binären Suchbaums erfüllt. Beachten Sie, dass die Laufzeit-Komplexität Ihrer Implementierung linear zur Anzahl der Knoten im Baum sein muss.

Lösungsvorschlag

```

boolean isSearchTree(Knoten knoten) {
 if (knoten == null) {
 return true;
 }

 if (knoten.getLeft() != null && knoten.getValue() <
 ↪ knoten.getLeft().getValue()) {
 return false;
 }
}

```

```

if (knoten.getRight() != null && knoten.getValue() >
 ↪ knoten.getRight().getValue()) {
 return false;
}

return isSearchTree(knoten.getLeft()) && isSearchTree(knoten.getRight());
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/fruehjahr/BinBaum.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java)

## 66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 3

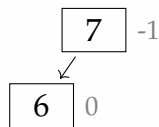
- (a) Fügen Sie die Zahlen (7, 6, 2, 1, 5, 3, 8, 4) in dieser Reihenfolge in einen anfangs leeren AVL Baum ein. Stellen Sie die AVL Eigenschaft ggf. nach jedem Einfügen mit geeigneten Rotationen wieder her. Zeichnen Sie den AVL Baum einmal vor und einmal nach jeder einzelnen Rotation.

Lösungsvorschlag

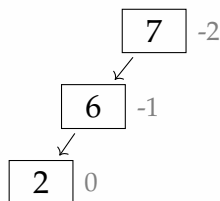
Nach dem Einfügen von „7“:



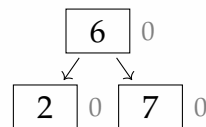
Nach dem Einfügen von „6“:



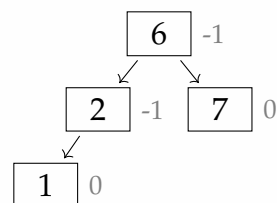
Nach dem Einfügen von „2“:



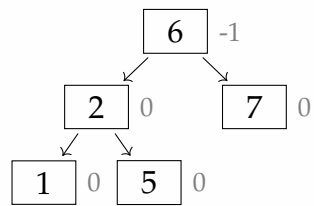
Nach der Rechtsrotation:



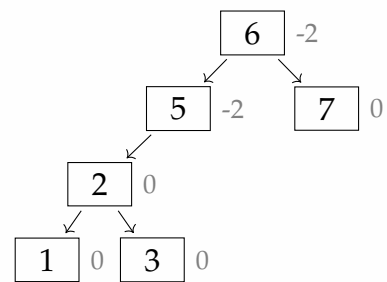
Nach dem Einfügen von „1“:



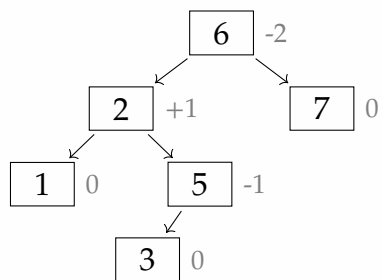
Nach dem Einfügen von „5“:



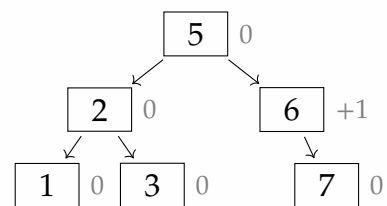
Nach der Linksrotation:



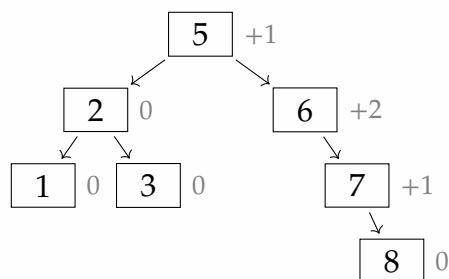
Nach dem Einfügen von „3“:



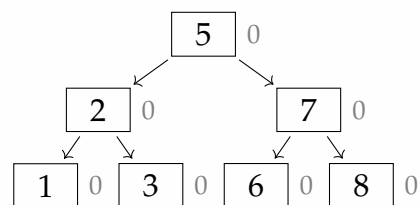
Nach der Rechtsrotation:



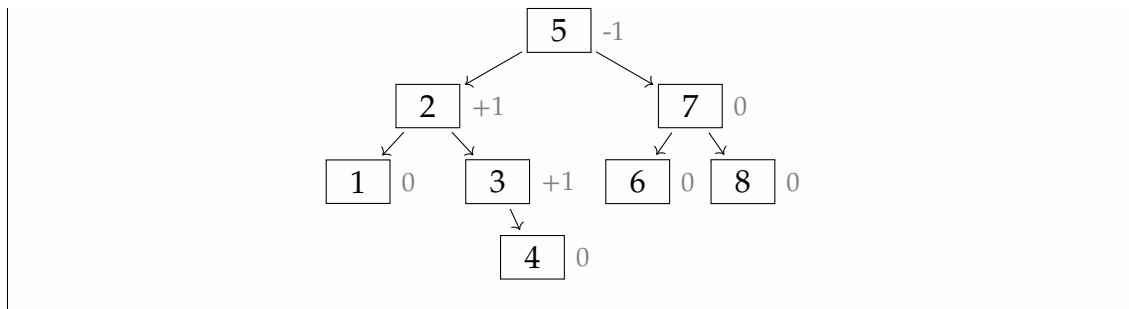
Nach dem Einfügen von „8“:



Nach der Linksrotation:

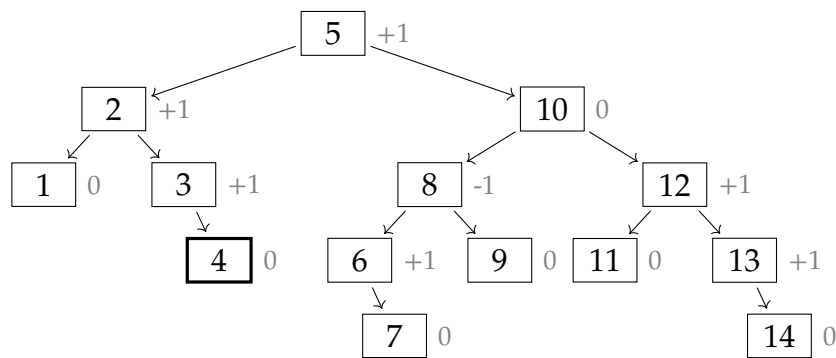


Nach dem Einfügen von „4“:



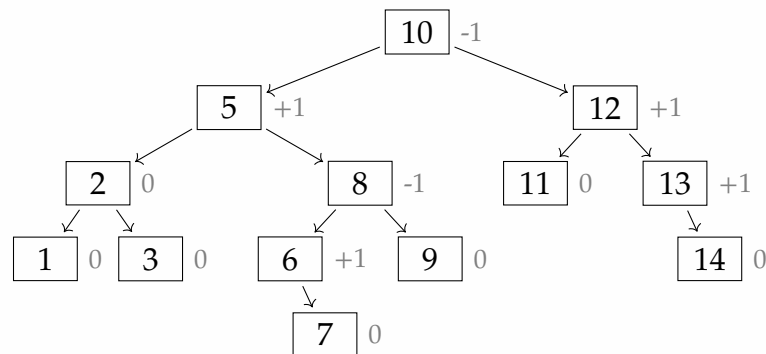
- (b) Entfernen Sie den jeweils markierten Knoten aus den folgenden AVL-Bäumen. Stellen Sie die AVL-Eigenschaft ggf. durch geeignete Rotationen wieder her. Zeichnen Sie nur den resultierenden Baum.

(i) Baum 1:

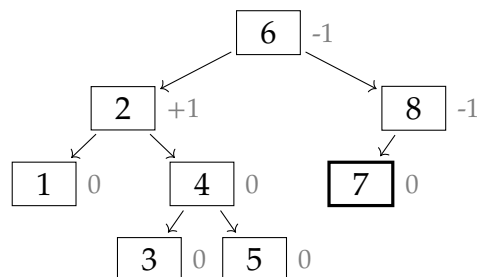


Lösungsvorschlag

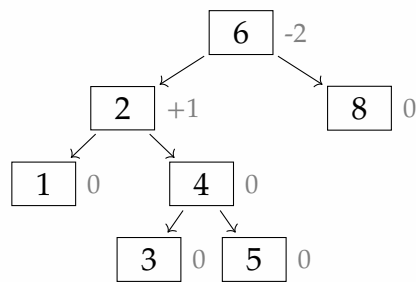
Nach dem Löschen von „4“:



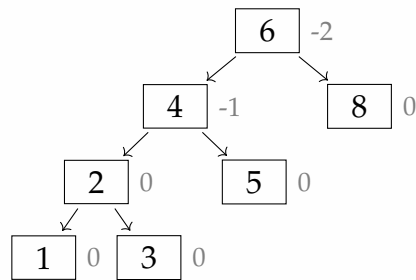
(ii) Baum 2:



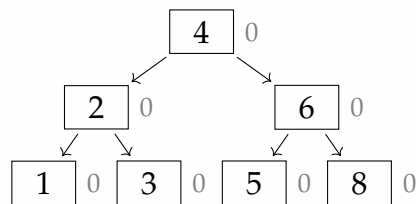
Nach dem Löschen von „7“:



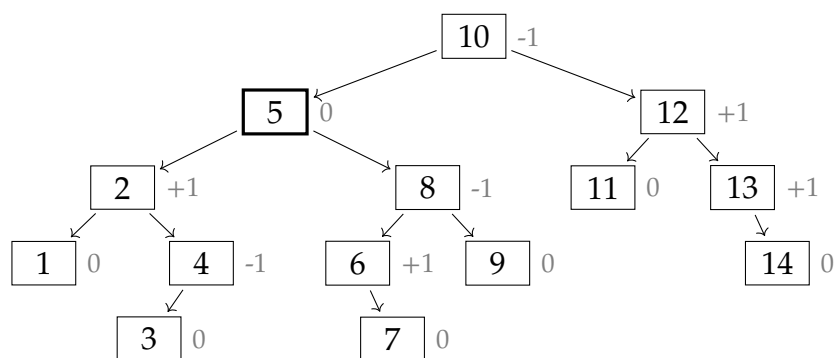
Nach der Linksrotation:



Nach der Rechtsrotation:

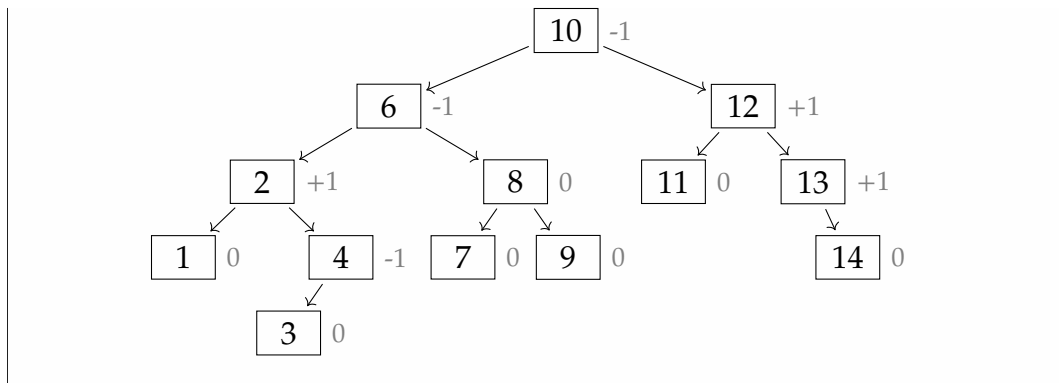


(iii) Baum 3:



Nach dem Löschen von „5“:





### 66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 5

- (a) Definieren Sie die zum Halteproblem für Turing-Maschinen bei fester Eingabe  $m \in \mathbb{N}_0 = \{0, 1, 2, \dots\}$  gehörende Menge  $H_m$ .

Lösungsvorschlag

$H_m = \{c(M) \in \mathbb{N} \mid c(M) \text{ hält auf Eingabe } m\}$ , wobei  $c(M)$  der Codierung der Turingmaschine (Gödelnummer) entspricht.

- (b) Gegeben sei das folgende Problem E:

Entscheiden Sie, ob es für die deterministische Turing-Maschine mit der Gödelnummer  $n$  eine Eingabe  $w \in \mathbb{N}_0$  gibt, so dass  $w$  eine gerade Zahl ist und die Maschine  $n$  gestartet mit  $w$  hält.

Zeigen Sie, dass  $E$  nicht entscheidbar ist. Benutzen Sie, dass  $H_m$  aus (a) für jedes  $m \in \mathbb{N}_0$  nicht entscheidbar ist.

Lösungsvorschlag

Wir zeigen dies durch Reduktion  $H_2 \leq E$ :

- Berechenbare Funktion  $f$ : lösche Eingabe, schreibe eine 2 und starte dich selbst.
- $M$  ist eine Turingmaschine, die  $E$  entscheidet.
- $x \in H_2$  (Quellcode der Programme, die auf die Eingabe von 2 halten)
- $M_x$  (kompiliertes Programm, TM)
- Für alle  $x \in H_2$  gilt,  $M_x$  hält auf Eingabe von 2  $\Leftrightarrow f(x) = c(M) \in E$ . Denn sofern die ursprüngliche Maschine auf das Wort 2 hält, hält  $M$  auf alle Eingaben und somit auch auf Eingaben gerader Zahlen. Hält die ursprüngliche Maschine  $M$  nicht auf die Eingabe der Zahl 2, so hält  $M$  auf keine Eingabe.

- (c) Zeigen Sie, dass das Problem  $E$  aus (b) partiell-entscheidbar (= rekursiv aufzählbar) ist.

**66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 7**

- (a) Sei SAT das Erfüllbarkeitsproblem, und sei  $H_m$ , das Halteproblem bei fester Eingabe  $m$  (siehe Aufgabe 5).

Zeigen Sie: SAT kann in polynomieller Zeit auf  $H_m$  reduziert werden. Als Relation:

$$\text{SAT} \preceq_p H_m$$

- (b) Angenommen, es wurde gezeigt, dass  $P = NP$  ist. Zeigen Sie, dass dann jede Sprache  $L \in P$  über dem Alphabet  $\Sigma$  mit  $L \neq \emptyset$  und  $L \neq \Sigma^*$  sogar NP-vollständig ist.

**66115 / 2014 / Herbst / Thema 2 / Aufgabe 5**

Gegeben sei eine Standarddatenstruktur Stapel (Stack) mit den Operationen

- `void push(Element e)`
- `Element pop()`,
- `boolean isEmpty()`.

sowie dem Standardkonstruktor `Stapel()`, der einen leeren Stapel zur Verfügung stellt.

- (a) Geben Sie eine Methode `Stapel merge(Stapel s, Stapel t)` an, die einen aufsteigend geordneten Stapel zurückgibt, unter der Bedingung, dass die beiden übergebenen Stapel aufsteigend sortiert sind, d.h. `S.pop()` liefert das größte Element in `s` zurück und `T.pop()` liefert das größte Element in `t` zurück. Als Hilfsdatenstruktur dürfen Sie nur Stapel verwenden, keine Felder oder Listen.

Hinweis: Nehmen Sie an, dass Objekte der Klasse `Element`, die auf dem Stapel liegen mit `compareTo()` verglichen werden können. Zum Testen haben wir Ihnen eine Klasse `StapelTest` zur Verfügung gestellt, sie können Ihre Methode hier einfügen und testen, ob die Stapel korrekt sortiert werden. Überlegen Sie auch, was geschieht, wenn einer der Stapel (oder beide) leer ist!

Lösungsvorschlag

```
public static Stapel merge(Stapel s, Stapel t) {
 // Die beiden Stapel unsortiert aneinander hängen.
 Stapel mergedStack = new Stapel();
 while (!s.isEmpty()) {
 mergedStack.push(s.pop());
 }
 while (!t.isEmpty()) {
 mergedStack.push(t.pop());
 }
 // https://www.geeksforgeeks.org/sort-stack-using-temporary-stack/
 Stapel tmpStack = new Stapel();
 while (!mergedStack.isEmpty()) {
 Element tmpElement = mergedStack.pop();
 while (!tmpStack.isEmpty() && tmpStack.top.getValue() >
 tmpElement.getValue()) {
 mergedStack.push(tmpStack.pop());
 }
 mergedStack.push(tmpElement);
 }
}
```

```
 }
 tmpStack.push(tmpElement);
}
return tmpStack;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/herbst/Stapel.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2014/herbst/Stapel.java)

## Komplette Klasse Stapel

```
/**
 * https://www.studon.fau.de/file2860857_download.html
 */
public class Stapel {
 public Element top;

 public Stapel() {
 top = null;
 }

 /**
 * @param element Das Element, dass hinzugefügt werden soll zur Stapel.
 */
 public void push(Element element) {
 element.setNext(top);
 top = element;
 }

 /**
 * @return Das Element oder null, wenn der Stapel leer ist.
 */
 public Element pop() {
 if (top == null) {
 return null;
 }
 Element element = top;
 top = top.getNext();
 return element;
 }

 /**
 * @return Wahr wenn der Stapel leer ist.
 */
 public boolean isEmpty() {
 return top == null;
 }

 /**
 * @param s Stapel s
 * @param t Stapel t
 *
 * @return Ein neuer Stapel.
 */
}
```

```

 */
 public static Stapel merge(Stack s, Stack t) {
 // Die beiden Stapel unsortiert aneinander hängen.
 Stack mergedStack = new Stack();
 while (!s.isEmpty()) {
 mergedStack.push(s.pop());
 }
 while (!t.isEmpty()) {
 mergedStack.push(t.pop());
 }
 // https://www.geeksforgeeks.org/sort-stack-using-temporary-stack/
 Stack tmpStack = new Stack();
 while (!mergedStack.isEmpty()) {
 Element tmpElement = mergedStack.pop();
 while (!tmpStack.isEmpty() && tmpStack.top().getValue() >
 tmpElement.getValue()) {
 mergedStack.push(tmpStack.pop());
 }
 tmpStack.push(tmpElement);
 }
 return tmpStack;
 }

 public static void main(String[] args) {
 Stack sa = new Stack();
 sa.push(new Element(1));
 sa.push(new Element(2));
 sa.push(new Element(4));
 sa.push(new Element(5));
 sa.push(new Element(7));
 sa.push(new Element(8));
 Stack sb = new Stack();
 sb.push(new Element(2));
 sb.push(new Element(3));
 sb.push(new Element(6));
 sb.push(new Element(9));
 sb.push(new Element(10));

 Stack sc = Stack.merge(sa, sb);

 while (!sc.isEmpty()) {
 System.out.print(sc.pop().getValue() + ", ");
 }
 }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/herbst/Stapel.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2014/herbst/Stapel.java)

## Komplette Klasse Element

```

/**
 * https://www.studon.fau.de/file2860856_download.html

```

```
*/
public class Element {
 public int value;

 public Element next;

 public Element() {
 this.next = null;
 }

 public Element(int value, Element element) {
 this.value = value;
 this.next = element;
 }

 public Element(int value) {
 this.value = value;
 this.next = null;
 }

 public int getValue() {
 return value;
 }

 public Element getNext() {
 return next;
 }

 public void setNext(Element element) {
 next = element;
 }

 public int compareTo(Element element) {
 if (getValue() > element.getValue()) {
 return 1;
 } else if (element.getValue() == getValue()) {
 return 0;
 } else {
 return -1;
 }
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/herbst/Element.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2014/herbst/Element.java)

## Test-Klasse

```
import static org.junit.Assert.*;
import org.junit.Test;

/**
```

```
* https://www.studon.fau.de/file2860850_download.html
*/
public class StapelTest {

 @Test
 public void testeMethodenPushPop() {
 Stapel stapel = new Stapel();
 stapel.push(new Element(1));
 stapel.push(new Element(2));
 stapel.push(new Element(3));

 assertEquals(3, stapel.pop().value);
 assertEquals(2, stapel.pop().value);
 assertEquals(1, stapel.pop().value);
 }

 @Test
 public void testeMethodeMerge() {
 Stapel sa = new Stapel();
 sa.push(new Element(1));
 sa.push(new Element(3));
 sa.push(new Element(5));

 Stapel sb = new Stapel();
 sb.push(new Element(2));
 sb.push(new Element(4));

 Stapel sc = Stapel.merge(sa, sb);

 assertEquals(5, sc.pop().getValue());
 assertEquals(4, sc.pop().getValue());
 assertEquals(3, sc.pop().getValue());
 assertEquals(2, sc.pop().getValue());
 assertEquals(1, sc.pop().getValue());
 }

 @Test
 public void testeMethodeMergeMehrWerte() {
 Stapel sa = new Stapel();
 sa.push(new Element(1));
 sa.push(new Element(2));
 sa.push(new Element(4));
 sa.push(new Element(5));
 sa.push(new Element(7));
 sa.push(new Element(8));
 Stapel sb = new Stapel();
 sb.push(new Element(2));
 sb.push(new Element(3));
 sb.push(new Element(6));
 sb.push(new Element(9));
 sb.push(new Element(10));

 Stapel sc = Stapel.merge(sa, sb);
```

```
 assertEquals(10, sc.pop().getValue());
 assertEquals(9, sc.pop().getValue());
 assertEquals(8, sc.pop().getValue());
 assertEquals(7, sc.pop().getValue());
 assertEquals(6, sc.pop().getValue());
 assertEquals(5, sc.pop().getValue());
 assertEquals(4, sc.pop().getValue());
 assertEquals(3, sc.pop().getValue());
 assertEquals(2, sc.pop().getValue());
 assertEquals(2, sc.pop().getValue());
 assertEquals(1, sc.pop().getValue());
}

@Test
public void testeMethodeMergeBLEer() {
 Stapel sa = new Stapel();
 sa.push(new Element(1));
 sa.push(new Element(3));
 sa.push(new Element(5));

 Stapel sb = new Stapel();

 Stapel sc = Stapel.merge(sa, sb);

 assertEquals(5, sc.pop().getValue());
 assertEquals(3, sc.pop().getValue());
 assertEquals(1, sc.pop().getValue());
}

@Test
public void testeMethodeMergeALEer() {
 Stapel sa = new Stapel();

 Stapel sb = new Stapel();
 sb.push(new Element(2));
 sb.push(new Element(4));

 Stapel sc = Stapel.merge(sa, sb);

 assertEquals(4, sc.pop().getValue());
 assertEquals(2, sc.pop().getValue());
}
}
```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/herbst/StapelTest.java](https://github.com/src/test/java/org/bschlangaul/examen/examen_66115/jahr_2014/herbst/StapelTest.java)

(b) Analysieren Sie die Laufzeit Ihrer Methode.

Lösungsvorschlag

Best case:  $\mathcal{O}(1)$  Worst case:  $\mathcal{O}(n^2)$

**66115 / 2014 / Herbst / Thema 2 / Aufgabe 6**

Gegeben sei ein einfacher Sortieralgorithmus, der ein gegebenes Feld  $A$  dadurch sortiert, dass er das *Minimum*  $m$  von  $A$  findet, dann das Minimum von  $A$  ohne das Element  $m$  usw.

- (a) Geben Sie den Algorithmus in Java an. Implementieren Sie den Algorithmus *in situ*, dso, dass er außer dem Eingabefeld nur konstanten Extraspeicher benötigt. Es steht eine Testklasse zur Verfügung.

Lösungsvorschlag

```
public class SortierungDurchAuswaehlen {
 static void vertausche(int[] zahlen, int index1, int index2) {
 int tmp = zahlen[index1];
 zahlen[index1] = zahlen[index2];
 zahlen[index2] = tmp;
 }

 static void sortiereDurchAuswählen(int[] zahlen) {
 // Am Anfang ist die Markierung das erste Element im Zahlen-Array.
 int markierung = 0;
 while (markierung < zahlen.length) {
 // Bestimme das kleinste Element.
 // 'min' ist der Index des kleinsten Elements.
 // Am Anfang auf das letzte Element setzen.
 int min = zahlen.length - 1;
 // Wir müssen nicht bis letzten Index gehen, da wir 'min' auf das
 // ↳ letzte Element
 // setzen.
 for (int i = markierung; i < zahlen.length - 1; i++) {
 if (zahlen[i] < zahlen[min]) {
 min = i;
 }
 }

 // Tausche zahlen[markierung] mit gefundenem Element.
 vertausche(zahlen, markierung, min);
 // Die Markierung um eins nach hinten verlegen.
 markierung++;
 }
 }

 public static void main(String[] args) {
 int[] zahlen = { 5, 2, 7, 1, 6, 3, 4 };
 sortiereDurchAuswählen(zahlen);
 for (int i = 0; i < zahlen.length; i++) {
 System.out.print(zahlen[i] + " ");
 }
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/herbst/SortierungDurchAuswaehlen.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2014/herbst/SortierungDurchAuswaehlen.java)

- (b) Analysieren Sie die Laufzeit Ihres Algorithmus.



Beim ersten Durchlauf des *Selectionsort*-Algorithmus muss  $n - 1$  mal das Minimum durch Vergleich ermittelt werden, beim zweiten Mal  $n - 2$ . Mit Hilfe der *Gaußschen Summenformel* kann die Komplexität gerechnet werden:

$$(n - 1) + (n - 2) + \dots + 3 + 2 + 1 = \frac{(n - 1) \cdot n}{2} = \frac{n^2}{2} - \frac{n}{2} \approx \frac{n^2}{2} \approx n^2$$

Da es bei der Berechnung der Komplexität um die Berechnung der asymptotischen oberen Grenze geht, können Konstanten und die Addition, Subtraktion, Multiplikation und Division mit Konstanten z. B.  $\frac{n^2}{2}$  vernachlässigt werden.

Der *Selectionsort*-Algorithmus hat deshalb die Komplexität  $\mathcal{O}(n^2)$ , er ist von der Ordnung  $\mathcal{O}(n^2)$ .

- (c) Geben Sie eine Datenstruktur an, mit der Sie Ihren Algorithmus beschleunigen können.

Der *Selectionsort*-Algorithmus kann mit einer Min- (in diesem Fall) bzw. einer Max-Heap beschleunigt werden. Mit Hilfe dieser Datenstruktur kann sehr schnell das Minimum gefunden werden. So kann auf die vielen Vergleiche verzichtet werden. Die Komplexität ist dann  $\mathcal{O}(n \log n)$ .

## 66115 / 2015 / Frühjahr / Thema 1 / Aufgabe 1

Die Sprache  $L$  über den Alphabet  $\Sigma = \{0, 1\}$  enthält alle Wörter, bei denen beim Lesen von links nach rechts der Unterschied in der Zahl der 0en und 1en stets höchstens 3 ist. Also ist  $w \in L$  genau dann, wenn für alle  $u, v$  mit  $w = uv$  gilt  $||u|_0 - |u|_1| \leq 3$ . Erinnerung:  $|w|_a$  bezeichnet die Zahl der  $a$ 's im Wort  $w$ .

- (a) Sei  $A = (Q, \Sigma, \delta, q_0, E)$  ein deterministischer endlicher Automat für  $L$ . Es sei  $w_1 = 111, w_2 = 11, w_3 = 1, w_4 = \varepsilon, w_5 = 0, w_6 = 00, w_7 = 000$ . Machen Sie sich klar, dass der Automat jedes dieser Wörter verarbeiten können muss. Folgern Sie, dass der Automat mindestens sieben Zustände haben muss. Schreiben Sie Ihr Argumentation schlüssig und vollständig auf.

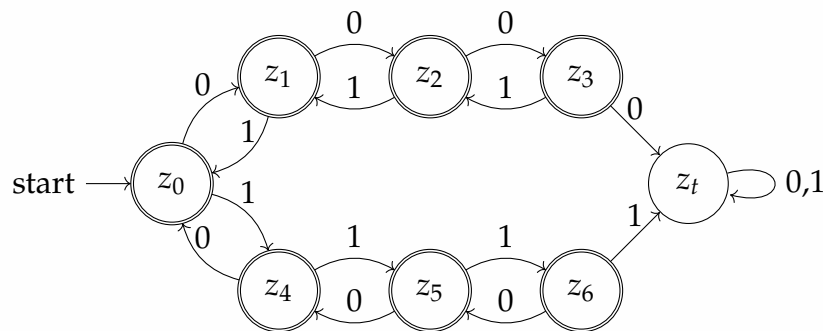
Ein deterministischer endlicher Automat hat keinen zusätzlichen Speicher zur Verfügung, in dem die Anzahl der bisher vorkommenden Einsen und Nullen gespeichert werden könnte. Ein deterministischer endlicher Automat kann die von der Sprache benötigten Anzahl an Einsen und Nullen nur in Form von Zuständen speichern. Um die Anzahl von 3 Einsen bzw. 3 Nullen zu speichern, sind also 6 Zustände nötig.

Die Wörter 01 oder 0011 oder 0101 etc. haben eine Differenz von 0, wenn die Anzahl an Nullen und Einsen abgezogen wird. Um auch diese Wörter darstellen

zu können, ist mindestens ein weiterer Zustand nötig.

- (b) Begründen Sie, dass  $L$  regulär ist.

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ait6va31c

- (c) Jemand behauptet, diese Sprache sei nicht regulär und gibt folgenden „Beweis“ dafür an: Wäre  $L$  regulär, so sei  $n$  eine entsprechende Pumping-Zahl. Nun ist  $w = (01)^n \in L$ . Zerlegt man nun  $w = uxv$ , wobei  $u = 0, x = 1, v = (01)^{n-1}$ , so ist zum Beispiel  $ux^5v \notin L$ , denn es ist  $ux^5v = 01111101010101\dots$ . Legen Sie genau dar, an welcher Stelle dieser „Beweis“ fehlerhaft ist.

#### Exkurs: Pumping-Lemma für Reguläre Sprachen

Es sei  $L$  eine reguläre Sprache. Dann gibt es eine Zahl  $j$ , sodass für alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  (jedes Wort  $\omega$  in  $L$  mit Mindestlänge  $j$ ) jeweils eine Zerlegung  $\omega = uvw$  existiert, sodass die folgenden Eigenschaften erfüllt sind:

- (i)  $|v| \geq 1$  (Das Wort  $v$  ist nicht leer.)
- (ii)  $|uv| \leq j$  (Die beiden Wörter  $u$  und  $v$  haben zusammen höchstens die Länge  $j$ .)
- (iii) Für alle  $i = 0, 1, 2, \dots$  gilt  $uv^i w \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^i w$  in der Sprache  $L$ )

Die kleinste Zahl  $j$ , die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache  $L$  genannt.

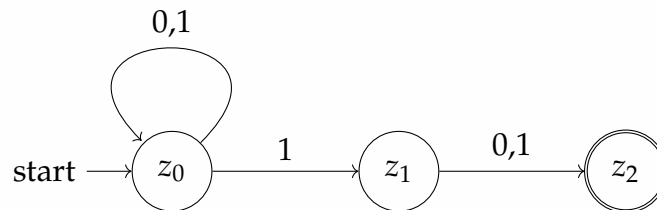
Lösungsvorschlag

Das Wort  $(01)^n$  wurde falsch zerlegt. Für die Pumping-Zahl  $n = 3$  gibt es sehr wohl eine Zerlegung, die beim Aufpumpen regulär ist, also:  $\omega = 010101$  ( $u = 01, x = 01$  und  $v = 01$ ).  $ux^5v = 010101010101 \in L$ . Es gibt also eine Zerlegung, die beim Aufpumpen die 3 Pumping-Lemma-Eigenschaften erfüllt. Daher kann man das Pumping-Lemma so nicht widerlegt werden, indem man ein einziges Gegenbeispiel gibt.

- (d) In anderen Fällen können nichtdeterministische endliche Automaten echt kleiner sein als die besten deterministischen Automaten. Ein Beispiel ist die Sprache

$L_2 = \Sigma^*1\Sigma$  aller Wörter, deren vorletztes Symbol 1 ist. Geben Sie einen nicht-deterministischen Automaten mit nur drei Zuständen an,  $L_2$  erkennt.

Lösungsvorschlag



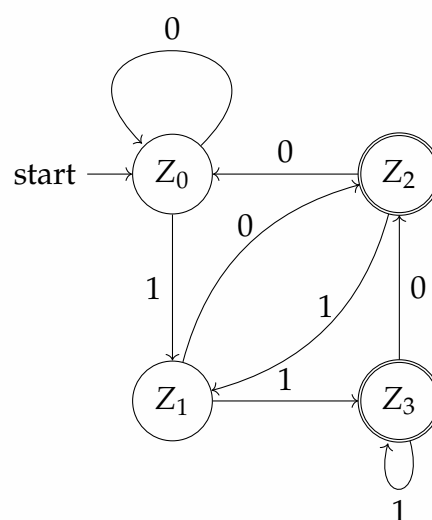
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apwezjufbg

- (e) Führen Sie auf Ihrem Automaten die Potenzmengenkonstruktion und anschließend den Minimierungsalgorithmus durch. Wie viele Zustände muss ein deterministischer Automat für  $L_2$  also mindestens haben?

Lösungsvorschlag

### Potenzmengenkonstruktion

| Name  | Zustandsmenge           | Eingabe 0          | Eingabe 1               |
|-------|-------------------------|--------------------|-------------------------|
| $Z_0$ | $Z_0 \{z_0\}$           | $Z_0 \{z_0\}$      | $Z_1 \{z_0, z_1\}$      |
| $Z_1$ | $Z_1 \{z_0, z_1\}$      | $Z_2 \{z_0, z_2\}$ | $Z_3 \{z_0, z_1, z_2\}$ |
| $Z_2$ | $Z_2 \{z_0, z_2\}$      | $Z_0 \{z_0\}$      | $Z_1 \{z_0, z_1\}$      |
| $Z_3$ | $Z_3 \{z_0, z_1, z_2\}$ | $Z_2 \{z_0, z_2\}$ | $Z_3 \{z_0, z_1, z_2\}$ |



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajfcf9pb9

**Minimierungsalgorithmus**

|       |             |             |             |             |
|-------|-------------|-------------|-------------|-------------|
| $Z_0$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $Z_1$ | $x_2$       | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $Z_2$ | $x_1$       | $x_1$       | $\emptyset$ | $\emptyset$ |
| $Z_3$ | $x_1$       | $x_1$       | $x_2$       | $\emptyset$ |
|       | $Z_0$       | $Z_1$       | $Z_2$       | $Z_3$       |

$x_1$  Paar aus End- / Nicht-Endzustand kann nicht äquivalent sein.

$x_2$  Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

$x_3$  In weiteren Iterationen markierte Zustände.

$x_4$  ...

**Übergangstabelle**

| Zustandspaar | 0                      | 1                      |
|--------------|------------------------|------------------------|
| $(Z_0, Z_1)$ | $(Z_0, Z_2) \quad x_2$ | $(Z_1, Z_3) \quad x_2$ |
| $(Z_2, Z_3)$ | $(Z_0, Z_1)$           | $(Z_1, Z_3) \quad x_2$ |

Wie aus der oben stehenden Tabelle abzulesen ist, gibt es keine äquivalenten Zustände. Der Automat kann nicht minimiert werden. Er ist bereits minimal.

**66115 / 2015 / Frühjahr / Thema 2 / Aufgabe 4**

Sei  $M_0, M_1, \dots$  eine Gödelisierung aller Registermaschinen (RAMs). Geben Sie für die folgenden Mengen  $D_1, D_2, D_3$  an, ob sie entscheidbar oder aufzählbar sind. Begründen Sie Ihre Behauptungen, wobei Sie die Aufzählbarkeit und Unentscheidbarkeit des speziellen Halteproblems  $K_0 = \{x \in \mathbb{N} \mid M_x \text{ h\ae}lt \text{ bei Eingabe } x\}$  verwenden dürfen.  $D_1 = \{x \in \mathbb{N} \mid x < 9973 \text{ und } M_x \text{ h\ae}lt \text{ bei Eingabe } x\}$   $D_2 = \{x \in \mathbb{N} \mid x \geq 9973 \text{ und } M_x \text{ h\ae}lt \text{ bei Eingabe } x\}$   $D_3 = \{x \in \mathbb{N} \mid M_x \text{ h\ae}lt \text{ nicht bei Eingabe } x\}$

$D_1$  ist eine endliche Menge und damit entscheidbar. Auch eine endliche Teilmenge des Halteproblems. Anschaulich kann man sich dies so vorstellen: Man stellt dem Rechner eine Liste zur Verfügung, die alle haltenden Maschinen  $M_x$  mit  $x < 9973$  enthält. Diese Liste kann zum Beispiel vorab von einem Menschen erstellt worden sein, denn die Menge der zu prüfenden Programme ist endlich.

$D_2$   $x \geq 9973$  entscheidbar,  $L$  halt semi-entscheidbar  $\rightarrow$  semi-entscheidbar (Hier wäre auch eine Argumentation über die Cantorsche Paarungsfunktion möglich). Es ist weiterhin nicht entscheidbar. Dazu betrachten wir die Reduktion des speziellen Halteproblems  $H_0 : H_0 \leq D_2$  Für alle  $x < 9973$  lassen wir  $M_x$  durch eine

Turingmaschine  $M_y$  simulieren, die eine höhere Nummer hat.

$D_3$  ist unentscheidbar, denn angenommen  $D_3$  wäre semi-entscheidbar, dann würde sofort folgen, dass  $L$  halt entscheidbar ist, da aus der Semientscheidbarkeit von  $L$  halt und  $L$  halt die Entscheidbarkeit von  $L$  halt folgen würde

## 66115 / 2015 / Frühjahr / Thema 2 / Aufgabe 5

Gegeben seien die Standardstrukturen Stapel (Stack) und Schlange (Queue) mit folgenden Standardoperationen:

| Stapel                         | Schlange                       |
|--------------------------------|--------------------------------|
| <code>boolean isEmpty()</code> | <code>boolean isEmpty()</code> |
| <code>void push(int e)</code>  | <code>enqueue(int e)</code>    |
| <code>int pop()</code>         | <code>int dequeue()</code>     |
| <code>int top()</code>         | <code>int head()</code>        |

Beim Stapel gibt die Operation `top()` das gleiche Element wie `pop()` zurück, bei der Schlange gibt `head()` das gleiche Element wie `dequeue()` zurück. Im Unterschied zu `pop()`, beziehungsweise `dequeue()`, wird das Element bei `top()` und `head()` nicht aus der Datenstruktur entfernt.

- (a) Geben Sie in Pseudocode einen Algorithmus `sort(Stack s)` an, der als Eingabe einen Stapel `s` mit `n` Zahlen erhält und die Zahlen in `s` sortiert. (Sie dürfen die Zahlen wahlweise entweder aufsteigend oder absteigend sortieren.) Verwenden Sie als Hilfsdatenstruktur ausschließlich eine Schlange `q`. Sie erhalten volle Punktzahl, wenn Sie außer `s` und `q` keine weiteren Variablen benutzen. Sie dürfen annehmen, dass alle Zahlen in `s` verschieden sind.

Lösungsvorschlag

```
q := neue Schlange
while s not empty:
 q.enqueue(S.pop())
while q not empty:
 while s not empty and s.top() < q.head():
 q.enqueue(s.pop())
 s.push(q.dequeue())
```

### Als Java-Code

```
/**
 * So ähnlich wie der <a href=
 * "https://www.geeksforgeeks.org/sort-stack-using-temporary-
 * stack/">Stapel-Sortiert-Algorithmus
 * der nur Stapel verwendet, nur mit einer Warteschlange.
 *
 * @param s Der Stapel, der sortiert wird.
 */
public static void sort(Stack s) {
```

```
Schlange q = new Schlange();
while (!s.isEmpty()) {
 q.enqueue(s.pop());
}
while (!q.isEmpty()) {
 // Sortiert aufsteigend. Für absteigend einfach das „kleiner“
 // Zeichen umdrehen.
 while (!s.isEmpty() && s.top() < q.head()) {
 q.enqueue(s.pop());
 }
 s.push(q.dequeue());
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2015/fruehjahr/schlange/Sort.java](src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Sort.java)

## Klasse Sort

```
public class Sort {

 /**
 * So ähnlich wie der <a href=
 * "https://www.geeksforgeeks.org/sort-stack-using-temporary-
 → stack/">Stapel-Sortiert-Algorithmus
 * der nur Stapel verwendet, nur mit einer Warteschlange.
 *
 * @param s Der Stapel, der sortiert wird.
 */
 public static void sort(Stack s) {
 Schlange q = new Schlange();
 while (!s.isEmpty()) {
 q.enqueue(s.pop());
 }
 while (!q.isEmpty()) {
 // Sortiert aufsteigend. Für absteigend einfach das „kleiner“
 // Zeichen umdrehen.
 while (!s.isEmpty() && s.top() < q.head()) {
 q.enqueue(s.pop());
 }
 s.push(q.dequeue());
 }
 }

 public static Stack stapelBefüllen(int[] zahlen) {
 Stack s = new Stack();
 for (int i : zahlen) {
 s.push(i);
 }
 return s;
 }

 public static void zeigeStapel(Stack s) {
```

```
 while (!s.isEmpty()) {
 System.out.print(s.pop() + " ");
 }
 System.out.println();
 }

 public static void main(String[] args) {
 Stapel s1 = stapelBefüllen(new int[] { 4, 2, 1, 5, 3 });
 sort(s1);
 zeigeStapel(s1);

 Stapel s2 = stapelBefüllen(new int[] { 1, 2, 6, 3, 9, 11, 4 });
 sort(s2);
 zeigeStapel(s2);
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2015/fruehjahr/schlange/Sort.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Sort.java)

## Klasse Schlange

```
public class Schlange {

 public Element head;

 public Schlange() {
 head = null;
 }

 public int head() {
 if (head.getNext() == null) {
 return head.getValue();
 }
 Element element = head;
 Element previous = head;
 while (element.getNext() != null) {
 previous = element;
 element = element.getNext();
 }
 element = previous.getNext();
 return element.getValue();
 }

 /**
 * @param value Eine Zahl, die zur Schlange hinzugefügt werden soll.
 */
 public void enqueue(int value) {
 Element element = new Element(value);
 element.setNext(head);
 head = element;
 }
}
```



```
/**
 * @return Das Element oder null, wenn der Schlange leer ist.
 */
public int dequeue() {
 if (head.getNext() == null) {
 int result = head.getValue();
 head = null;
 return result;
 }
 Element element = head;
 Element previous = null;
 while (element.getNext() != null) {
 previous = element;
 element = element.getNext();
 }
 element = previous.getNext();
 previous.setNext(null);
 return element.getValue();
}

/**
 * @return Wahr wenn der Schlange leer ist.
 */
public boolean isEmpty() {
 return head == null;
}

public static void main(String[] args) {
 Schlange s = new Schlange();
 s.enqueue(1);
 s.enqueue(2);
 s.enqueue(3);
 System.out.println(s.head());
 System.out.println(s.dequeue());
 System.out.println(s.head());
 System.out.println(s.dequeue());
 System.out.println(s.head());
 System.out.println(s.dequeue());
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2015/fruehjahr/schlange/Schlange.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Schlange.java)

## Klasse Element

```
public class Element {
 public int value;

 public Element next;

 public Element() {
```

```
 this.next = null;
 }

 public Element(int value, Element element) {
 this.value = value;
 this.next = element;
 }

 public Element(int value) {
 this.value = value;
 this.next = null;
 }

 public int getValue() {
 return value;
 }

 public Element getNext() {
 return next;
 }

 public void setNext(Element element) {
 next = element;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2015/fruehjahr/schlange/Element.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/Element.java)

## Test-Klasse

```
import static org.junit.Assert.*;

import org.junit.Test;

public class TestCase {

 @Test
 public void testeStapel() {
 Stapel s = new Stapel();
 s.push(1);
 s.push(2);
 s.push(3);

 assertEquals(false, s.isEmpty());

 assertEquals(3, s.top());
 assertEquals(3, s.pop());

 assertEquals(2, s.top());
 assertEquals(2, s.pop());
 }
}
```

```

 assertEquals(1, s.top());
 assertEquals(1, s.pop());
 assertEquals(true, s.isEmpty());
}

@Test
public void testeSchlange() {
 Schlange s = new Schlange();
 s.enqueue(1);
 s.enqueue(2);
 s.enqueue(3);

 assertEquals(false, s.isEmpty());

 assertEquals(1, s.head());
 assertEquals(1, s.dequeue());

 assertEquals(2, s.head());
 assertEquals(2, s.dequeue());

 assertEquals(3, s.head());
 assertEquals(3, s.dequeue());
 assertEquals(true, s.isEmpty());
}
}

```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2015/fruehjahr/schlange/TestCase.java](https://github.com/src/test/java/org/bschlangaul/examen/examen_66115/jahr_2015/fruehjahr/schlange/TestCase.java)

(b) Analysieren Sie die Laufzeit Ihrer Methode in Abhängigkeit von  $n$ .

Lösungsvorschlag

Zeitkomplexität:  $\mathcal{O}(n^2)$ , da es zwei ineinander verschachtelte **while**-Schleifen gibt, die von der Anzahl der Elemente im Stapel abhängen.

## 66115 / 2015 / Frühjahr / Thema 2 / Aufgabe 7

Auf folgendem ungerichteten, gewichteten Graphen wurde der Dijkstra-Algorithmus (wie auf der nächsten Seite beschrieben) ausgeführt, doch wir wissen lediglich, welcher Knoten als letztes schwarz (black) wurde (Nr. 8) und was seine Distanz zum Startknoten (Nr. 1) ist. Die Gewichte der Kanten sind angegeben.

Finden Sie zunächst den Startknoten, nummerieren Sie anschließend die Knoten in der Reihenfolge, in der sie schwarz wurden, und geben Sie in jedem Knoten die Distanz zum Startknoten an.

Hinweis: Der Startknoten ist eindeutig.

Dijkstra(WeightedGraph G, Vertex s)

```

Initialize(G, s);
S=∅;
Q = new PriorityQueue(V, d) ;
while not Q.Empty() do

```

```

u = Q.ExtractMin() ;
S = S U {u};
foreach v ∈ Adj[u] do
 Relax(u, v; w);

u.color = black;

```

Initialize(Graph G, Vertex s)

```

foreach u ∈ V do
 u.color = white;
 u.d = 00;
s.color = gray;
s.d = 0;

```

Relax(u, v; w)

```

if v.d > u.d + w(u,v) then
 v.color = gray;
 v.d = u.d + w(u,v);
 Q.DecreaseKey(v, v.d);

```

## 66115 / 2016 / Frühjahr / Thema 1 / Aufgabe 1

- (a) Geben Sie einen möglichst einfachen regulären Ausdruck für die Sprache  $L_1 = \{a_1 a_2 \dots a_n \mid n \geq 3, a_i \in \{a, b\} \text{ für alle } i = 1, \dots, n \text{ und } a_1 \geq a_n\}$  an.

Lösungsvorschlag

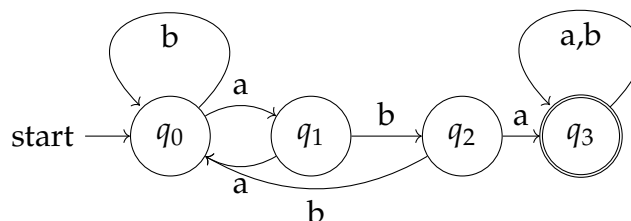
$((a(a|b)+b) | (b(a|b)+a))$

- (b) Geben Sie einen möglichst einfachen regulären Ausdruck für die Sprache  $L_2 = \{w \in \{a, b\}^* \mid w \text{ enthält genau ein } b \text{ und ist von ungerader Länge}\}$  an.

Lösungsvorschlag

$(aa)^*(b|aba)(aa)^*$

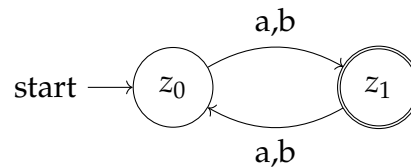
- (c) Beschreiben Sie die Sprache des folgenden Automaten  $A_1$ , möglichst einfach und präzise in ihren eigenen Worten.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Arz003ccg](http://flaci.com/Arz003ccg)

Die Sprache enthält das Teilwort *aba*

(d) Betrachten Sie folgenden Automaten  $A_2$ :



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ap9qbkumc](http://flaci.com/Ap9qbkumc)

Im Original sind die Zustände mit  $q_x$  benannt. Damit wir die Schnittmenge besser bilden können, wird hier  $z_x$  verwendet.

Konstruieren Sie einen endlichen Automaten, der die Schnittmenge der Sprachen  $L(A_1)$  und  $L(A_2)$  akzeptiert.

$A_1$

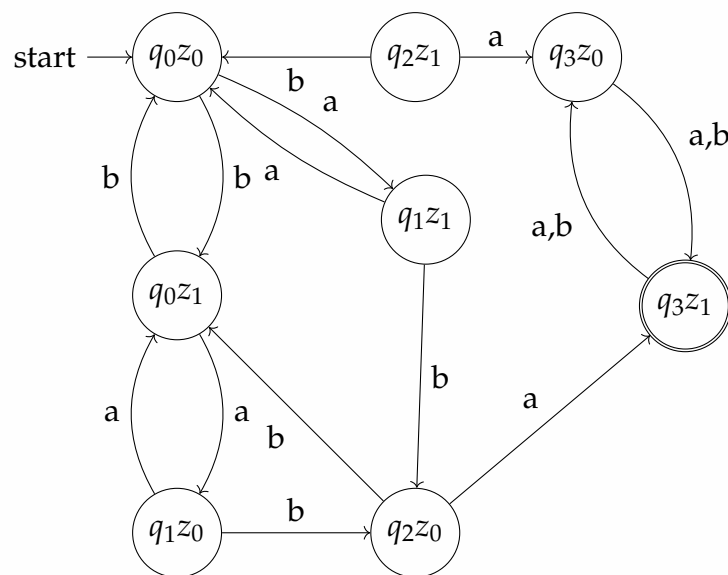
|       | a     | b     |
|-------|-------|-------|
| $q_0$ | $q_1$ | $q_0$ |
| $q_1$ | $q_0$ | $q_2$ |
| $q_2$ | $q_3$ | $q_0$ |
| $q_3$ | $q_3$ | $q_3$ |

$A_2$

|       | a     | b     |
|-------|-------|-------|
| $z_0$ | $z_1$ | $z_1$ |
| $z_1$ | $z_0$ | $z_0$ |

Neuer Endzustand:  $q_3z_1$

|          | a        | b        |
|----------|----------|----------|
| $q_0z_0$ | $q_1z_1$ | $q_0z_1$ |
| $q_1z_0$ | $q_0z_1$ | $q_2z_1$ |
| $q_2z_0$ | $q_3z_1$ | $q_0z_1$ |
| $q_3z_0$ | $q_3z_1$ | $q_3z_1$ |
| $q_0z_1$ | $q_1z_0$ | $q_0z_0$ |
| $q_1z_1$ | $q_0z_0$ | $q_2z_0$ |
| $q_2z_1$ | $q_3z_0$ | $q_0z_0$ |
| $q_3z_1$ | $q_3z_0$ | $q_3z_0$ |



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ar3pc5rh7](http://flaci.com/Ar3pc5rh7)

## 66115 / 2016 / Frühjahr / Thema 1 / Aufgabe 2

Betrachten Sie die folgende Grammatik  $G = (\{S, A\}, \{0, 1, 2\}, P, S)$  mit

$P = \{$

$$S \rightarrow 0S0 \mid 1S1 \mid 2A2 \mid 0 \mid 1 \mid \varepsilon$$

$$A \rightarrow A2$$

$\}$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Gf6scqja9](http://flaci.com/Gf6scqja9)

Konstruieren Sie für die Grammatik  $G$  schrittweise eine äquivalente Grammatik in Chomsky-Normalform. Geben Sie für jeden einzelnen Schritt des Verfahrens das vollständige Zwischenergebnis an und erklären Sie kurz, was in dem Schritt getan wurde.

Lösungsvorschlag

Die Regeln  $\{S \rightarrow 2A2\}$  und  $\{A \rightarrow A2\}$  können gelöscht werden, da es keine Regel  $\{A \rightarrow \varepsilon\}$  oder  $\{A \rightarrow S\}$  gibt. So erhalten wir:

$$P = \left\{ \begin{array}{l} S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon \end{array} \right\}$$

**(a) Elimination der  $\varepsilon$ -Regeln**

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen. —

falls  $S \rightarrow \varepsilon \in P$  neuen Startzustand  $S_1$  einführen

$$P = \left\{ \begin{array}{l} S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid 00 \mid 11 \\ S_1 \rightarrow \varepsilon \mid S \end{array} \right\}$$

**(b) Elimination von Kettenregeln**

— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren. —

$\emptyset$  Nichts zu tun

**(c) Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt. —

$N = \text{Null}$   $E = \text{Eins}$

$$P = \left\{ \begin{array}{l} S \rightarrow NSN \mid ESE \mid 0 \mid 1 \mid NN \mid EE \\ S_1 \rightarrow \varepsilon \mid S \\ A \rightarrow AZ \\ N \rightarrow 0 \\ E \rightarrow 1 \end{array} \right\}$$

}

**(d) Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1 B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1} B_n, A_{n-1} \rightarrow A_{n-2} B_{n-1}, \dots, A_2 \rightarrow B_1 B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. —

$$P = \left\{ \right.$$

$$S \rightarrow NS_N \mid ES_E \mid 0 \mid 1 \mid NN \mid EE$$

$$S_1 \rightarrow \varepsilon \mid S$$

$$S_N \rightarrow SN$$

$$S_E \rightarrow SE$$

$$N \rightarrow 0$$

$$E \rightarrow 1$$

}

**66115 / 2016 / Frühjahr / Thema 1 / Aufgabe 4**

(a) Geben Sie eine deterministische 2-Band Turingmaschine  $M$  an, die die Funktion

$$f_M(a^n) = a^n b^n$$

berechnet. Die Maschine  $M$  nimmt somit immer einen String der Form  $a^n$  (ein String, der aus  $n$   $a$ 's für beliebiges  $n \in \mathbb{N}$  besteht) als Eingabe und produziert anschließend auf Band 2 als Ausgabe den String  $a^n b^n$  (ein String aus  $n$   $a$ 's gefolgt von  $n$   $b$ 's).

Beschreiben Sie außerdem die Idee hinter Ihrer Konstruktion.

Lösungsvorschlag

```

name: 66115 2016 03 1 4
init: z0
accept: z2

z0, a, _
z0, a, a, >, >

z0, _, _
z1, _, _, <, -

z1, a, _
z1, a, _, <, -

z1, _, _
z2, _, _, >, -

```



z2, a, \_  
z2, a, b, >, >

a

|    |                                                              |
|----|--------------------------------------------------------------|
| z0 | a's auf das 2. Band kopieren                                 |
| z1 | Zu Beginn der Eingabe auf dem 1. Band, 2. Band bleibt        |
| z2 | Für jedes a's auf dem 1. Band ein b auf dem 2. Band erzeugen |

<sup>a</sup><http://turingmachinesimulator.com/shared/lyptczerhe>

- (b) Geben Sie die Konfigurationsfolge der Turingmaschine aus (a) für die Eingabe *aa* an.

Lösungsvorschlag

z0 a a, z0 □□  
a z0 a, a z0 □  
a a z0 □, a a z0 □  
a z1 a, a a z1 □  
z1 a a □, a a z1 □  
z1 □a a, a a z1 □  
z2 a a, a a z2 □  
a z2 a, a a b z2 □  
a a z2 □, a a b b z2 □

## 66115 / 2016 / Frühjahr / Thema 1 / Aufgabe 5

Das Problem k-COL ist wie folgt definiert:

**k-COL**

**Gegeben:** Ein ungerichteter Graph  $G = (V, E)$ .

**Frage:** Kann man jedem Knoten  $v$  in  $V$  eine Zahl  $z(v) \in \{1, \dots, k\}$  zuordnen, so dass für alle Kanten  $(u_1, u_2) \in E$  gilt:  $z(u_1) \neq z(u_2)$ ?

Zeigen Sie, dass man 3-COL in polynomieller Zeit auf 4-COL reduzieren kann. Beschreiben Sie dazu die Reduktion und zeigen Sie anschließend ihre Korrektheit.

Lösungsvorschlag

Zu Zeigen:

$$3\text{-COL} \preceq_p 4\text{-COL}$$

also 4-COL ist mindestens so schwer wie 3-COL Eingabeinstanz von 3-COL durch

eine Funktion in eine Eingabeinstanz von 4-CoL umbauen so, dass jede JA- bzw. NEIN-Instanz von 3-CoL eine JA- bzw. NEIN-Instanz von 4-CoL ist.

Funktion ergänzt einen beliebigen gegebenen Graphen um einen weiteren Knoten, der mit allen Knoten des ursprünglichen Graphen durch eine Kante verbunden ist.

**total** ja

**in Polynomialzeit berechenbar** ja (Begründung: z. B. Adjazenzmatrix  $\rightarrow$  neue Spalte)

**Korrektheit:** ja

Färbe den „neuen“ Knoten mit einer Farbe. Da er mit allen anderen Knoten verbunden ist, bleiben für die übrigen Knoten nur drei Farben.

## 66115 / 2016 / Frühjahr / Thema 1 / Aufgabe 6

Sortieren Sie die Werte

1 45 8 53 9 2 17 10

mit Quicksort.

Lösungsvorschlag

Sortieralgorithmus nach Saake

```

1 45 8 53 9 2 17 10 zerlege
1 45 8 53* 9 2 17 10 markiere (i 3)
1 45 8 >53 9 2 17 10< vertausche (i 3<>7)
>1< 45 8 10 9 2 17 53 vertausche (i 0<>0)
1 >45< 8 10 9 2 17 53 vertausche (i 1<>1)
1 45 >8< 10 9 2 17 53 vertausche (i 2<>2)
1 45 8 >10< 9 2 17 53 vertausche (i 3<>3)
1 45 8 10 >9< 2 17 53 vertausche (i 4<>4)
1 45 8 10 9 >2< 17 53 vertausche (i 5<>5)
1 45 8 10 9 2 >17< 53 vertausche (i 6<>6)
1 45 8 10 9 2 17 >53< vertausche (i 7<>7)
1 45 8 10 9 2 17 zerlege
1 45 8 10* 9 2 17 markiere (i 3)
1 45 8 >10 9 2 17< vertausche (i 3<>6)
>1< 45 8 17 9 2 10 vertausche (i 0<>0)
1 >45 8< 17 9 2 10 vertausche (i 1<>2)
1 8 >45 17 9< 2 10 vertausche (i 2<>4)
1 8 9 >17 45 2< 10 vertausche (i 3<>5)
1 8 9 2 >45 17 10< vertausche (i 4<>6)
1 8 9 2 zerlege
1 8* 9 2 markiere (i 1)
1 >8 9 2< vertausche (i 1<>3)

```

```

>1< 2 9 8 vertausche (i 0<>0)
1 >2< 9 8 vertausche (i 1<>1)
1 2 >9 8< vertausche (i 2<>3)
1 2
1* 2
>1 2<
>2 1<
 17 45 zerlege
 17* 45 markiere (i 5)
 >17 45< vertausche (i 5<>6)
 >45 17< vertausche (i 5<>6)

```

### Sortieralgorithmus nach Horare

```

1 45 8 53 9 2 17 10 zerlege
1 45 8 53* 9 2 17 10 markiere (i 3)
1 45 8 >53 9 2 17 10< vertausche (i 3<>7)
1 45 8 10 9 2 17 zerlege
1 45 8 10* 9 2 17 markiere (i 3)
1 >45 8 10 9 2< 17 vertausche (i 1<>5)
1 2 8 >10 9< 45 17 vertausche (i 3<>4)
1 2 8 9
1 2* 8 9
1 2
1* 2
 8 9
 8* 9
 10 45 17
 10 45* 17
 10 >45 17<
 10 17
 10* 17

```

## 66115 / 2016 / Frühjahr / Thema 2 / Aufgabe 1

Beantworten Sie kurz, präzise und mit Begründung folgende Fragen: (Die Begründungen müssen keine formellen mathematischen Beweise sein).

- (a) Welche Möglichkeiten gibt es, eine formale Sprache vom Typ 3 zu definieren?

Lösungsvorschlag

- reguläre Grammatik
- endlicher nichtdeterministische und deterministischer Automat
- regulärer Ausdruck

- (b) Was ist die Komplexität des Wortproblems für Typ-3 Sprachen und wieso ist das so?

$P$ , CYK-Algorithmus löst es in Polynomialzeit.

- (c) Sind Syntaxbäume zu einer Grammatik immer eindeutig? Falls nicht, geben Sie ein Gegenbeispiel.

Nein. Syntaxbäume zu einer Grammatik sind nicht immer eindeutig.

**Gegenbeispiel**

$$G = (\{S, A, B\}, \{a\}, P, S)$$

$$P = \left\{ \right.$$

$$S \rightarrow AA$$

$$S \rightarrow BB$$

$$A \rightarrow a$$

$$B \rightarrow a$$

$$\left. \right\}$$

$$S \vdash AA \vdash aA \vdash aa$$

$$S \vdash BB \vdash aB \vdash aa$$

- (d) Wie kann man die Äquivalenz zweier Typ-3 Sprachen nachweisen?

Myhill-Nerode-Äquivalenz. Wir können von den auf Äquivalenz zu überprüfenden Sprachen jeweils einen minimalen endlichen Automaten bilden. Sind diese entstanden zwei Automaten äquivalent so sind auch die Sprachen äquivalent.

- (e) Wie kann man das Wortproblem für das Komplement einer Typ-3 Sprache lösen?

Da das Komplement einer regulären Sprache wieder eine reguläre Sprache ergibt, kann das Wortproblem beim Komplement durch einen deterministisch endlichen Automaten gelöst werden. Tausche akzeptierende mit nicht akzeptierenden Zuständen des zugehörigen Automaten.

Alternativ: Ergebnis des CYK-Algorithmus invertieren oder CYK auf Komplement, da reguläre Sprachen unter dem Komplement abgeschlossen sind.

- (f) Weshalb gilt das Pumping-Lemma für Typ 3 Sprachen?

endliche Anzahl von Zuständen  $n$  im Automaten  $\rightarrow$  für Wörter  $|\omega| > n$  muss Zyklus vorhanden sein.

- (g) Ist der Nachweis, dass das Typ-3 Pumping-Lemma für eine gegebene Sprache gilt, ausreichend, um zu zeigen, dass die Sprache vom Typ 3 ist? Falls nicht, geben Sie ein Gegenbeispiel, mit Begründung.

Nein:

Die Sprache  $L = \{a^m b^n c^n \mid m, n \geq 1\} \cup \{b^m c^n \mid m, n \geq 0\}$  ist nicht regulär. Allerdings erfüllt  $L$  die Eigenschaften des Pumping-Lemmas, denn jedes Wort  $z \in L$  lässt sich so zerlegen  $z = uvw$ , dass auch für alle  $i \geq 0$   $uv^i w \in L$ . Dazu kann  $v$  einfach als erster Buchstabe gewählt werden. Dieser ist entweder ein  $a$ , die Anzahl von führenden  $as$  ist beliebig. Oder er ist ein  $b$  oder  $c$ , ohne führende  $as$  ist aber die Anzahl von führenden  $bs$  oder  $cs$  beliebig.<sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/Pumping-Lemma>

- (h) Geben Sie ein Beispiel, an dem deutlich wird, dass deterministische und nichtdeterministische Typ-2 Sprachen unterschiedlich sind.

**Deterministisch Kontextfrei**  $L = \{0^n 1^n \mid n \geq 0\}$

**Nichtdeterministisch Kontextfrei**  $L = \{\omega \omega^R \mid \omega \in \{0,1\}^*\}$  (R steht für rückwärts)

<sup>a</sup>

<sup>a</sup><https://docplayer.org/19566652-Einfuehrung-in-die-theoretische-informatik.html>

- (i) Worin macht sich der Unterschied zwischen Typ 0 und 1 bemerkbar, wenn man Turingmaschinen benutzt, um das Wortproblem vom Typ 0 oder 1 zu lösen. Warum ist das so?

Typ 0: semi-entscheidbar, Typ 1: entscheidbar

Typ 0: Unendlichkeit des Band kann die unendlich lange Berechenbarkeit zustande kommen.

Typ 1: Linear beschränkte Turingmaschine endlich, dadurch Anzahl an Kombination

Typ 1 Sprachen sind monoton wachsend.

Da Typ 1 nur Wörter verlängert, kann daher in Polynomialzeit überprüft werden, ob das Wort in der Sprache liegt, indem die Regeln angewendet werden, bis das Wortende erreicht ist.

**66115 / 2016 / Frühjahr / Thema 2 / Aufgabe 2**

Beantworten Sie kurz, präzise und mit Begründung folgende Fragen: (Die Begründungen müssen keine formellen mathematischen Beweise sein)

- (a) Warum genügt es, sich auf Funktionen zu beschränken, die natürliche Zahlen auf natürliche Zahlen abbilden, wenn man untersuchen will, was heutige Computer im Prinzip berechnen können?

Lösungsvorschlag

Jede WHILE-berechenbare Funktion ist turing-berechenbar. Jede turing-berechenbare Funktion ist WHILE-berechenbar. Jede nichtdeterministische Turing-Maschine kann durch eine deterministische Turing-Maschine simuliert werden.

- (b) Was besagt die Church-Turing-These? Könnte man sie beweisen oder widerlegen?

Lösungsvorschlag

Die Church-Turing-These besagt, dass die Klasse der turing-berechenbaren Funktionen mit der Klasse der intuitiv berechenbaren Funktionen übereinstimmt. Die These kann nicht bewiesen oder widerlegt werden, weil es sich bei dem Begriff „*intuitiv berechenbare Funktion*“ um keinen mathematisch exakt definierten Begriff handelt. Würde man ihn genau definiert, würde ein konkretes Berechnungsmodell festgelegt werden, was der Kernaussage dieses Begriffes widersprechen würde.

- (c) Für reelle Zahlen, wie z. B.  $\pi$ , lässt sich die Dezimaldarstellung durch entsprechende Programme beliebig genau approximieren. Gilt das für alle reellen Zahlen, lässt sich für jede reelle Zahl die Dezimaldarstellung mit entsprechenden Programmen beliebig genau approximieren?

Lösungsvorschlag

Ja mit einer Berechnungsvorschrift, ja solange der Speicherplatz reicht, z. B. mittels Intervallschachtelung.

- (d) Was ist für die Berechnungskraft der wesentliche Unterschied zwischen While-Berechenbarkeit und Loop-Berechenbarkeit.

Lösungsvorschlag

Alle LOOP-Programme sind mathematisch betrachtet totale Funktionen, die terminieren immer. WHILE-Programme hingegen partiellen Funktionen, die nicht für alle Eingabekombinationen terminieren.

- (e) Die Ackermannfunktion ist ein Beispiel einer totalen Funktion, die While-berechenbar, aber nicht Loop-berechenbar ist. Sie verallgemeinert die Idee, dass Multiplikation die wiederholte Addition ist, Exponentiation die wiederholte Multiplikation, Hyperexponentiation die wiederholte Exponentiation usw. Die Stufe dieser hyperhyper ... Exponentiation ist ein Parameter der Ackermannfunktion. Generieren Sie aus dieser Idee ein Argument, das illustriert, warum die Ackermannfunktion nicht Loop-berechenbar ist.

Jedes LOOP-Programm benötigt zur Berechnung der Funktion  $x \uparrow^n y$  mindestens  $n + 2$  Schleifen.

Gäbe es ein LOOP-Programm, das  $ack(n, m)$  tatsächlich für beliebige Werte von  $n$  berechnet, so müsste dieses — im Widerspruch zum endlichen Aufbau eines Loop-Programms — unendlich viele Schleifenkonstrukte enthalten.

### Hyperexponentiation

- $x \uparrow^{-1} y = x + y$
- $x \uparrow^0 y = x \cdot y$
- $x \uparrow^1 y = x^y$
- $x \uparrow^2 y = x^{y^y}$

- (f) Geben Sie ein Beispiel einer Menge an, die abzählbar, aber nicht rekursiv aufzählbar ist, und begründen Sie es.

Lösungsvorschlag

- Die Menge der Primzahlen. Die Primzahl sind unendlich groß und könnten abgezählt werden. Bei Primzahlen gibt es keine Berechnungsvorschrift, die z. b. die 7. Primzahl berechnet.
- Die Menge der Turingmaschine
- Diagonalsprache
- Das Komplement des Halteproblems. Die dem Halteproblem zugrundeliegende Lösungsmenge ist rekursiv aufzählbar. Wäre das Komplement des Halteproblems auch rekursiv aufzählbar, dann wäre das Halteproblem entscheidbar, was es aber nicht ist. <sup>a</sup>

<sup>a</sup><https://www.youtube.com/watch?v=om4ZT0eQuD0>

- (g) Wie ist der Zusammenhang zwischen rekursiv aufzählbar und semi-entscheidbar?

Lösungsvorschlag

Die beiden Begriffe sind äquivalent. <sup>a</sup>

<sup>a</sup><https://www.youtube.com/watch?v=om4ZT0eQuD0>

## 66115 / 2016 / Frühjahr / Thema 2 / Aufgabe 3

Beantworten Sie kurz, präzise und mit Begründung folgende Fragen: (Die Begründungen müssen keine formellen mathematischen Beweise sein)

- (a) In der O-Notation insbesondere für die Zeitkomplexität von Algorithmen lässt man i. A. konstante Faktoren oder kleinere Terme weg. Z. B. schreibt man anstelle  $\mathcal{O}(3n^2 + 5)$  einfach nur  $\mathcal{O}(n^2)$ . Warum macht man das so?

Das Wachstum im Unendlich ist bestimmt durch den größten Exponenten. Konstante falle bei einer asymptotischen. Analyse weg. nicht wesentlich schneller

- (b) Was ist die typische Vorgehensweise, wenn man für ein neues Problem die NP-Vollständigkeit untersuchen will?

Die alten Probleme werden reduziert. Das neue Problem ist größer als die alten Probleme. Das Problem muss in NP liegen.

- (i) Problem *in* NP durch Angabe eines nichtdeterministischen Algorithmus in Polynomialzeit
- (ii) Problem NP-schwer via Reduktion:  $L_{\text{NP-vollständig}} \leq L_{\text{neues Problem}}$

- (c) Was könnte man tun, um  $P = NP$  zu beweisen?

Es würde genügen, zu einem einzigen NP-Problem beweisen, dass es in P liegt.

Zu einem Problem einen deterministischen Turingmaschin finden, die es in polynomineller Zeit löst.

- (d) Sind NP-vollständige Problem mit Loop-Programmen lösbar? (Antwort mit Begründung!)

nicht lösbar mit Loop-Programmen. Begründung ähnlich wie bei der Ackermann-Funktion. z. B. Passwort. Zu Passwort beliebig bräuchte man beliebige for schleifen, was dem endlichen Anzahl an Loop-Schleifen widerspricht.

- (e) Wie zeigt man aus der NP-Härte des SAT-Problems die NP-Härte des 3SAT-Problems? (3SAT ist ein SAT-Problem wobei alle Klauseln maximal 3 Literale haben.)

in den Lösungen enthalten

## 66115 / 2016 / Frühjahr / Thema 2 / Aufgabe 4

Betrachte eine Hashtabelle der Größe  $m = 10$ .

- (a) Welche der folgenden Hashfunktionen ist für Hashing mit verketteten Listen am besten geeignet? Begründen Sie Ihre Wahl!

- (i)  $h_1(x) = (4x + 3) \bmod m$



$$\begin{aligned}
1 \quad & h_1(1) = (4 \cdot 1 + 3) \bmod 10 = 7 \\
2 \quad & h_1(2) = (4 \cdot 2 + 3) \bmod 10 = 1 \\
3 \quad & h_1(3) = (4 \cdot 3 + 3) \bmod 10 = 5 \\
4 \quad & h_1(4) = (4 \cdot 4 + 3) \bmod 10 = 9 \\
5 \quad & h_1(5) = (4 \cdot 5 + 3) \bmod 10 = 3 \\
6 \quad & h_1(6) = (4 \cdot 6 + 3) \bmod 10 = 7 \\
7 \quad & h_1(7) = (4 \cdot 7 + 3) \bmod 10 = 1 \\
8 \quad & h_1(8) = (4 \cdot 8 + 3) \bmod 10 = 5 \\
9 \quad & h_1(9) = (4 \cdot 9 + 3) \bmod 10 = 9 \\
10 \quad & h_1(10) = (4 \cdot 10 + 3) \bmod 10 = 3
\end{aligned}$$

(ii)  $h_2(x) = (3x + 3) \bmod m$

$$\begin{aligned}
1 \quad & h_2(1) = (3 \cdot 1 + 3) \bmod 10 = 6 \\
2 \quad & h_2(2) = (3 \cdot 2 + 3) \bmod 10 = 9 \\
3 \quad & h_2(3) = (3 \cdot 3 + 3) \bmod 10 = 2 \\
4 \quad & h_2(4) = (3 \cdot 4 + 3) \bmod 10 = 5 \\
5 \quad & h_2(5) = (3 \cdot 5 + 3) \bmod 10 = 8 \\
6 \quad & h_2(6) = (3 \cdot 6 + 3) \bmod 10 = 1 \\
7 \quad & h_2(7) = (3 \cdot 7 + 3) \bmod 10 = 4 \\
8 \quad & h_2(8) = (3 \cdot 8 + 3) \bmod 10 = 7 \\
9 \quad & h_2(9) = (3 \cdot 9 + 3) \bmod 10 = 0 \\
10 \quad & h_2(10) = (3 \cdot 10 + 3) \bmod 10 = 3
\end{aligned}$$

Damit die verketteten Listen möglichst klein bleiben, ist eine möglichst gleichmäßige Verteilung der Schlüssel in die Buckets anzustreben.  $h_2$  ist dafür besser geeignet als  $h_1$ , da  $h_2$  in alle Buckets Schlüssel ablegt,  $h_1$  jedoch nur in Buckets mit ungerader Zahl.

(b) Welche der folgenden Hashfunktionen ist für Hashing mit offener Adressierung am besten geeignet? Begründen Sie Ihre Wahl!

(i)  $h_1(x, i) = (7 \cdot x + i \cdot m) \bmod m$

(ii)  $h_2(x, i) = (7 \cdot x + i \cdot (m - 1)) \bmod m$

$h_2(x, i)$  ist besser geeignet.  $h_1$  sondiert immer im selben Bucket,  $(i \cdot m) \bmod m$  heben sich gegenseitig auf, zum Beispiel ergibt:

$$- h_1(3, 0) = (7 \cdot 3 + 0 \cdot 10) \bmod 10 = 1$$

$$- h_1(3,1) = (7 \cdot 3 + 1 \cdot 10) \bmod 10 = 1$$

$$- h_1(3,2) = (7 \cdot 3 + 2 \cdot 10) \bmod 10 = 1$$

Während hingegen  $h_2$  verschiedene Buckets belegt.

$$- h_2(3,0) = (7 \cdot 3 + 0 \cdot 9) \bmod 10 = 1$$

$$- h_2(3,1) = (7 \cdot 3 + 1 \cdot 9) \bmod 10 = 0$$

$$- h_2(3,2) = (7 \cdot 3 + 2 \cdot 9) \bmod 10 = 9$$

## 66115 / 2016 / Frühjahr / Thema 2 / Aufgabe 6

- (a) Berechnen Sie für folgenden Graphen den kürzesten Weg von Karlsruhe nach Kassel und dokumentieren Sie den Berechnungsweg:

### Verwendete Abkürzungen:

**A** Augsburg

**EF** Erfurt

**F** Frankfurt

**KA** Karlsruhe

**KS** Kassel

**M** München

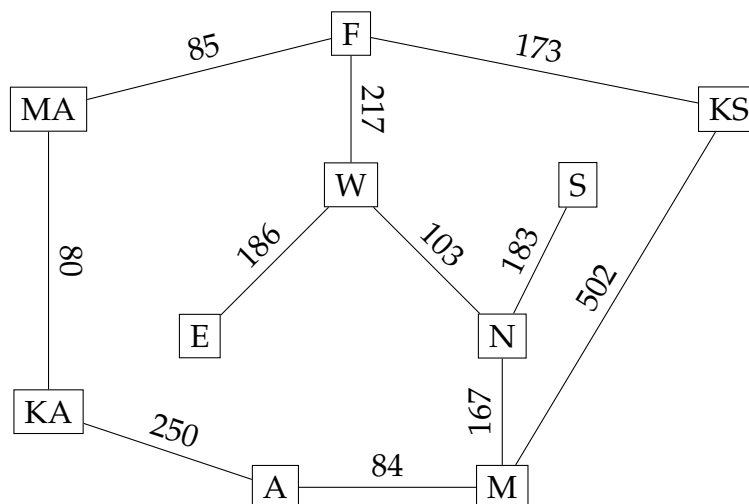
**MA** Mannheim

**N** Nürnberg

**S** Stuttgart

**WÜ** Würzburg

Zahl = Zahl in Kilometern



| Nr.                 | besucht | A          | E          | F           | KA       | KS                                                                                  | M          | MA        | N          | S          | W          |
|---------------------|---------|------------|------------|-------------|----------|-------------------------------------------------------------------------------------|------------|-----------|------------|------------|------------|
| 0                   |         | $\infty$   | $\infty$   | $\infty$    | 0        | $\infty$                                                                            | $\infty$   | $\infty$  | $\infty$   | $\infty$   | $\infty$   |
| 1                   | KA      | <b>250</b> | $\infty$   | $\infty$    | <b>0</b> | $\infty$                                                                            | $\infty$   | 80        | $\infty$   | $\infty$   | $\infty$   |
| 2                   | MA      |            | $\infty$   | 165         |          | $\infty$                                                                            | $\infty$   | <b>80</b> | $\infty$   | $\infty$   | $\infty$   |
| 3                   | F       |            | $\infty$   | <b>165</b>  |          | 338                                                                                 | $\infty$   |           | $\infty$   | $\infty$   | 382        |
| 4                   | A       |            | $\infty$   |             |          | 338                                                                                 | 334        |           | $\infty$   | $\infty$   | 382        |
| 5                   | M       |            | $\infty$   |             |          | 338                                                                                 | <b>334</b> |           | 501        | $\infty$   | 382        |
| 6                   | KS      |            | $\infty$   |             |          | <b>338</b>                                                                          |            |           | 501        | $\infty$   | 382        |
| 7                   | W       |            | 568        |             |          |                                                                                     |            |           | 485        | $\infty$   | <b>382</b> |
| 8                   | N       |            | 568        |             |          |                                                                                     |            |           | <b>485</b> | 668        |            |
| 9                   | E       |            | <b>568</b> |             |          |                                                                                     |            |           |            | 668        |            |
| 10                  | S       |            |            |             |          |                                                                                     |            |           |            | <b>668</b> |            |
| nach                |         | Entfernung |            | Reihenfolge |          | Pfad                                                                                |            |           |            |            |            |
| KA $\rightarrow$ A  | 250     |            | 0          |             |          | KA $\rightarrow$ A                                                                  |            |           |            |            |            |
| KA $\rightarrow$ E  | 568     |            | 9          |             |          | KA $\rightarrow$ MA $\rightarrow$ F $\rightarrow$ W $\rightarrow$ E                 |            |           |            |            |            |
| KA $\rightarrow$ F  | 165     |            | 3          |             |          | KA $\rightarrow$ MA $\rightarrow$ F                                                 |            |           |            |            |            |
| KA $\rightarrow$ KA | 0       |            | 1          |             |          |                                                                                     |            |           |            |            |            |
| KA $\rightarrow$ KS | 338     |            | 6          |             |          | KA $\rightarrow$ MA $\rightarrow$ F $\rightarrow$ KS                                |            |           |            |            |            |
| KA $\rightarrow$ M  | 334     |            | 5          |             |          | KA $\rightarrow$ A $\rightarrow$ M                                                  |            |           |            |            |            |
| KA $\rightarrow$ MA | 80      |            | 2          |             |          | KA $\rightarrow$ MA                                                                 |            |           |            |            |            |
| KA $\rightarrow$ N  | 485     |            | 8          |             |          | KA $\rightarrow$ MA $\rightarrow$ F $\rightarrow$ W $\rightarrow$ N                 |            |           |            |            |            |
| KA $\rightarrow$ S  | 668     |            | 10         |             |          | KA $\rightarrow$ MA $\rightarrow$ F $\rightarrow$ W $\rightarrow$ N $\rightarrow$ S |            |           |            |            |            |
| KA $\rightarrow$ W  | 382     |            | 7          |             |          | KA $\rightarrow$ MA $\rightarrow$ F $\rightarrow$ W                                 |            |           |            |            |            |

- (b) Könnte man den Dijkstra Algorithmus auch benutzen, um das Travelling-Salesman Problem zu lösen?

## 66115 / 2016 / Frühjahr / Thema 2 / Aufgabe 7

Wofür eignen sich die folgenden Baum-Datenstrukturen im Vergleich zu den anderen angeführten Baumstrukturen am besten, und warum. Sprechen Sie auch die Komplexität der wesentlichen Operationen und die Art der Speicherung an.

- (a) Rot-Schwarz-Baum

**Einfügen (Zeitkomplexität)** $\mathcal{O}(\log n)$  (im Durchschnitt) $\mathcal{O}(\log n)$  (im schlechtesten Fall)**Löschen (Zeitkomplexität)** $\mathcal{O}(\log n)$  (im Durchschnitt) $\mathcal{O}(\log n)$  (im schlechtesten Fall)**Suchen (Zeitkomplexität)** $\mathcal{O}(\log n)$  (im Durchschnitt) $\mathcal{O}(\log n)$  (im schlechtesten Fall) <sup>a</sup><sup>a</sup>tutorialspoint.com

## (b) AVL-Baum

**Einfügen (Zeitkomplexität)** $\mathcal{O}(\log_2 n)$  (im Durchschnitt) $\mathcal{O}(\log_2 n)$  (im schlechtesten Fall)**Löschen (Zeitkomplexität)** $\mathcal{O}(\log_2 n)$  (im Durchschnitt) $\mathcal{O}(\log_2 n)$  (im schlechtesten Fall)**Suchen (Zeitkomplexität)** $\mathcal{O}(\log_2 n)$  (im Durchschnitt) $\mathcal{O}(\log_2 n)$  (im schlechtesten Fall) <sup>a</sup><sup>a</sup>tutorialspoint.com

## (c) Binärer-Heap

**Verwendungszweck** zum effizienten Sortieren von Elementen. <sup>a</sup>**Einfügen (Zeitkomplexität)** $\mathcal{O}(1)$  (im Durchschnitt) $\mathcal{O}(\log n)$  (im schlechtesten Fall)**Löschen (Zeitkomplexität)** $\mathcal{O}(\log n)$  (im Durchschnitt) $\mathcal{O}(\log n)$  (im schlechtesten Fall)**Suchen (Zeitkomplexität)** $\mathcal{O}(n)$  (im Durchschnitt) $\mathcal{O}(n)$  (im schlechtesten Fall) <sup>b</sup><sup>a</sup>deut. Wikipedia<sup>b</sup>engl. Wikipedia

## (d) B-Baum

**Einfügen (Zeitkomplexität)** $\mathcal{O}(\log n)$  (im Durchschnitt) $\mathcal{O}(\log n)$  (im schlechtesten Fall)**Löschen (Zeitkomplexität)** $\mathcal{O}(\log n)$  (im Durchschnitt) $\mathcal{O}(\log n)$  (im schlechtesten Fall)**Suchen (Zeitkomplexität)** $\mathcal{O}(\log n)$  (im Durchschnitt) $\mathcal{O}(\log n)$  (im schlechtesten Fall) <sup>a</sup><sup>a</sup>tutorialspoint.com

(e) R-Baum

**Verwendungszweck** Ein R-Baum erlaubt die schnelle Suche in mehrdimensionalen ausgedehnten Objekten. <sup>a</sup>**Suchen (Zeitkomplexität)** $\mathcal{O}(\log_M n)$  (im Durchschnitt) <sup>b</sup> $\mathcal{O}(n)$  (im schlechtesten Fall) <sup>c</sup><sup>a</sup>deut. Wikipedia<sup>b</sup>eng. Wikipedia<sup>c</sup>Simon Fraser University, Burnaby, Kanada**66115 / 2016 / Herbst / Thema 1 / Aufgabe 1**

Gegeben ist der deterministische endliche Automat  $A = (\{A, B, C, D, E\}, \{0, 1\}, \delta, \{E\}, A)$ , wobei

- (a) Minimieren Sie den Automaten mit dem bekannten Minimierungsalgorithmus. Dokumentieren Sie die Schritte geeignet.
- (b) Geben Sie einen regulären Ausdruck für die erkannte Sprache an.
- (c) Geben Sie die Äquivalenzklassen der Myhill-Nerode-Äquivalenz der Sprache durch reguläre Ausdrücke an.
- (d) Geben Sie ein Beispiel einer regulären Sprache an, für die kein deterministischer endlicher Automat mit höchstens zwei Endzuständen existiert.
- (e) Geben Sie ein Beispiel einer regulären Sprache an, für die kein deterministischer endlicher Automat mit weniger als fünf Zuständen existiert.

**66115 / 2016 / Herbst / Thema 1 / Aufgabe 2**

Ordnen Sie die folgenden Sprachen über  $\Sigma = \{a, b\}$  bestmöglich in die Chomsky-Hierarchie ein und geben Sie jeweils eine kurze Begründung (1-2 Sätze).

(a)  $L_1 = \{a^n b^n \mid n \geq 1\}$

Lösungsvorschlag

Typ-2-Sprache: Die Sprache  $L_1$  ist kontextfrei, denn die Sprache braucht einen Speicher, da sie sich die Anzahl der  $a$ 's merken muss, um die gleiche Anzahl an  $b$ 's produzieren zu können. Dies ist mit einem Kellerautomaten möglich. Eine Grammtik der Sprache ist  $G = (\{S\}, \Sigma, \{S \rightarrow aSb \mid \varepsilon\}, S)$

(b)  $L_2 = \{a^n b^n \mid \text{die Turingmaschine mit Gödelnummer } n \text{ hält auf leerer Eingabe}\}$

Lösungsvorschlag

Typ-0-Sprache: Die Sprache hat eine Typ-0-Grammatik, da sie offensichtlich semientscheidbar, aber nicht entscheidbar ist.

(c)  $L_3 = \Sigma^* \setminus L_1$

Lösungsvorschlag

Typ-2-Sprache: Die Sprache  $L_3$  ist kontextfrei, da ein PDA existiert, der nicht akzeptiert, wenn er  $L_1$  akzeptiert. (Ausgänge umgepolt)

(d)  $L_4 = \Sigma^* \setminus L_2$

Lösungsvorschlag

Nicht in der Hierarchie: Das Komplement einer semi- aber unentscheidbaren Sprache kann nicht semi-entscheidbar sein, da  $L$  sonst entscheidbar wäre.

(e)  $L_5 = \{a^n b^m \mid n + m \text{ ist ein Vielfaches von drei}\}$

Lösungsvorschlag

Typ-3-Sprache: regulär, 2 Teilautomaten mit je 3 Zuständen (modulo 2 mal)

(f)  $L_6 = \{a^n b^n \mid n \text{ Quadratzahl}\}$

Lösungsvorschlag

nicht regulär, nicht kontextfrei (Pumping-Lemma)

**66115 / 2016 / Herbst / Thema 1 / Aufgabe 4**

Es sei  $A[0 \dots n-1]$  ein Array von paarweise verschiedenen ganzen Zahlen.

Wir interessieren uns für die Zahl der Inversionen von  $A$ ; das sind Paare von Indices  $(i, j)$ , sodass  $i < j$  aber  $A[i] > A[j]$ . Die Inversionen im Array  $[2, 3, 8, 6, 1]$  sind  $(0, 4)$ , da  $A[0] > A[4]$  und weiter  $(1, 4)$ ,  $(2, 3)$ ,  $(2, 4)$ ,  $(3, 4)$ . Es gibt also 5 Inversionen.

(a) Wie viel Inversionen hat das Array  $[3, 7, 1, 4, 5, 9, 2]$ ?

- (0,1):  $3 > 1$
- (0,6):  $3 > 2$
- (1,2):  $7 > 1$
- (1,3):  $7 > 4$
- (1,4):  $7 > 5$
- (1,6):  $7 > 2$
- (3,6):  $4 > 2$
- (4,6):  $5 > 2$

- (b) Welches Array mit den Einträgen  $\{1, \dots, n\}$  hat die meisten Inversionen, welches hat die wenigsten?

Folgt nach der 1 eine absteigend sortierte Folge, so hat sie am meisten Inversionen, z. B.  $\{1, 7, 6, 5, 4, 3, 2\}$ . Eine aufsteigend sortierte Zahlenfolge hat keine Inversionen, z. B.  $\{1, 2, 3, 4, 5, 6, 7\}$ .

- (c) Entwerfen Sie eine Prozedur `int merge(int[] a, int i, int h, int j);` welche das Teilarray  $a[i..j]$  sortiert und die Zahl der in ihm enthaltenen Inversionen zurückliefert, wobei die folgenden Vorbedingungen angenommen werden:

- $0 \leq i \leq h \leq j < n$ , wobei  $n$  die Länge von  $a$  ist ( $n = a.length$ ).
- $a[i \dots h]$  und  $a[h + 1 \dots j]$  sind aufsteigend sortiert.
- Die Einträge von  $a[i \dots j]$  sind paarweise verschieden.

Ihre Prozedur soll in linearer Zeit, also  $\mathcal{O}(j - i)$  laufen. Orientieren Sie sich bei Ihrer Lösung an der Mischoperation des bekannten Mergesort-Verfahrens.

- (d) Entwerfen Sie nun ein Divide-and-Conquer-Verfahren zur Bestimmung der Zahl der Inversionen, indem Sie angelehnt an das Mergesort-Verfahren einen Algorithmus `ZI` beschreiben, der ein gegebenes Array in sortierter Form liefert und gleichzeitig dessen Inversionsanzahl berechnet. Im Beispiel wäre also

$$ZI([2, 3, 8, 6, 1]) = ([1, 2, 3, 6, 8], 5)$$

Die Laufzeit Ihres Algorithmus auf einem Array der Größe  $n$  soll  $\mathcal{O}(n \log(n))$  sein.

Sie dürfen die Hilfsprozedur `merge` aus dem vorherigen Aufgabenteil verwenden, auch, wenn Sie diese nicht gelöst haben.

- (e) Begründen Sie, dass Ihr Algorithmus die Laufzeit  $\mathcal{O}(n \log(n))$  hat.

(f) Geben Sie die Lösungen folgender asymptotischer Rekurrenzen (in O-Notation) an:

- (i)  $T(n) = 2 \cdot T(\frac{n}{2}) + \mathcal{O}(\log n)$
- (ii)  $T(n) = 2 \cdot T(\frac{n}{2}) + \mathcal{O}(n^2)$
- (iii)  $T(n) = 3 \cdot T(\frac{n}{2}) + \mathcal{O}(n)$

## 66115 / 2016 / Herbst / Thema 2 / Aufgabe 3

Sei  $M_0, M_1, \dots$  eine Registermaschinen (RAMs). Beantworten Sie folgende Fragen zur Aufzählbarkeit und Entscheidbarkeit. Beweisen Sie Ihre Antwort.

### Exkurs: Registermaschinen (RAMs)

Die Random Access Machine (kurz RAM) ist eine spezielle Art von Registermaschine. Sie hat die Fähigkeit der indirekten Adressierung der Register.

Die Random Access Machine besteht aus:

- einem Programm bestehend aus endlich vielen durchnummerierten Befehlen (beginnend mit Nummer 1)
- einem Befehlszähler  $b$
- einem Akkumulator  $c(0)$
- und einem unendlich großen Speicher aus durchnummerierten Speicherzellen (Registern)  $c(1), c(2), c(3), \dots$

Jedes Register (einschließlich  $b$  und  $c(0)$ ) speichert eine beliebig große natürliche Zahl.

(a) Ist folgende Menge entscheidbar?

$$A = \{x \in \mathbb{N} \mid x = 100 \text{ oder } M_x \text{ hält bei Eingabe } x\}$$

Lösungsvorschlag

Ja,  $x \geq 100$  ist entscheidbar und aufgrund des „oder“ ist die 2. Bedingung nur für  $x < 100$  relevant. Da  $x < 100$  eine endliche Menge darstellt, kann eine endliche Liste geführt werden und ein Experte kann für jeden Fall entscheiden, ob  $M_x$  hält oder nicht, somit ist  $A$  entscheidbar.

(b) Ist folgende Menge entscheidbar?

$$B = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid M_x \text{ hält bei Eingabe } x \text{ genau dann, wenn } M_y \text{ bei Eingabe } y \text{ hält}\}$$

Lösungsvorschlag

Nein. Dieses Problem entspricht der parallelen Ausführung des Halteproblems auf zwei Bändern. Das Halteproblem ist unentscheidbar, damit ist auch die parallele Ausführung des Halteproblems und damit  $B$  unentscheidbar.

(c) Ist folgende Menge aufzählbar?

$$C = \{x \in \mathbb{N} \mid M_x \text{ hält bei Eingabe } 0 \text{ mit dem Ergebnis } 1\}$$



Ja, die Menge ist aufzählbar, da die Menge aller Turningmaschinen aufzählbar und über natürliche Zahlen definiert ist (die wiederum aufzählbar sind).

## 66115 / 2016 / Herbst / Thema 2 / Aufgabe 7

- (a) Gegeben ist die Ausgabe der Methode **Partition** (s. Pseudocode), rekonstruieren Sie die Eingabe.

Konkret sollen Sie das Array  $A = (\_, \_, 1, \_, \_)$  so vervollständigen, dass der Aufruf  $\text{Partition}(A, 1, 5)$  die Zahl 3 zurückgibt und nach dem Aufruf gilt, dass  $A = (1, 2, 3, 4, 5)$  ist.

Geben Sie  $A$  nach jedem Durchgang der for-Schleife in **Partition** an.

Lösungsvorschlag

```

2 4 1 5 3 Eingabe
2 4 1 5 3 zerlege
2 4 1 5 3* markiere (i 4)
>2< 4 1 5 3 vertausche (i 0<>0)
2 >4 1< 5 3 vertausche (i 1<>2)
2 1 >4 5 3< vertausche (i 2<>4)
2 1
2 1*
>2 1<
5 4 zerlege
5 4* markiere (i 4)
>5 4< vertausche (i 3<>4)
1 2 3 4 5 Ausgabe

```

- (b) Beweisen Sie die Korrektheit von **Partition** (z. B. mittels einer Schleifeninvarianten)!
- (c) Geben Sie für jede natürliche Zahl  $n$  eine Instanz  $I_n$ , der Länge  $n$  an, so dass  $\text{QuickSort}(I_n) \Omega(n^2)$  Zeit benötigt. Begründen Sie Ihre Behauptung.

Lösungsvorschlag

$$I_n = 1, 2, 3, \dots, n$$

Die Methode **Partition** wird  $n$  mal aufgerufen, weil bei jedem Aufruf der Methode nur eine Zahl, nämlich die größte Zahl, abgespalten wird.

- $\text{Partition}(A, 1, n)$
- $\text{Partition}(A, 1, n - 1)$
- $\text{Partition}(A, 1, n - 2)$
- $\text{Partition}(A, 1, \dots)$
- $\text{Partition}(A, 1, 1)$

In der For-Schleife der Methode Partition wird bei jeder Wiederholung ein Vertauschvorgang durchgeführt (Die Zahlen werden mit sich selbst getauscht.)

```

1 2 3 4 5 6 7 zerlege
1 2 3 4 5 6 7* markiere (i 6)
>1< 2 3 4 5 6 7 vertausche (i 0<>0)
1 >2< 3 4 5 6 7 vertausche (i 1<>1)
1 2 >3< 4 5 6 7 vertausche (i 2<>2)
1 2 3 >4< 5 6 7 vertausche (i 3<>3)
1 2 3 4 >5< 6 7 vertausche (i 4<>4)
1 2 3 4 5 >6< 7 vertausche (i 5<>5)
1 2 3 4 5 6 >7< vertausche (i 6<>6)
1 2 3 4 5 6 zerlege
1 2 3 4 5 6* markiere (i 5)
>1< 2 3 4 5 6 vertausche (i 0<>0)
1 >2< 3 4 5 6 vertausche (i 1<>1)
1 2 >3< 4 5 6 vertausche (i 2<>2)
1 2 3 >4< 5 6 vertausche (i 3<>3)
1 2 3 4 >5< 6 vertausche (i 4<>4)
1 2 3 4 5 >6< vertausche (i 5<>5)
1 2 3 4 5 zerlege
1 2 3 4 5* markiere (i 4)
>1< 2 3 4 5 vertausche (i 0<>0)
1 >2< 3 4 5 vertausche (i 1<>1)
1 2 >3< 4 5 vertausche (i 2<>2)
1 2 3 >4< 5 vertausche (i 3<>3)
1 2 3 4 >5< vertausche (i 4<>4)
1 2 3 4 zerlege
1 2 3 4* markiere (i 3)
>1< 2 3 4 vertausche (i 0<>0)
1 >2< 3 4 vertausche (i 1<>1)
1 2 >3< 4 vertausche (i 2<>2)
1 2 3 >4< vertausche (i 3<>3)
1 2 3 zerlege
1 2 3* markiere (i 2)
>1< 2 3 vertausche (i 0<>0)
1 >2< 3 vertausche (i 1<>1)
1 2 >3< vertausche (i 2<>2)
1 2 zerlege
1 2* markiere (i 1)
>1< 2 vertausche (i 0<>0)
1 >2< vertausche (i 1<>1)

```

- (d) Was müsste Partition (in Linearzeit) leisten, damit QuickSort Instanzen der Länge  $n$  in  $\mathcal{O}(n \cdot \log n)$  Zeit sortiert? Zeigen Sie, dass Partition mit der von Ihnen geforderten Eigenschaft zur gewünschten Laufzeit von QuickSort führt.

**Exkurs: Master-Theorem**

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

$a$  = Anzahl der rekursiven Aufrufe, Anzahl der Unterprobleme in der Rekursion ( $a \geq 1$ ).

$\frac{1}{b}$  = Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird, Anteil an der Verkleinerung des Problems ( $b > 1$ ).

$f(n)$  = Kosten (Aufwand, Nebenkosten), die durch die Division des Problems und die Kombination der Teillösungen entstehen. Eine von  $T(n)$  unabhängige und nicht negative Funktion.

Dann gilt:

**1. Fall:**  $T(n) \in \Theta\left(n^{\log_b a}\right)$

falls  $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$  für  $\varepsilon > 0$

**2. Fall:**  $T(n) \in \Theta\left(n^{\log_b a} \cdot \log n\right)$

falls  $f(n) \in \Theta\left(n^{\log_b a}\right)$

**3. Fall:**  $T(n) \in \Theta(f(n))$

falls  $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$  für  $\varepsilon > 0$  und ebenfalls für ein  $c$  mit  $0 < c < 1$  und alle hinreichend großen  $n$  gilt:  $a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n)$

Lösungsvorschlag

Die Methode **Partition** müsste die Instanzen der Länge  $n$  in zwei gleich große Teile spalten ( $\frac{n-1}{2}$ ).

**Allgemeine Rekursionsgleichung:**

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

**Anzahl der rekursiven Aufrufe ( $a$ ):**

2

**Anteil Verkleinerung des Problems ( $b$ ):**

um  $\frac{1}{2}$  also  $b = 2$

**Laufzeit der rekursiven Funktion ( $f(n)$ ):**

$n$

**Ergibt folgende Rekursionsgleichung:**

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$$

**1. Fall:**  $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$ :

für  $\varepsilon = 4$ :

$$f(n) = n \notin \mathcal{O}\left(n^{\log_2 2 - \varepsilon}\right)$$

**2. Fall:**  $f(n) \in \Theta(n^{\log_b a})$ :

$$f(n) = n \in \Theta(n^{\log_2 2}) = \Theta(n)$$

**3. Fall:**  $f(n) \in \Omega(n^{\log_b a + \epsilon})$ :

$$f(n) = n \notin \Omega(n^{\log_2 2 + \epsilon})$$

$$\Rightarrow T(n) \in \Theta(n^{\log_2 2} \cdot \log n) = \Theta(n \cdot \log n)$$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

**Funktion** Quicksort( $A, l = 1, r = A.length$ )

```

if $l < r$ then
 $m = \text{Partition}(A, l, r);$
 Quicksort($A, l, m - 1$);
 Quicksort($A, m + 1, r$);
end
```

**Funktion** Partition( $A, \text{int } l, \text{int } r$ )

```

pivot = $A[r];$
 $i = l;$
for $j = l$ to $r - 1$ do
 if $A[j] \leq \text{pivot}$ then
 Swap(A, i, j);
 $i = i + 1;$
 end
end
```

**Funktion** Swap( $A, \text{int } l, \text{int } r$ )

```

temp = $A[i];$
 $A[i] = A[j];$
 $A[j] = \text{temp};$
```

## 66115 / 2017 / Frühjahr / Thema 1 / Aufgabe 1

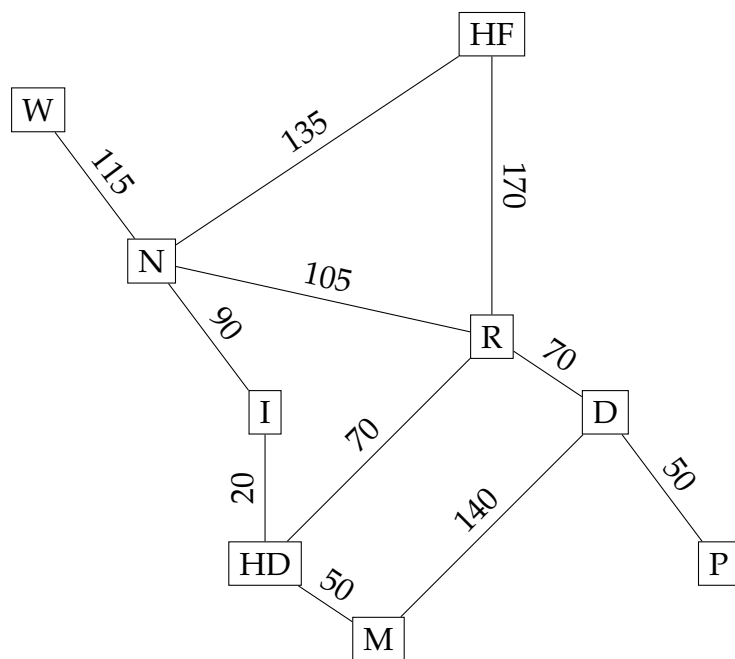
Die folgende Abbildung zeigt die wichtigsten bayerischen Autobahnen zusammen mit einigen anliegenden Orten und die Entfernungen zwischen diesen.

### Entfernungstabelle

| von           | nach          | km  |
|---------------|---------------|-----|
| Würzburg      | Nürnberg      | 115 |
| Nürnberg      | Regensburg    | 105 |
| Regensburg    | AK Deggendorf | 70  |
| AK Deggendorf | Passau        | 50  |
| Hof           | Nürnberg      | 135 |
| Nürnberg      | Ingolstadt    | 90  |
| Ingolstadt    | AD Holledau   | 20  |
| AD Holledau   | München       | 50  |
| München       | AK Deggendorf | 140 |
| Hof           | Regensburg    | 170 |
| Regensburg    | AD Holledau   | 70  |

## Abkürzungen

|    |            |
|----|------------|
| D  | Deggendorf |
| HF | Hof        |
| HD | Holledau   |
| I  | Ingolstadt |
| M  | München    |
| N  | Nürnberg   |
| P  | Passau     |
| R  | Regensburg |
| W  | Würzburg   |



- (a) Bestimmen Sie mit dem Algorithmus von *Dijkstra* den kürzesten Weg von Ingolstadt zu allen anderen Orten. Verwenden Sie zur Lösung eine Tabelle gemäß folgendem Muster und markieren Sie in jeder Zeile den jeweils als nächstes zu betrachtenden Ort. Setzen Sie für die noch zu bearbeitenden Orte eine Prioritätswarteschlange ein, wobei gleicher Entfernung wird der ältere Knoten gewählt.

Lösungsvorschlag

| Nr. | besucht | D          | HD        | HF         | I        | M         | N         | P          | R         | W          |
|-----|---------|------------|-----------|------------|----------|-----------|-----------|------------|-----------|------------|
| 0   |         | $\infty$   | $\infty$  | $\infty$   | 0        | $\infty$  | $\infty$  | $\infty$   | $\infty$  | $\infty$   |
| 1   | I       | $\infty$   | 20        | $\infty$   | <b>0</b> | $\infty$  | 90        | $\infty$   | $\infty$  | $\infty$   |
| 2   | HD      | $\infty$   | <b>20</b> | $\infty$   |          | 70        | 90        | $\infty$   | 90        | $\infty$   |
| 3   | M       | 210        |           | $\infty$   |          | <b>70</b> | 90        | $\infty$   | 90        | $\infty$   |
| 4   | N       | 210        |           | 225        |          |           | <b>90</b> | $\infty$   | 90        | 205        |
| 5   | R       | 160        |           | 225        |          |           |           | $\infty$   | <b>90</b> | 205        |
| 6   | D       | <b>160</b> |           | 225        |          |           |           | 210        |           | 205        |
| 7   | W       |            |           | 225        |          |           |           | 210        |           | <b>205</b> |
| 8   | P       |            |           | 225        |          |           |           | <b>210</b> |           |            |
| 9   | HF      |            |           | <b>225</b> |          |           |           |            |           |            |

- (b) Die bayerische Landesregierung hat beschlossen, die eben betrachteten Orte mit einem breitbandigen Glasfaser-Backbone entlang der Autobahnen zu verbinden. Dabei soll aus Kostengründen so wenig Glasfaser wie möglich verlegt werden. Identifizieren Sie mit dem Algorithmus von Kruskal diejenigen Strecken, entlang

welcher Glasfaser verlegt werden muss. Geben Sie die Ortspaare (Autobahnsegmente) in der Reihenfolge an, in der Sie sie in Ihre Verkabelungsliste aufnehmen.

Lösungsvorschlag

- (c) Um Touristen den Besuch aller Orte so zu ermöglichen, dass sie dabei jeden Autobahnabschnitt genau einmal befahren müssen, bedarf es zumindest eines sogenannten offenen Eulerzugs. Zwischen welchen zwei Orten würden Sie eine Autobahn bauen, damit das bayerische Autobahnnetz mindestens einen Euler-Pfad enthält?

#### Exkurs: offener Eulerzug

Ein offener Eulerzug ist gegeben, wenn Start- und Endknoten nicht gleich sein müssen, wenn also statt eines Zyklus lediglich eine Kantenfolge verlangt wird, welche jede Kante des Graphen genau einmal enthält. Ein bekanntes Beispiel ist das „Haus vom Nikolaus“.

Lösungsvorschlag

Zwischen Deggendorf und Würzburg

$P \rightarrow D \rightarrow R \rightarrow N \rightarrow W \rightarrow D \rightarrow M \rightarrow HD \rightarrow R \rightarrow HF \rightarrow N \rightarrow I \rightarrow HD$

## 66115 / 2017 / Frühjahr / Thema 1 / Aufgabe 2

In dieser Aufgabe sei vereinfachend angenommen, dass sich Top-Level-Domains (TLD) ausschließlich aus zwei oder drei der 26 Kleinbuchstaben des deutschen Alphabets ohne Umlaute zusammensetzen. Im Folgenden sollen TLDs lexikographisch aufsteigend sortiert werden, eine TLD  $(s_1, s_2)$  mit zwei Buchstaben (z. B. „co“ für Kolumbien) wird also vor einer TLD  $(t_1, t_2, t_3)$  der Länge drei (z. B. „com“) einsortiert, wenn  $s_1 < t_1 \vee (s_1 = t_1 \wedge s_2 \leq t_2)$  gilt.

- (a) Sortieren Sie zunächst die Reihung [„de“, „com“, „uk“, „org“, „co“, „net“, „fr“, „ee“] schrittweise unter Verwendung des Radix-Sortierverfahrens (Bucketsort). Erstellen Sie dazu eine Tabelle wie das folgende Muster und tragen Sie dabei in das Feld „Stelle“ die Position des Buchstabens ein, nach dem im jeweiligen Durchgang sortiert wird (das Zeichen am TLD-Anfang habe dabei die „Stelle“ 1).

#### Exkurs: Alphabet

abcdefghijklmnopqrstuvwxyz

Lösungsvorschlag

| Stelle | Reihung |     |     |     |     |     |     |     |
|--------|---------|-----|-----|-----|-----|-----|-----|-----|
|        | de_     | com | uk_ | org | co_ | net | fr_ | ee_ |
| 3      | de_     | uk_ | co_ | fr_ | ee_ | org | com | net |
| 2      | de_     | ee_ | net | uk_ | co_ | com | fr_ | org |
| 1      | co_     | com | de_ | ee_ | fr_ | net | org | uk_ |

- (b) Sortieren Sie nun die gleiche Reihung wieder schrittweise, diesmal jedoch unter Verwendung des Mergesort-Verfahrens (Sortieren durch Mischen). Erstellen Sie dazu eine Tabelle wie das folgende Muster und vermerken Sie in der ersten Spalte jeweils welche Operation durchgeführt wurde: Wenn Sie die Reihung geteilt haben, schreiben Sie in die linke Spalte ein T und markieren Sie die Stelle, an der Sie die Reihung geteilt haben, mit einem senkrechten Strich „|“. Wenn Sie zwei Teilreihungen durch Mischen zusammengeführt haben, schreiben Sie ein M in die linke Spalte und unterstreichen Sie die zusammengemischten Einträge. Beginnen Sie mit dem rekursiven Abstieg immer in der linken Hälfte einer (Teil-)Reihung.

```

0 | Reihung
T | de_ com uk_ org | co_ net fr_ ee_
T | de_ com | uk_ org
T | de_ | com
M | com de_
T | uk_ | org
M | org uk_
M | com de_ org uk_
T | co_ net | fr_ ee_
T | co_ | net
M | co_ net
T | fr_ | ee_
T | ee_ | fr_
M | co_ ee_ fr_ net
M | co_ com de_ ee_ fr_ net org uk_

```

- (c) Implementieren Sie das Sortierverfahren Quicksort für String-TLDs in einer gängigen Programmiersprache Ihrer Wahl. Ihr Programm (Ihre Methode) wird mit drei Parametern gestartet: dem String-Array mit den zu sortierenden TLDs selbst sowie jeweils der Position des ersten und des letzten zu sortierenden Eintrags im Array.

Lösungsvorschlag

```

public class Quicksort {

 public static void swap(String[] array, int index1, int index2) {
 String tmp = array[index1];
 array[index1] = array[index2];
 array[index2] = tmp;
 }
}

```



```
}

public static int partition(String[] array, int first, int last) {
 int pivotIndex = (last + first) / 2;
 String pivotValue = array[pivotIndex];
 int pivotIndexFinal = first;
 swap(array, pivotIndex, last);
 for (int i = first; i < last; i++) {
 if (array[i].compareTo(pivotValue) < 0) {
 swap(array, i, pivotIndexFinal);
 pivotIndexFinal++;
 }
 }
 swap(array, last, pivotIndexFinal);
 return pivotIndexFinal;
}

public static void sort(String[] array, int first, int last) {
 if (first < last) {
 int pivotIndex = partition(array, first, last);
 sort(array, first, pivotIndex - 1);
 sort(array, pivotIndex + 1, last);
 }
}

public static void main(String[] args) {
 String[] array = new String[] { "de", "com", "uk", "org", "co", "net",
 ↪ "fr", "ee" };
 sort(array, 0, array.length - 1);
 for (int i = 0; i < array.length; i++) {
 System.out.println(array[i]);
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2017/fruehjahr/Quicksort.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/Quicksort.java)

## 66115 / 2017 / Frühjahr / Thema 1 / Aufgabe 3

Gegeben seien die folgenden Formeln zur Berechnung der *ersten* Fibonacci-Zahlen:

$$\text{fib}_n = \begin{cases} 1 & \text{falls } n \leq 2 \\ \text{fib}_{n-1} + \text{fib}_{n-2} & \text{sonst} \end{cases}$$

sowie der Partialsumme der Fibonacci-Quadrate:

$$\text{sos}_n = \begin{cases} \text{fib}_n & \text{falls } n = 1 \\ \text{fib}_n^2 + \text{sos}_{n-1} & \text{sonst} \end{cases}$$

Sie dürfen im Folgenden annehmen, dass die Methoden nur mit  $1 \leq n \leq 46$  aufgerufen werden, so dass der Datentyp `long` zur Darstellung aller Werte ausreicht.

**Exkurs: Fibonacci-Folge**

Die Fibonacci-Folge beginnt zweimal mit der Zahl 1. Im Anschluss ergibt jeweils die Summe zweier aufeinanderfolgender Zahlen die unmittelbar danach folgende Zahl: 1, 1, 2, 3, 5, 8, 13

**Exkurs: Partialsumme**

Unter der  $n$ -ten Partialsumme  $s_n$  einer Zahlenfolge  $a_n$  versteht man die Summe der Folgenglieder von  $a_1$  bis  $a_n$ . Die immer weiter fortgesetzte Partialsumme einer (unendlichen) Zahlenfolge nennt man eine (unendliche) Reihe.<sup>a</sup> Partialsummen sind das Bindeglied zwischen Summen und Reihen. Gegeben sei die Reihe  $\sum_{k=1}^{\infty} a_k$ . Die  $n$ -te Partialsumme dieser Reihe lautet:  $\sum_{k=1}^n a_k$ . Öwir summieren unsere Reihe nur bis zum Endindex  $n$ .<sup>b</sup>

<sup>a</sup><https://www.lernhelfer.de/schuelerlexikon/mathematik/artikel/folgen-partialsummen>

<sup>b</sup><https://www.massmatics.de/merkzettel/index.php#!164:Partialsummen>

sos steht für *Summe of Squares*

| n  | fib <sub>n</sub> | fib <sub>n</sub> <sup>2</sup> |                                                   | $\sum_{k=1}^n \text{fib}^k$ |
|----|------------------|-------------------------------|---------------------------------------------------|-----------------------------|
| 1  | 1                | 1                             | 1                                                 | 1                           |
| 2  | 1                | 1                             | 1 + 1                                             | 2                           |
| 3  | 2                | 4                             | 1 + 1 + 4                                         | 6                           |
| 4  | 3                | 9                             | 1 + 1 + 4 + 9                                     | 15                          |
| 5  | 5                | 25                            | 1 + 1 + 4 + 9 + 25                                | 40                          |
| 6  | 8                | 64                            | 1 + 1 + 4 + 9 + 25 + 64                           | 104                         |
| 7  | 13               | 169                           | 1 + 1 + 4 + 9 + 25 + 64 + 169                     | 273                         |
| 8  | 21               | 441                           | 1 + 1 + 4 + 9 + 25 + 64 + 169 + 441               | 714                         |
| 9  | 34               | 1156                          | 1 + 1 + 4 + 9 + 25 + 64 + 169 + 441 + 1156        | 1870                        |
| 10 | 55               | 3025                          | 1 + 1 + 4 + 9 + 25 + 64 + 169 + 441 + 1156 + 3025 | 4895                        |

- (a) Implementieren Sie die obigen Formeln zunächst rekursiv (ohne Schleifenkonstrukte wie `for` oder `while`) und ohne weitere Optimierungen („naiv“) in Java als:

```
long fibNaive (int n) {
```

bzw.

```
long sosNaive (int n) {
```

Lösungsvorschlag

```
public static long fibNaive(int n) {
 if (n <= 2) {
 return 1;
 }
 return fibNaive(n - 1) + fibNaive(n - 2);
}

public static long sosNaive(int n) {
```

```

 if (n <= 1) {
 return fibNaive(n);
 }
 return fibNaive(n) * fibNaive(n) + sosNaive(n - 1);
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2017/fruehjahr/Fibonacci.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/Fibonacci.java)

- (b) Offensichtlich ist die naive Umsetzung extrem ineffizient, da viele Zwischenergebnisse wiederholt rekursiv ausgewertet werden müssen. Die Dynamische Programmierung (DP) erlaubt es Ihnen, die Laufzeit auf Kosten des Speicherbedarfs zu reduzieren, indem Sie alle einmal berechneten Zwischenergebnisse speichern und bei erneutem Bedarf „direkt abrufen“. Implementieren Sie obige Formeln nun rekursiv aber mittels DP in Java als:

```
long fibDP (int n) {
```

bzw.

```
long sosDP (int n) {
```

Lösungsvorschlag

```

public static long fibDP(int n) {
 // Nachschauen, ob die Fibonacci-Zahl bereits berechnet wurde.
 if (fib[n] != 0) {
 return fib[n];
 }
 // Die Fibonacci-Zahl neu berechnen.
 if (n <= 2) {
 fib[n] = 1;
 } else {
 fib[n] = fibDP(n - 1) + fibDP(n - 2);
 }
 return fib[n];
}

public static long sosDP(int n) {
 // Nachschauen, ob die Quadratsumme bereits berechnet wurde.
 if (sos[n] != 0) {
 return sos[n];
 }
 // Die Quadratsumme neu berechnen.
 if (n <= 1) {
 sos[n] = fibDP(n);
 } else {
 long tmp = fibDP(n);
 sos[n] = tmp * tmp + sosDP(n - 1);
 }
 return sos[n];
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2017/fruehjahr/Fibonacci.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/Fibonacci.java)

- (c) Am „einfachsten“ und bzgl. Laufzeit [in  $\mathcal{O}(n)$ ] sowie Speicherbedarf [in  $\mathcal{O}(1)$ ]

am effizientesten ist sicherlich eine iterative Implementierung der beiden Formeln. Geben Sie eine solche in Java an als:

```
long fibIter (int n) {
```

bzw.

```
long sosIter (int n) {
```

Lösungsvorschlag

```
public static long fibIter(int n) {
 long a = 1;
 long b = 1;
 for (int i = 2; i < n; i++) {
 long tmp = a + b;
 b = a;
 a = tmp;
 }
 return a;
}

public static long sosIter(int n) {
 long a = 1;
 long b = 0;
 long sosSum = 1;
 for (int i = 2; i <= n; i++) {
 long tmp = a + b;
 b = a;
 a = tmp;
 sosSum += a * a;
 }
 return sosSum;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2017/fruehjahr/Fibonacci.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/Fibonacci.java)

## 66115 / 2017 / Frühjahr / Thema 1 / Aufgabe 4

Sie dürfen im Folgenden davon ausgehen, dass keinerlei Under- oder Overflows auftreten.

Gegeben sei folgende rekursive Methode für  $n \geq 0$ :

```
long sumOfSquares (long n) {
 if (n == 0)
 return 0;
 else
 return n * n + sumOfSquares(n - 1);
}
```

(a) Beweisen Sie formal mittels vollständiger Induktion:

$$\forall n \in \mathbb{N} : \text{sumOfSquares}(n) = \frac{n(n+1)(2n+1)}{6}$$

Sei  $f(n) : \frac{n(n+1)(2n+1)}{6}$

### Induktionsanfang

— Beweise, dass  $A(1)$  eine wahre Aussage ist. \_\_\_\_\_

Für  $n = 0$  gilt:

$$\text{sumOfSquares}(0) \stackrel{\text{if}}{=} 0 = f(0)$$

### Induktionsvoraussetzung

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . \_\_\_\_\_

Für ein festes  $n \in \mathbb{N}$  gelte:

$$\text{sumOfSquares}(n) = f(n)$$

### Induktionsschritt

— Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss.

$$n \rightarrow n + 1$$

$$\begin{aligned}
f(n+1) &= \text{sumOfSquares}(n+1) && \text{Java-Methode eingesetzt} \\
&\stackrel{\text{else}}{=} (n+1) * (n+1) + \text{sumOfSquares}(n) && \text{Java-Code der else-Verzweigung verwendet} \\
&\stackrel{\text{I.H.}}{=} (n+1)(n+1) + f(n) && \text{mathematisch notiert} \\
&= (n+1)(n+1) + \frac{n(n+1)(2n+1)}{6} && \text{Formel eingesetzt} \\
&= (n+1)^2 + \frac{n(n+1)(2n+1)}{6} && \text{potenziert} \\
&= \frac{6(n+1)^2}{6} + \frac{n(n+1)(2n+1)}{6} && (n+1)^2 \text{ in Bruch umgewandelt} \\
&= \frac{6(n+1)^2 + n(n+1)(2n+1)}{6} && \text{Addition gleichnamiger Brüche} \\
&= \frac{(n+1)6(n+1) + (n+1)n(2n+1)}{6} && n+1 \text{ ausklammern vorbereitet} \\
&= \frac{(n+1)(6(n+1) + n(2n+1))}{6} && n+1 \text{ ausgeklammert} \\
&= \frac{(n+1)(6n+6+2n^2+n)}{6} && \text{Klammern ausmultipliziert / aufgelöst} \\
&= \frac{(n+1)(2n^2+7n+6)}{6} && \text{umsortiert, addiert } 6n+n=7n \\
&= \frac{(n+1)(2n^2+3n+4n+6)}{6} && \text{Ausklammern vorbereitet} \\
&= \frac{(n+1)(n+2)(2n+3)}{6} && (n+2) \text{ ausgeklammert} \\
&= \frac{(n+1)((n+1)+1)(2(n+1)+1)}{6} && (n+1) \text{ verdeutlicht}
\end{aligned}$$

a

<sup>a</sup>[https://mathcs.org/analysis/reals/infinity/answers/sm\\_sq\\_cb.html](https://mathcs.org/analysis/reals/infinity/answers/sm_sq_cb.html)

(b) Beweisen Sie die Terminierung von `sumOfSquares(n)` für alle  $n \geq 0$ .

Lösungsvorschlag

Sei  $T(n) = n$ . Die Funktion  $T(n)$  ist offenbar ganzzahlig. In jedem Rekursionsschritt wird  $n$  um eins verringert, somit ist  $T(n)$  streng monoton fallend. Durch die Abbruchbedingung `n==0` ist  $T(n)$  insbesondere nach unten beschränkt. Somit ist  $T$  eine gültige Terminierungsfunktion.

## 66115 / 2017 / Frühjahr / Thema 1 / Aufgabe 5

Zeigen oder widerlegen Sie die folgenden Aussagen (die jeweiligen Beweise sind sehr kurz):

- (a) Alle regulären Sprachen liegen in NP.

Lösungsvorschlag

Stimmt. Alle regulären Sprachen sind in Polynomialzeit entscheidbar (es existiert ein Automat dazu), sie liegen also in P und folglich auch in NP.

- (b) Es gibt Sprachen  $A, B$  mit  $A \subseteq B$ , sodass  $B$  regulär und  $A$  kontextfrei ist.

Lösungsvorschlag

Stimmt. Es existieren Sprachen mit der Eigenschaft wie gefordert. Wir wählen:  $B = (a|b)^*$  und  $A = \{a^n b^n \mid n \in \mathbb{N}\}$ .  $A$  ist bekanntermaßen nicht regulär, wie man mit dem Pumping Lemma beweisen kann, kann aber durch eine Grammatik  $G = (V, \{a, b\}, \{S \rightarrow aSb \mid \varepsilon\}, S)$  erzeugt werden. Für  $B$  gibt es einen deterministischen endlichen Automaten.

- (c) Es gibt unentscheidbare Sprachen  $L$  über dem Alphabet  $\Sigma$ , so dass sowohl  $L$  als auch das Komplement  $\bar{L} = \Sigma^* \setminus L$  rekursiv aufzählbar (= partiell entscheidbar) sind.

Lösungsvorschlag

Stimmt nicht. Ist  $L$  und sein Komplement rekursiv aufzählbar, so können wir  $L$  entscheiden, denn wir haben eine Maschine, die auf Eingabe  $x$  hält und akzeptiert, wenn  $x \in L$  ist, sowie eine Maschine, die hält, wenn  $x$  nicht akzeptiert, wenn  $x \notin L$  ist. Daraus lässt sich eine Maschine konstruieren, die  $L$  entscheidet.

- (d) Sei  $L$  eine beliebige kontextfreie Sprache über dem Alphabet  $\Sigma$ . Dann ist das Komplement  $\bar{L} = \Sigma^* \setminus L$  entscheidbar.

Lösungsvorschlag

Stimmt. Es gibt einen Entscheider für die Sprache  $L$ . Dieser entscheidet für eine Eingabe  $x$ , ob diese in  $L$  ist oder nicht. Negiert man diese Entscheidung, so ergibt sich ein Entscheider für  $\bar{L}$ .

Schreiben Sie zuerst zur Aussage „Stimmt“ oder „Stimmt nicht“ und dann Ihre Begründung.

## 66115 / 2017 / Frühjahr / Thema 1 / Aufgabe 6

Es sei  $E$  die Menge aller (geeignet codierten) Turingmaschinen  $M$  mit folgender Eigenschaft: Es gibt eine Eingabe  $w$ , so dass  $M$  gestartet auf  $w$  mindestens 1000 Schritte rechnet und dann irgendwann hält.

Das Halteproblem auf leerer Eingabe  $H_0$  ist definiert als die Menge aller Turingmaschinen, die auf leerer Eingabe gestartet, irgendwann halten.

- (a) Zeigen Sie, dass  $E$  unentscheidbar ist (etwa durch Reduktion vom Halteproblem  $H_0$ ).

zu zeigen:  $L_H \leq L \rightarrow L$  ist genauso unentscheidbar wie  $L_H$

Eingabeinstanzen von  $L_H(TM(M), u)$  durch Funktion umbauen in Eingabeinstanzen von  $L(TM(M'))$ .

Idee: Turingmaschine so modifizieren, dass sie zunächst 1000 Schritte macht und dann  $M$  auf  $u$  startet.

Dazu definieren wir die Funktion  $f : \Sigma^* \rightarrow \Sigma^*$  wie folgt:

$$f(u) = \begin{cases} c(M') & \text{falls } u = c(M')w \text{ ist für eine Turingmaschine } M \text{ und Eingabe } w \\ 0 & \text{sonst} \end{cases}$$

Dabei sei  $M'$  eine Turingmaschine, die sich wie folgt verhält:

- (i) Geht 1000 Schritte nach rechts
- (ii) Schreibt festes Wort  $w$  (für  $M'$  ist  $w$  demnach fest!)
- (iii) Startet  $M$

**total:** ja

**berechenbar:** Syntaxcheck, 1000 Schritte über 1000 weitere Zustände realisierbar

**Korrektheit:**  $u \in L_{halt} \Leftrightarrow u = c(M)w$  für TM  $M$ , die auf  $w$  hält  $\Leftrightarrow f(u) = c(M')$ , wobei  $M'$  1000 Schritte macht und dann hält  $\Leftrightarrow f(u) \in L$

(b) Begründen Sie, dass  $E$  partiell entscheidbar ist.

(c) Geben Sie ein Problem an, welches nicht einmal partiell entscheidbar ist.

## 66115 / 2017 / Frühjahr / Thema 2 / Aufgabe 2

(a) Gegeben sei die kontextfreie Grammatik  $G = (V, \Sigma, P, S)$  mit Sprache  $L(G)$ , wobei  $V = S, T, U$  und  $\Sigma = \{a, b, c, d, e\}$ .  $P$  bestehe aus den folgenden Produktionen:

$$P = \left\{ \begin{array}{l} S \rightarrow U \mid SbU \\ T \rightarrow dSe \mid a \\ U \rightarrow T \mid UcT \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gib25c5oc



(i) Zeigen Sie  $acdae \in L(G)$ .

Lösungsvorschlag

$$S \vdash U \vdash UcT \vdash TcT \vdash acT \vdash acdSe \vdash acdUe \vdash acdae$$

(ii) Bringen Sie  $G$  in Chomsky-Normalform.

Lösungsvorschlag

**i. Elimination der  $\varepsilon$ -Regeln**

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen.

$\emptyset$  Nichts zu tun

**ii. Elimination von Kettenregeln**

— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.

$$P = \left\{ \begin{array}{l} S \rightarrow dSe \mid a \mid UcT \mid SbU \\ T \rightarrow dSe \mid a \\ U \rightarrow dSe \mid a \mid UcT \end{array} \right\}$$

**iii. Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt.

$$P = \left\{ \begin{array}{l} S \rightarrow DSE \mid a \mid UCT \mid SBU \\ T \rightarrow DSE \mid a \\ U \rightarrow DSE \mid a \mid UCT \\ B \rightarrow b \\ C \rightarrow c \\ D \rightarrow d \\ E \rightarrow e \end{array} \right\}$$

**iv. Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1}B_n, A_{n-1} \rightarrow A_{n-2}B_{n-1}, \dots, A_2 \rightarrow B_1B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht.

$$P = \left\{ \right.$$

$$\begin{aligned}
S &\rightarrow DS_E \mid a \mid UC_T \mid SB_U \\
T &\rightarrow DS_E \mid a \\
U &\rightarrow DS_E \mid a \mid UC_T \\
B &\rightarrow b \\
C &\rightarrow c \\
D &\rightarrow d \\
E &\rightarrow e \\
S_E &\rightarrow SE \\
C_T &\rightarrow CT \\
B_U &\rightarrow BU
\end{aligned}$$

Pumping-Lemma  
(Kontextfreie Sprache)  
Berechenbarkeit

}

- (b) Geben Sie eine kontextfreie Grammatik für  $L = \{a^i b^k c^i \mid i, k \in \mathbb{N} \mid a\}^n$ .

Lösungsvorschlag

Wir interpretieren  $\mathbb{N}$  als  $\mathbb{N}_0$ .

$$P = \{$$

$$S \rightarrow aSc \mid aBc \mid B \mid \varepsilon B \quad \rightarrow b \mid Bb$$

}

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ghp3bftg

- (c) Zeigen Sie, dass  $L = \{a^i b^k c^i \mid i, k \in \mathbb{N} \wedge i < k \mid n\}$  nicht kontextfrei ist, indem Sie das Pumping-Lemma für kontextfreie Sprachen anwenden.

#### Exkurs: Pumping-Lemma für Reguläre Sprachen

Es sei  $L$  eine kontextfreie Sprache. Dann gibt es eine Zahl  $j$ , sodass sich alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  zerlegen lassen in  $\omega = uvwxy$ , sodass die folgenden Eigenschaften erfüllt sind:

- (i)  $|vx| \geq 1$  (Die Wörter  $v$  und  $x$  sind nicht leer.)
- (ii)  $|vwx| \leq j$  (Die Wörter  $v$ ,  $w$  und  $x$  haben zusammen höchstens die Länge  $j$ .)
- (iii) Für alle  $i \in \mathbb{N}_0$  gilt  $uv^iwx^iy \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^iwx^iy$  in der Sprache  $L$ )

Lösungsvorschlag

## 66115 / 2017 / Frühjahr / Thema 2 / Aufgabe 3

- (a) Primitiv rekursive Funktionen

- (i) Zeigen Sie, dass die folgendermaßen definierte Funktion  $if: \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  primitiv rekursiv ist.

sonst

- (ii) Wir nehmen eine primitiv rekursive Funktion  $p: \mathbb{N} \rightarrow \mathbb{N}$  an und definieren  $g(n)$  als die Funktion, welche die größte Zahl  $i < n$  zurückliefert, für die  $p(i) = 0$  gilt. Falls kein solches  $i$  existiert, soll  $g(n) = 0$  gelten:

$$a(n) = \max \{ i < n \mid p(i) = 0 \} \cup \{0\}$$

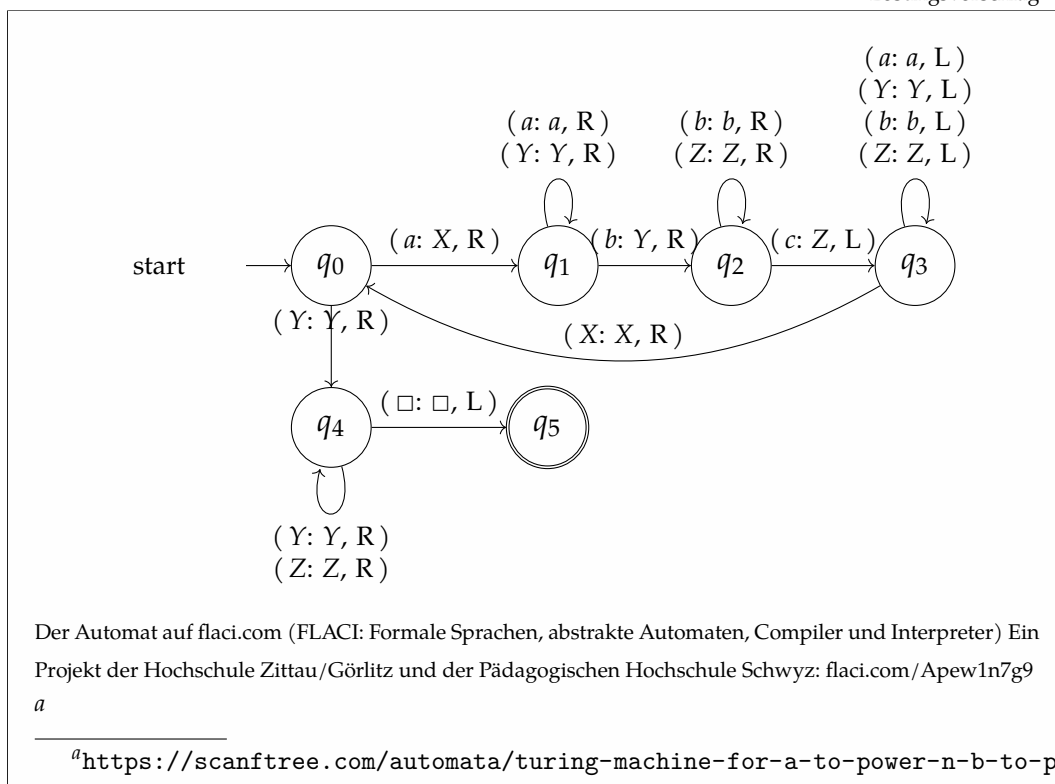
$$if(b, x, y) = ( \text{ falls } b=0$$

Zeigen Sie, dass  $g: \mathbb{N} \rightarrow \mathbb{N}$  primitiv rekursiv ist. (Sie dürfen obige Funktion  $if$  als primitiv rekursiv voraussetzen.)

- (b) Sei  $\Sigma = \{a, b, c\}$  und  $L \subseteq \Sigma^*$  mit  $L = \{a^i b^j c^i \mid i \in \mathbb{N}\}$ .

- (i) Beschreiben Sie eine Turingmaschine, welche die Sprache  $Z$  entscheidet. Eine textuelle Beschreibung der Konstruktionsidee ist ausreichend.

Lösungsvorschlag



- (ii) Geben Sie Zeit- und Speicherkomplexität (abhängig von der Länge der Eingabe) Ihrer Turingmaschine an.

**Speicherkomplexität**  $n$  (Das Eingabewort wird einmal überschrieben)

**Zeitkomplexität** the turing machine time complexity is the number of transition execution will executed is call time complexity of the turing machine. first we start we main loop execution is  $(n/3)-1$ . transition  $(a,x,R)$  from state 1 to 2 = 1. transition  $(a,a,R)$  and  $(y,y,R)$  on

state 2 is  $= (n/3)-1$ . transition  $(b,y,R)$  from state 2 to  $3=1$ . on state 3  $(b,b,R)$  and  $(z,z,R)=(n/3)-1$ . transition  $(c,z,L)$  from state 3 to  $4=1$ . on state 4  $(y,y,L),(b,b,L),(z,z,L)$  and state 5  $(a,a,L)=(n/3)-1$ . transition  $(a,a,L)$  from state 4 to  $5=1$ . transition  $(x,x,R)$  from 5 to  $1=1$  total  $(n+2)$  following transition will executed transition  $(a,x,R)$  from state 1 to  $2=1$ . transition  $(y,y,R)$  on state 2 is  $= (n/3)-1$ . transition  $(b,y,R)$  from state 2 to  $3=1$ . transition  $(z,z,R)$  on state 3  $= (n/3)-1$  transition  $(c,z,L)$  from state 3 to  $4=1$ . on state 4  $(y,y,L),(z,z,L)$  and state  $(n/3)-1$ . transition  $(x,x,R)$  from state 5 to  $6=1$  transition on state 6  $(y,y,R),(z,z,R)=(n/3)$  transition  $(d,d,R)$  from state 6 to  $7=1$  total  $= (4n/3)+2$  over alti time complexity  $(n+2)(n/3)-1 + (4n/3)+2$   
<sup>a</sup>

<sup>a</sup>[https://www.youtube.com/watch?v=vwnz9e\\_Lrfo](https://www.youtube.com/watch?v=vwnz9e_Lrfo)

(c) Sei  $\Sigma = \{0,1\}$ . Jedes  $w \in \Sigma^*$  kodiert eine Turingmaschine  $M_w$ . Die von  $M_w$  berechnete Funktion bezeichnen wir mit  $\varphi_w(x)$ .

- (i) Warum ist  $L = \{w \in \Sigma^* \mid \exists x: \varphi_w(x) = xx\}$  nicht entscheidbar?
- (ii) Warum ist  $L = \{w \in \Sigma^* \mid \exists x: w = xx\}$  entscheidbar?

## 66115 / 2017 / Herbst / Thema 1 / Aufgabe 2

Betrachten Sie die Sprache  $L_1 = L_a \cup L_b$ .

- $L_a = \{a^n b c^n \mid n \in \mathbb{N}\}$
- $L_b = \{a b^m c^m \mid m \in \mathbb{N}\}$

(a) Geben Sie für  $L_1$  eine kontextfreie Grammatik an.

Lösungsvorschlag

$$P = \left\{ \begin{array}{l} S \rightarrow S_a \mid S_b \\ S_a \rightarrow a S_a c \mid b \\ S_b \rightarrow a \mid a B_b \\ B_b \rightarrow b B_b c \mid bc \end{array} \right\}$$

(b) Ist Ihre Grammatik aus a) eindeutig? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Nein. Die Sprache ist nicht eindeutig. Für das Wort  $abc$  gibt es zwei Ableitungen, nämlich  $S \vdash S_a \vdash a S_a c \vdash abc$  und  $S \vdash S_b \vdash a B_b \vdash abc$ .

- (c) Betrachten Sie die Sprache  $L_2 = \{ a^{2^n} \mid n \in \mathbb{N} \}$ . Zeigen Sie, dass  $L_2$  nicht kontextfrei ist.

Lösungsvorschlag

Annahme:  $L_2$  ist kontextfrei

→ Pumping-Lemma gilt für  $L_2$

→  $j \in \mathbb{N}$  als Pumping-Zahl

$\omega \in L_2: |\omega| \geq j$

Konsequenz:  $\omega = uvwxy$

- $|vx| \geq 1$
- $|vwx| \leq j$
- $uv^iwx^iy \in L_2$  für alle  $i \in \mathbb{N}_0$

Wir wählen:  $\omega = a^{2^i}: |\omega| \geq j$

**p**  $a \dots a$

**r**  $a \dots a$

**s**  $a \dots a$

**t**  $a \dots a$

**q**  $a \dots a$

$$q + r + s + t + q = 2^j$$

$$\Rightarrow r + t \geq 1$$

$$r + s + t \leq j$$

### 1. Fall

$$r + t = 2^{j-1}$$

$$2^{j-1} + 2^{j-1} = 2 \cdot 2^{j-1} = 2^1 \cdot 2^{j-1} = 2^{1+j-1} = 2^j$$

$$\omega' = uv^2wx^2y$$

$$p + 2 \cdot r + s + 2 \cdot t + q$$

$$p + s + q + 2 \cdot (r + t)$$

$$2^{j-1} + 2 \cdot 2^{j-1} = 3 \cdot 2^{j-1} = 2^{j-1} + 2^j \leq 2^{j+1}$$

keine Zweierpotenz

$$\Rightarrow \omega \notin L_2$$

⇒ Widerspruch zur Annahme

⇒  $L_2$  nicht kontextfrei

**2. Fall**

$$r + t \neq 2^{j-1}$$

$$\omega' = uv^0wx^0y$$

$$\Rightarrow p + s + q = 2^j - (r + t)$$

$$(r + t) \neq 2^{j-i}$$

ist keine Zweierpotenz

$$\Rightarrow \omega \notin L_2$$

$$\Rightarrow L_2 \text{ nicht kontextfrei}$$

**66115 / 2017 / Herbst / Thema 1 / Aufgabe 3**

Betrachten Sie die folgenden Probleme:

**Exkurs: SAT**

Das **Erfüllbarkeitsproblem der Aussagenlogik** SAT und  $\kappa$ -SAT mit  $k \geq 3, k \in \mathbb{N}$  (Satz von Cook) fragt, ob eine aussagenlogische Formel erfüllbar ist. Das Erfüllbarkeitsproblem der *Aussagenlogik* ist in exponentieller Zeit in Abhängigkeit der Anzahl der Variablen mit Hilfe einer Wahrheitstabelle entscheidbar. Diese *Wahrheitstabelle* kann nicht in polynomieller Zeit aufgestellt werden.

**Exkurs: NAE3SAT**

Like 3-satisfiability, an instance of the problem consists of a collection of Boolean variables and a collection of clauses, each of which combines three variables or negations of variables. However, unlike 3-satisfiability, which requires each clause to have at least one true Boolean value, NAE3SAT requires that the three values in each clause are not all equal to each other (in other words, at least one is true, and at least one is false) <sup>a</sup>

<sup>a</sup>[https://en.wikipedia.org/wiki/Not-all-equal\\_3-satisfiability](https://en.wikipedia.org/wiki/Not-all-equal_3-satisfiability)

**3SAT**

**Gegeben:** Eine aussagenlogische Formel  $\varphi$  in konjunktiver Normalform (drei Literale pro Klausel).

**Frage:** Ist  $\varphi$  erfüllbar?

**NAE-3SAT**

**Gegeben:** Eine aussagenlogische Formel  $\varphi$  in konjunktiver Normalform (drei Literale pro Klausel).

**Frage:** Gibt es eine Belegung, die in jeder Klausel mindestens ein Literal *wahr* und mindestens ein Literal *falsch* macht?

Wir erlauben, dass NAE-3SAT-Formeln Literale der Form *false* haben, die immer *falsch* sind. So ist

$$(x_1 \vee \text{false} \vee \text{false}) \wedge (\neg x_1 \vee x_1 \vee x_1)$$

in NAE-3SAT (setze  $x_1$  wahr).

- (a) Zeigen Sie, dass sich 3SAT in polynomieller Zeit auf NAE-3SAT reduzieren lässt.

Lösungsvorschlag

- (b) Was können Sie aus a) folgern, wenn Sie wissen, dass 3SAT NP-vollständig ist?

Lösungsvorschlag

- (c) Was können Sie aus a) folgern, wenn Sie wissen, dass NAE-3SAT NF-vollständig ist?

Lösungsvorschlag

## 66115 / 2017 / Herbst / Thema 1 / Aufgabe 8

Sei  $X = (I_1, I_2, \dots, I_n)$  eine Menge von  $n$  (geschlossenen) Intervallen über den reellen Zahlen  $\mathbb{R}$ . Das Intervall  $I_j$  sei dabei gegeben durch seine linke Intervallgrenze  $l_j \in \mathbb{R}$  sowie seine rechte Intervallgrenze  $r_j \in \mathbb{R}$  mit  $r_j > l_j$ ,  $I_j = [l_j, r_j]$ .

Wir nehmen in dieser Aufgabe der Einfachheit halber an, dass die Zahlen alle paarweise verschieden sind.

Zwei Intervalle  $I_j, I_k$  überlappen sich gdw. sie mindestens einen Punkt gemeinsam haben, d.h. falls für (o.B.d.A.)  $l_j < r_k$ , auch  $l_k < r_j$  gilt. Eine gültige Färbung von  $X$  mit  $c \in \mathbb{N}$  Farben ist eine Funktion  $F : X \rightarrow \{1, 2, \dots, c\}$  mit der Eigenschaft, dass für jedes Paar  $I_j, I_k$  von überlappenden Intervallen  $F(I_j) \neq F(I_k)$  gilt.

Abbildung 1: Eine gültige Färbung von  $X$

Eine minimale gültige Färbung von  $X$  ist eine gültige Färbung mit einer minimalen Anzahl an Farben. Die Anzahl von Farben in einer minimalen gültigen Färbung von  $X$  bezeichnen wir mit  $\chi(X)$ . Wir gehen im Folgenden davon aus, dass für  $X$  eine minimale gültige Färbung  $F^*$  gefunden wurde.

- (a) Nehmen wir an, dass aus  $X$  alle Intervalle einer bestimmten Farbe von  $F^*$  gelöscht werden. Ist die so aus  $F^*$  entstandene Färbung der übrigen Intervalle in jedem Fall immer noch eine minimale gültige Färbung? Begründen Sie Ihre Antwort.
- (b) Nehmen wir an, dass aus  $X$  ein beliebiges Intervall gelöscht wird. Ist die so aus  $F^*$  entstehende Färbung der übrigen Intervalle in jedem Fall immer noch eine minimale gültige Färbung? Begründen Sie Ihre Antwort.
- (c) Mit  $u_j(X)$  bezeichnen wir die maximale Anzahl von Intervallen in  $X$ , die sich paarweise überlappen. Zeigen Sie, dass  $\chi(X) \geq u_j(X)$  ist. Wir betrachten nun folgenden Algorithmus, der die Menge  $X = (I_1, I_2, \dots, I_n)$  von  $n$  Intervallen einfärbt:



- Zunächst sortieren wir die Intervalle von  $X$  aufsteigend nach ihren linken Intervallgrenzen. Die Intervalle werden jetzt in dieser Reihenfolge nacheinander eingefärbt; ist ein Intervall dabei erst einmal eingefärbt, ändert sich seine Farbe nie wieder. Angenommen die sortierte Reihenfolge der Intervalle sei  $I_a(i), \blacksquare \blacksquare \blacksquare, F(n)$ -
  - Das erste Intervall  $F(i)$  erhält die Farbe 1. Für  $1 < i < n$  verfahren wir im  $i$ -ten Schritt zum Färben des  $i$ -ten Intervalls wie folgt:  
Bestimme die Menge  $C_j$  aller Farben der bisher schon eingefärbten Intervalle die  $I_i$  überlappen. Färbe  $I_i$  dann mit der Farbe  $c_i = \min(\{1, 2, \dots, n\} \setminus C_j)$ .  
Fortsetzung nächste Seite!
- (d) Begründen Sie, warum der Algorithmus immer eine gültige Färbung von  $X$  findet (Hinweis: Induktion).
- (e) Zeigen Sie, dass die Anzahl an Farben, die der Algorithmus für das Einfärben benötigt, mindestens  $\chi(X)$  ist.
- (f) Zeigen Sie, dass die Anzahl an Farben, die der Algorithmus für das Einfärben benötigt, höchstens  $\chi(X)$  ist.
- (g) Begründen Sie mit Hilfe der o.g. Eigenschaften, warum der Algorithmus korrekt ist, d.h. immer eine minimale gültige Färbung von  $X$  findet.
- (h) Wir betrachten folgende Implementierung des Algorithmus in Pseudocode:  
Was ist die asymptotische Laufzeit dieses Algorithmus? Was ist der asymptotische Speicherbedarf dieses Algorithmus? Begründen Sie Ihre Antworten.

## 66115 / 2017 / Herbst / Thema 2 / Aufgabe 5

Sei  $G = (\{S, A, B, C\}, \{a, b\}, P, S)$  die kontextfreie Grammatik in Chomsky-Normalform und der Menge  $P$  der Produktionen:

$$P = \left\{ \begin{array}{l} S \rightarrow AB \mid BC \\ A \rightarrow BA \mid a \\ C \rightarrow AB \mid a \\ B \rightarrow CC \mid b \end{array} \right\}$$

Sei  $\omega = baaab$ . Folgende Tabelle entsteht durch Anwendung des CYK-Algorithmus. Z. B. bedeutet  $B \in V(3, 5)$ , dass aus der Variablen  $B$  das Teilwort  $\omega_3\omega_4\omega_5 = aab$  hergeleitet werden kann. Drei Einträge wurden weggelassen.

- (a) Bestimmen Sie die Mengen  $V(1, 2)$ ,  $V(1, 3)$  und  $V(1, 5)$ .

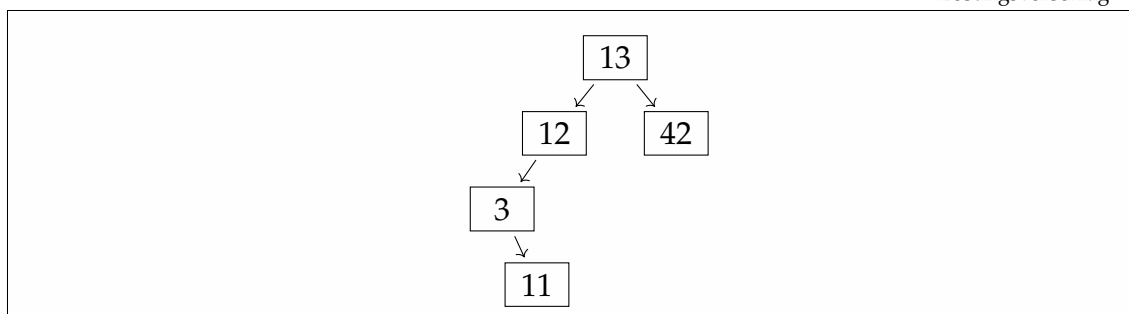
|       |       |     |     |   |
|-------|-------|-----|-----|---|
| b     | a     | a   | a   | b |
| B     | A,C   | A,C | A,C | B |
| A,S   | B     | B   | S,C |   |
| -     | S,C,A | B   |     |   |
| S,A,C | S,C   |     |     |   |
| S,C   |       |     |     |   |

- (b) Wie entnehmen Sie dieser Tabelle, dass  $\omega \in L(G)$  ist?

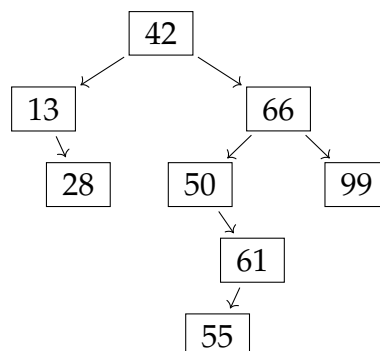
In der Menge  $V(1,5)$  ist das Startsymbol  $S$  der Sprache  $L(G)$  enthalten.

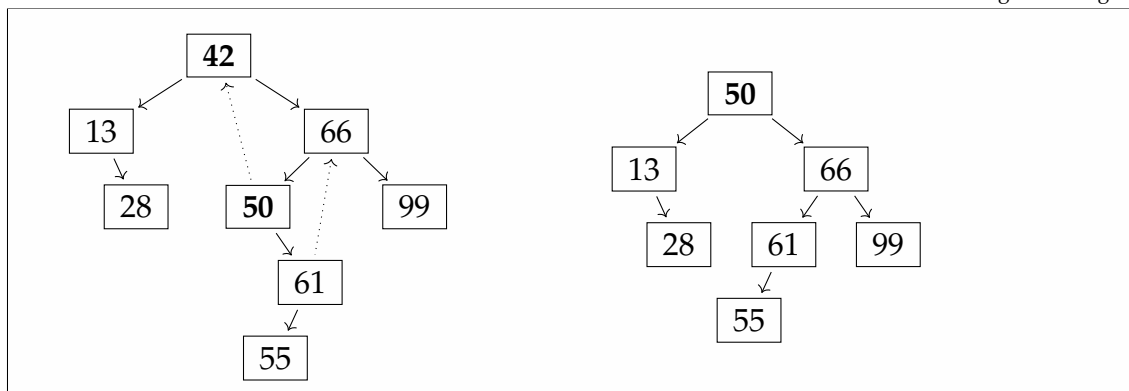
## 66115 / 2017 / Herbst / Thema 2 / Aufgabe 8

- (a) Fügen Sie die Zahlen 13, 12, 42, 3, 11 in der gegebenen Reihenfolge in einen zunächst leeren binären Suchbaum mit aufsteigender Sortierung ein. Stellen Sie nur das Endergebnis dar.

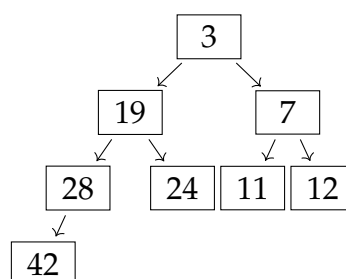


- (b) Löschen Sie den Wurzelknoten mit Wert 42 aus dem folgenden *binären* Suchbaum mit aufsteigender Sortierung und ersetzen Sie ihn dabei durch einen geeigneten Wert aus dem *rechten* Teilbaum. Lassen Sie möglichst viele Teilbäume unverändert und erhalten Sie die Suchbaumeigenschaft.

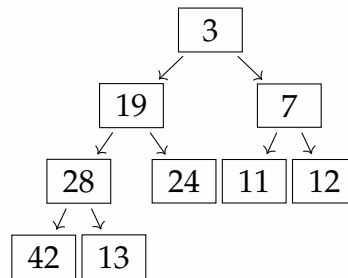




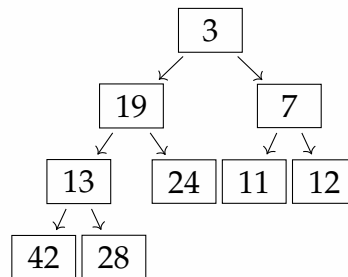
- (c) Fügen Sie einen neuen Knoten mit dem Wert 13 in die folgende Min-Halde ein und stellen Sie anschließend die Halden-Eigenschaft vom neuen Blatt aus beginnend wieder her, wobei möglichst viele Knoten der Halde unverändert bleiben und die Halde zu jedem Zeitpunkt links-vollständig sein soll. Geben Sie nur das Endergebnis an.



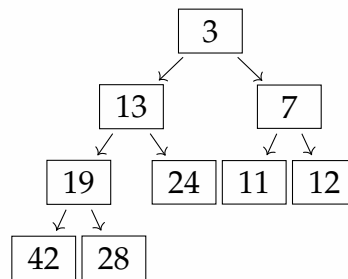
*Nach dem Einfügen von „13“:*



*Nach dem Vertauschen von „13“ und „28“:*



Nach dem Vertauschen von „13“ und „19“:

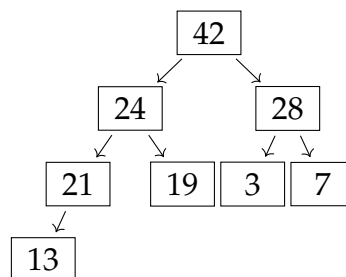


- (d) Geben Sie für die ursprüngliche Min-Halde aus Teilaufgabe c) (lösche den neu eingefügten Knoten mit dem Wert 13) die Feld-Einbettung (Array-Darstellung) an.

Lösungsvorschlag

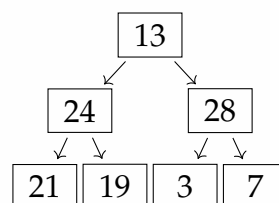
| 0 | 1  | 2 | 3  | 4  | 5  | 6  | 7  |
|---|----|---|----|----|----|----|----|
| 3 | 19 | 7 | 28 | 24 | 11 | 12 | 42 |

- (e) Löschen Sie den Wurzelknoten mit Wert 42 aus der folgenden Max-Halde und stellen Sie anschließend die Halden-Eigenschaft ausgehend von einer neuen Wurzel wieder her, wobei möglichst viele Knoten der Halde unverändert bleiben und die Halde zu jedem Zeitpunkt links-vollständig sein soll. Geben Sie nur das Endergebnis an.

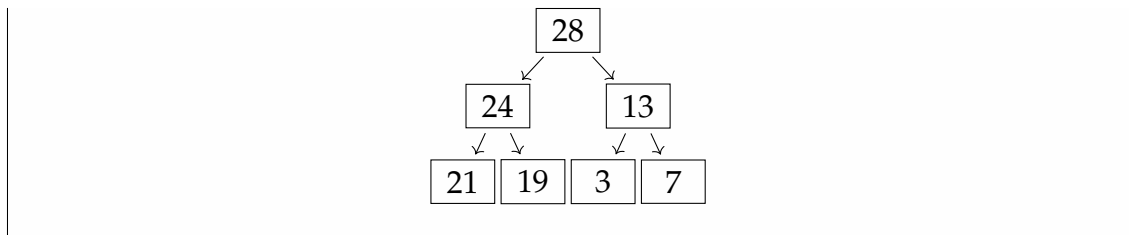


Lösungsvorschlag

Nach dem Ersetzen von „42“ mit „13“:

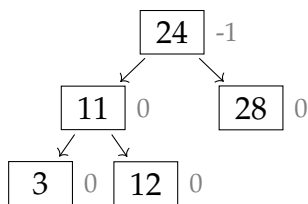


Nach dem Vertauschen von „13“ und „28“:



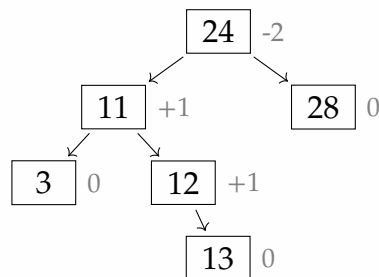
- (f) Fügen Sie in jeden der folgenden AVL-Bäume mit aufsteigender Sortierung jeweils einen neuen Knoten mit dem Wert 13 ein und führen Sie anschließend bei Bedarf die erforderliche(n) Rotation(en) durch. Zeichnen Sie den Baum vor und nach den Rotationen.

(i) AVL-Baum A

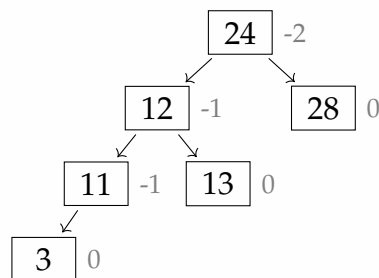


Lösungsvorschlag

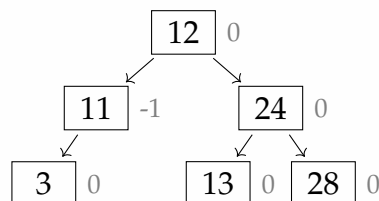
Nach dem Einfügen von „13“:



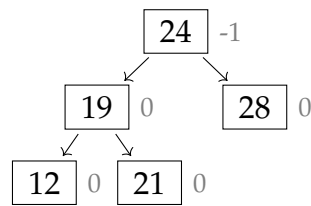
Nach der Linksrotation:



Nach der Rechtsrotation:

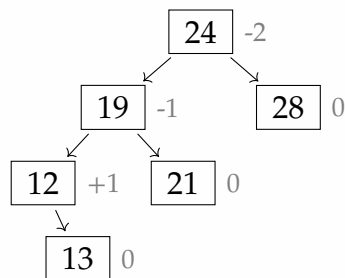


## (ii) AVL-Baum B

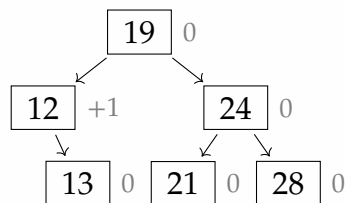


Lösungsvorschlag

Nach dem Einfügen von „13“:



Nach der Rechtsrotation:

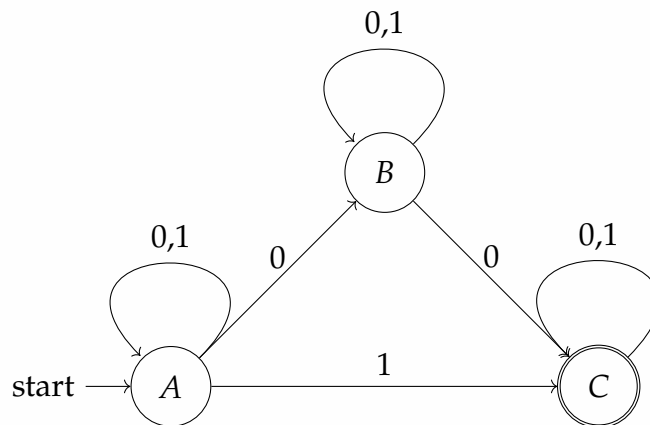


## 66115 / 2018 / Frühjahr / Thema 1 / Aufgabe 1

Gegeben ist der nichtdeterministische endliche Automat (NEA)  $(\{0, 1\}, Q, \delta, q_0, F)$ , wobei  $Q = \{A, B, C\}$ ,  $q_0 = A$ ,  $F = \{C\}$  und

| $\delta$ | 0          | 1          |
|----------|------------|------------|
| A        | $\{A, B\}$ | $\{A, C\}$ |
| B        | $\{B, C\}$ | $\{B\}$    |
| C        | $\{C\}$    | $\{C\}$    |

Lösungsvorschlag



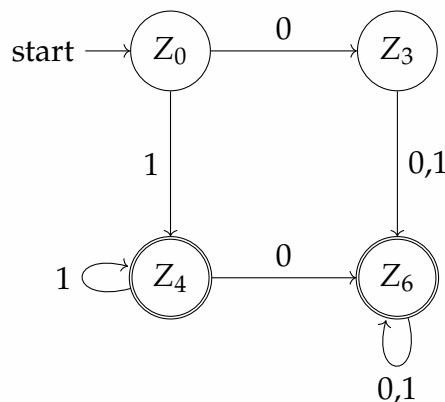
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aiq3xxgi9

- (a) Führen Sie für diesen NEA die Potenzmengenkonstruktion durch; geben Sie alle acht entstehenden Zustände mit ihren Transitionen an, nicht nur die erreichbaren.

Lösungsvorschlag

nicht erreichbar

| Name  | Zustandsmenge    | Eingabe 0        | Eingabe 1        |
|-------|------------------|------------------|------------------|
| $Z_0$ | $Z_0\{A\}$       | $Z_3\{A, B\}$    | $Z_4\{A, C\}$    |
| $Z_1$ | $Z_1\{B\}$       | $Z_5\{B, C\}$    | $Z_1\{B\}$       |
| $Z_2$ | $Z_2\{C\}$       | $Z_2\{C\}$       | $Z_2\{C\}$       |
| $Z_3$ | $Z_3\{A, B\}$    | $Z_6\{A, B, C\}$ | $Z_6\{A, B, C\}$ |
| $Z_4$ | $Z_4\{A, C\}$    | $Z_6\{A, B, C\}$ | $Z_4\{A, C\}$    |
| $Z_5$ | $Z_5\{B, C\}$    | $Z_5\{B, C\}$    | $Z_5\{B, C\}$    |
| $Z_6$ | $Z_6\{A, B, C\}$ | $Z_6\{A, B, C\}$ | $Z_6\{A, B, C\}$ |
| $Z_7$ | $Z_7\{\}$        | $Z_7\{\}$        | $Z_7\{\}$        |



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein



**66115 / 2018 / Frühjahr / Thema 2 / Aufgabe 3**

Gesucht ist eine reguläre Sprache  $C \subseteq \{a, b\}^*$ , deren minimaler deterministischer endlicher Automat (DEA) mindestens 4 Zustände mehr besitzt als der minimale nichtdeterministische endliche Automat (NEA). Gehen Sie wie folgt vor:

- (a) Definieren Sie  $C \subseteq \{a, b\}^*$  und erklären Sie kurz, warum es bei dieser Sprache NEAs gibt, die deutlich kleiner als der minimale DEA sind.

Lösungsvorschlag

Sprache mit exponentiellem Blow-Up:

Ein NEA der Sprache

$$\begin{aligned} L_k &= \{xay \mid x, y \in \{a, b\}^* \wedge |y| = k - 1\} \\ &= \{w \in \{a, b\}^* \mid \text{der } k\text{-te Buchstabe von hinten ist ein } a\} \end{aligned}$$

kommt mit  $k + 1$  Zuständen aus.

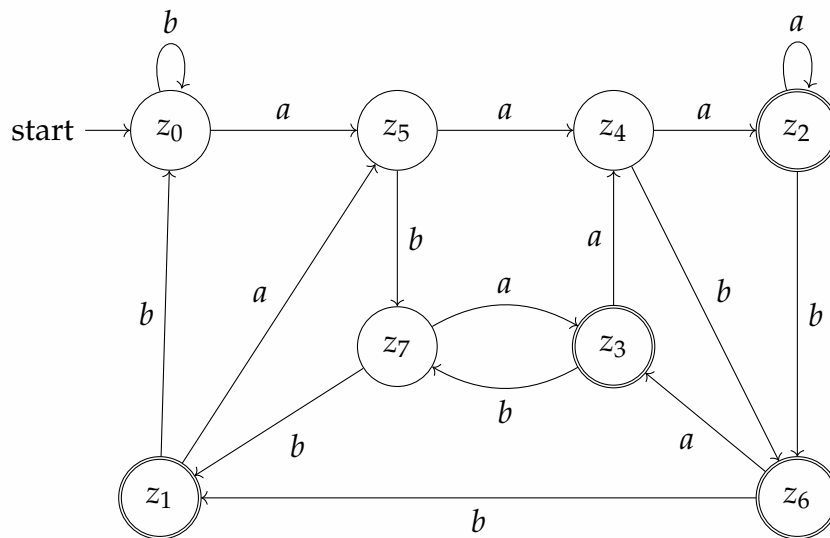
Jeder DEA  $M$  mit  $L(M) = L$  hat dann mindestens  $2^k$  Zustände. Wir wählen  $k = 3$ . Dann hat der zugehörige NEA 4 Zustände und der zugehörige DEA mindestens 8. Sei also  $L_k = \{xay \mid x, y \in \{a, b\}^* \wedge |y| = 2\}$  die gesuchte Sprache.

Der informelle Grund, warum ein DEA für die Sprache  $L_k$  groß sein muss, ist dass er sich immer die letzten  $n$  Symbole merken muss.<sup>a</sup>

<sup>a</sup><https://www.tcs.ifi.lmu.de/lehre/ss-2013/timi/handouts/handout-02>

- (b) Geben Sie den minimalen DEA  $M$  für  $C$  an. (Zeichnung des DEA genügt; die Minimalität muss nicht begründet werden.)

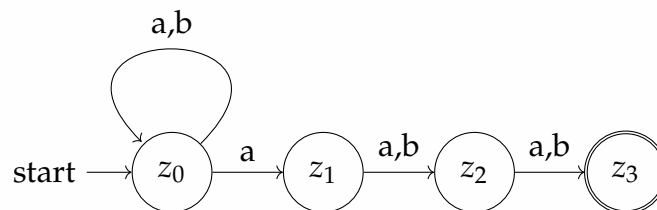
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ahhefpjir](http://flaci.com/Ahhefpjir)

- (c) Geben Sie einen NEA  $N$  für  $C$  an, der mindestens 4 Zustände weniger besitzt als  $M$ . (Zeichnung des NEA genügt)

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ajrz7h5r7](http://flaci.com/Ajrz7h5r7)

## 66115 / 2018 / Frühjahr / Thema 2 / Aufgabe 6

Der Hauptsatz der Laufzeitfunktionen ist bekanntlich folgendermaßen definiert:

### Exkurs: Master-Theorem

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

$a$  = Anzahl der rekursiven Aufrufe, Anzahl der Unterprobleme in der Rekursion ( $a \geq 1$ ).

$\frac{1}{b}$  = Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird, Anteil an der Verkleinerung des Problems ( $b > 1$ ).

$f(n)$  = Kosten (Aufwand, Nebenkosten), die durch die Division des Problems und die Kombination der Teillösungen entstehen. Eine von  $T(n)$  unabhängige und nicht negative Funktion.

Dann gilt:

1. Fall:  $T(n) \in \Theta(n^{\log_b a})$

falls  $f(n) \in \mathcal{O}(n^{\log_b a - \varepsilon})$  für  $\varepsilon > 0$

2. Fall:  $T(n) \in \Theta(n^{\log_b a} \cdot \log n)$

falls  $f(n) \in \Theta(n^{\log_b a})$

3. Fall:  $T(n) \in \Theta(f(n))$

falls  $f(n) \in \Omega(n^{\log_b a + \varepsilon})$  für  $\varepsilon > 0$  und ebenfalls für ein  $c$  mit  $0 < c < 1$  und alle hinreichend großen  $n$  gilt:  $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$

- (a) Betrachten Sie die folgende Methode `m` in Java, die initial mit `m(r, 0, r.length)` für das Array `r` aufgerufen wird. Geben Sie dazu eine Rekursionsgleichung  $T(n)$  an, welche die Anzahl an Rechenschritten von `m` in Abhängigkeit von der Länge  $n = r.length$  berechnet.

```
public static int m(int[] r, int lo, int hi) {
 if (lo < 8 || hi <= 10 || lo >= r.length || hi > r.length) {
 throw new IllegalArgumentException();
 }

 if (hi - lo == 1) {
 return r[lo];
 } else if (hi - lo == 2) {
 return Math.max(r[lo], r[lo + 1]); // O(1)
 } else {
 int s = (hi - lo) / 3;
 int x = m(r, lo, lo + s);
 int y = m(r, lo + s, lo + 2 * s);
 int z = m(r, lo + 2 * s, hi);
 return Math.max(Math.max(x, y), z); // O(1)
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2018/fruehjahr/MasterTheorem.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2018/fruehjahr/MasterTheorem.java)

Lösungsvorschlag

**Allgemeine Rekursionsgleichung:**

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

**Anzahl der rekursiven Aufrufe ( $a$ ):**

3

**Anteil Verkleinerung des Problems ( $b$ ):**

um  $\frac{1}{3}$  also  $b = 3$

**Laufzeit der rekursiven Funktion ( $f(n)$ ):**

$\mathcal{O}(1)$

**Ergibt folgende Rekursionsgleichung:**

$$T(n) = 3 \cdot T\left(\frac{n}{3}\right) + \mathcal{O}(1)$$

- (b) Ordnen Sie die rekursive Funktion  $T(n)$  aus (a) einem der drei Fälle des Mastertheorems zu und geben Sie die resultierende Zeitkomplexität an. Zeigen Sie dabei, dass die Voraussetzung des Falles erfüllt ist.

Lösungsvorschlag

**1. Fall:**  $f(n) \in \mathcal{O}(n^{\log_b a - \varepsilon})$ :

$$f(n) \in \mathcal{O}(n^{\log_3 3 - \varepsilon}) = \mathcal{O}(n^{1 - \varepsilon}) = \mathcal{O}(1) \text{ für } \varepsilon = 1$$

**2. Fall:**  $f(n) \in \Theta(n^{\log_b a})$ :

$$f(n) \notin \Theta(n^{\log_3 3}) = \Theta(n^1)$$

**3. Fall:**  $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ :

$$f(n) \notin \Omega(n^{\log_3 3 + \varepsilon}) = \Omega(n^{1 + \varepsilon})$$

Also:  $T(n) \in \Theta(n^{\log_b a})$

## 66115 / 2018 / Frühjahr / Thema 2 / Aufgabe 7

- (a) Gegeben ist das folgende Array von Zahlen:  $[23, 5, 4, 67, 30, 15, 25, 21]$ .

Sortieren Sie das Array mittels Quicksort in-situ aufsteigend von links nach rechts. Geben Sie die (Teil-)Arrays nach jeder Swap-Operation (auch wenn Elemente mit sich selber getauscht werden) und am Anfang jedes Aufrufs der rekursiven Methode an. Verwenden Sie als Pivotelement jeweils das rechteste Element im Teilarray und markieren Sie dieses entsprechend. Teilarrays der Länge  $\leq 2$  dürfen im rekursiven Aufruf durch direkten Vergleich sortiert werden. Geben Sie am Ende das sortierte Array an.

- (b) Welche Worst-Case-Laufzeit (O-Notation) hat Quicksort für  $n$  Elemente? Geben Sie ein Array mit fünf Elementen an, in welchem die Quicksort-Variante aus (a) diese Worst-Case-Laufzeit benötigt (ohne Begründung).

## 66115 / 2018 / Frühjahr / Thema 2 / Aufgabe 8

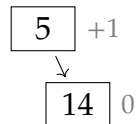
Bearbeiten Sie folgende Aufgaben zu AVL-Suchbäumen. Geben Sie jeweils bei jeder einzelnen Operation zum Einfügen, Löschen, sowie jeder elementaren Operation zum Wiederherstellen der AVL-Baumeigenschaften den entstehenden Baum als Baumzeichnung an. Geben Sie zur Darstellung der elementaren Operation auch vorübergehend ungültige AVL-Bäume an und stellen Sie Doppelrotationen in zwei Schritten dar. Dabei sollen die durchgeführten Operationen klar gekennzeichnet sein und die Baumknoten immer mit aktuellen Balancewerten versehen sein.

- (a) Fügen Sie (manuell) nacheinander die Zahlen 5, 14, 28, 10, 3, 12, 13 in einen anfangs leeren AVL-Baum ein.

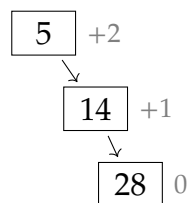
Einfügen von „5“:



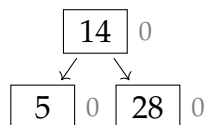
Einfügen von „14“:



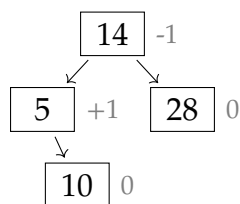
Einfügen von „28“:



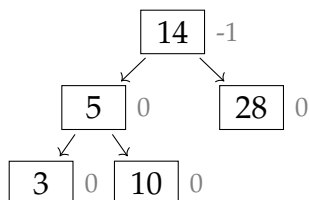
Linksrotation:



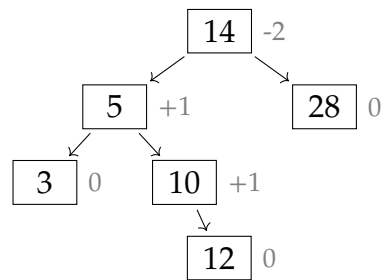
Einfügen von „10“:



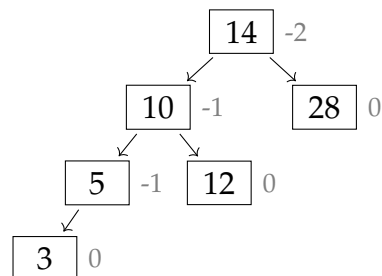
Einfügen von „3“:



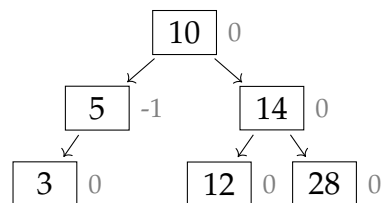
Einfügen von „12“:



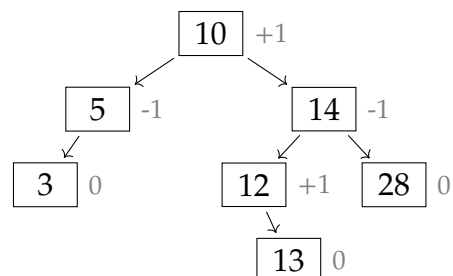
Linksrotation:



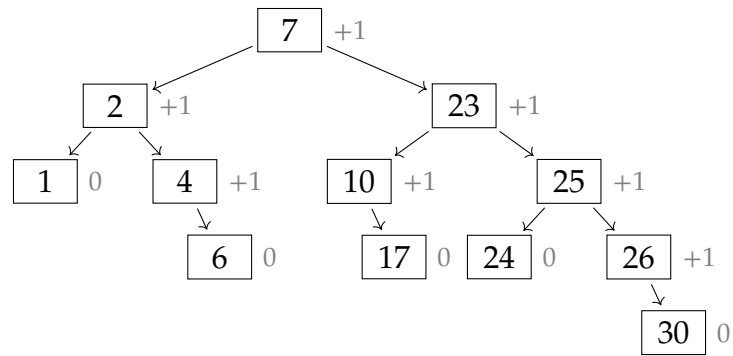
Rechtsrotation:



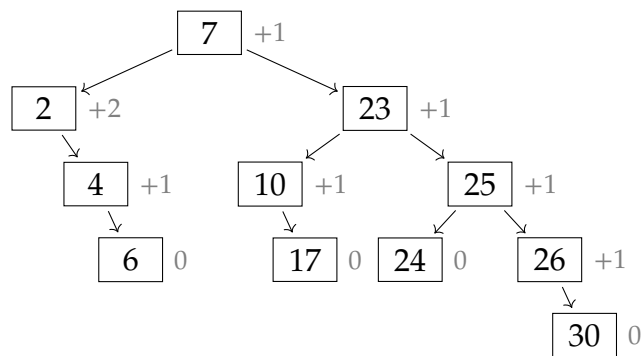
Einfügen von „13“:



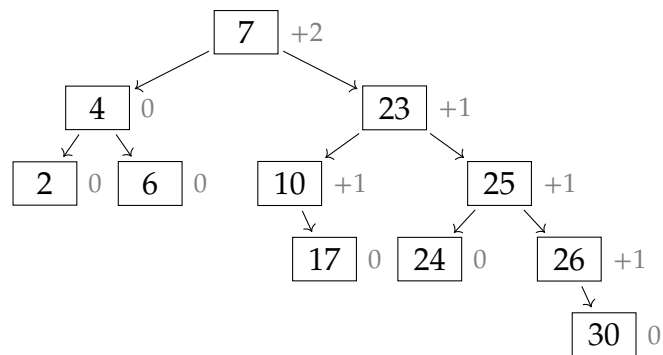
- (b) Gegeben sei folgender AVL-Baum. Löschen Sie nacheinander die Knoten 1 und 23. Bei Wahlmöglichkeiten nehmen Sie jeweils den kleineren Wert anstatt eines größeren.



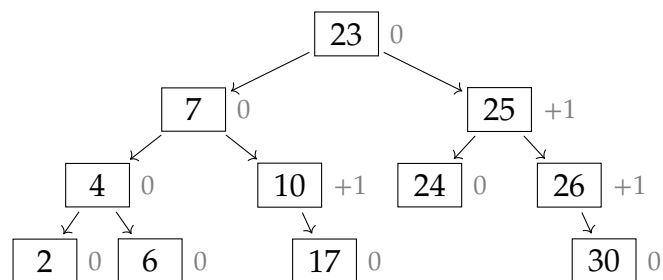
Löschen von „1“:



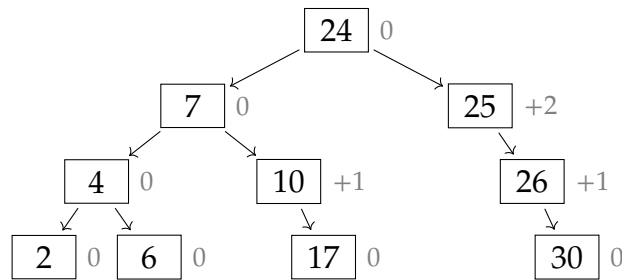
Linksrotation:



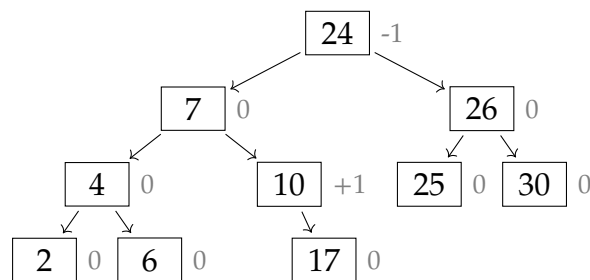
Linksrotation:



Löschen von „23“:

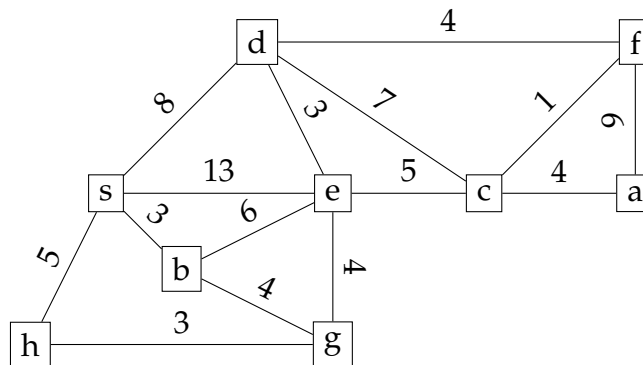


*Linksrotation:*



## 66115 / 2018 / Frühjahr / Thema 2 / Aufgabe 9

Gegeben sei folgender Graph  $G$ .



- (a) Berechnen Sie mithilfe des Algorithmus von Dijkstra die kürzesten Wege vom Knoten  $s$  zu allen anderen Knoten im Graphen  $G$ . Erstellen Sie dazu eine Tabelle mit zwei Spalten und stellen Sie jeden einzelnen Schritt des Verfahrens in einer eigenen Zeile dar. Geben Sie in der ersten Spalte den jeweils als nächstes fertigzustellenden Knoten  $v$  (wird sog. „schwarz“) als Tripel  $(v, p, \delta)$  mit  $v$  als Knotenname,  $p$  als aktueller Vorgängerknoten und  $\delta$  als aktuelle Distanz von  $s$  zu  $v$  über  $p$  an. Führen Sie in der zweiten Spalten alle anderen bisher erreichten Knoten  $v$  ebenfalls als Tripel  $(v, p, \delta)$  auf, wobei diese sog. „grauen Randknoten“ in folgenden Durchgängen erneut betrachtet werden müssen. Zeichnen Sie anschließend den entstandenen Wegebaum, öden Graphen  $G$ , in dem nur noch diejenigen Kanten vorkommen, die Teil der kürzesten Wege von  $s$  zu allen anderen Knoten sind.

| Nr | „schwarze“ Knoten | „graue“ Randknoten                         |
|----|-------------------|--------------------------------------------|
| 1  | (s, -, 0)         | [(b, s, 3)] (d, s, 8) (e, s, 13) (h, s, 5) |
| 2  | (b, s, 3)         | (d, s, 8) (e, b, 9) (g, b, 7) [(h, s, 5)]  |
| 3  | (h, s, 5)         | (d, s, 8) (e, b, 9) [(g, b, 7)]            |
| 4  | (g, b, 7)         | [(d, s, 8)] (e, b, 9)                      |
| 5  | (d, s, 8)         | (c, d, 15) [(e, b, 9)] (f, d, 12)          |
| 6  | (e, b, 9)         | (c, e, 14) [(f, d, 12)]                    |
| 7  | (f, d, 12)        | (a, f, 21) [(c, f, 13)]                    |
| 8  | (c, f, 13)        | [(a, c, 17)]                               |
| 9  | (a, c, 17)        |                                            |

```

graph TD
 s[s] ---|5|h[h]
 s ---|8|d[d]
 s ---|3|b[b]
 d ---|4|f[f]
 b ---|6|e[e]
 b ---|4|g[g]
 f ---|1|c[c]
 c ---|4|a[a]

```

### Alternativer Lösungsweg



| Nr. | besucht | a         | b        | c         | d        | e        | f         | g        | h        | s        |
|-----|---------|-----------|----------|-----------|----------|----------|-----------|----------|----------|----------|
| 0   |         | $\infty$  | $\infty$ | $\infty$  | $\infty$ | $\infty$ | $\infty$  | $\infty$ | $\infty$ | 0        |
| 1   | s       | $\infty$  | 3        | $\infty$  | 8        | 13       | $\infty$  | $\infty$ | 5        | <b>0</b> |
| 2   | b       | $\infty$  | <b>3</b> | $\infty$  | 8        | 9        | $\infty$  | 7        | 5        |          |
| 3   | h       | $\infty$  |          | $\infty$  | 8        | 9        | $\infty$  | 7        | <b>5</b> |          |
| 4   | g       | $\infty$  |          | $\infty$  | 8        | 9        | $\infty$  | <b>7</b> |          |          |
| 5   | d       | $\infty$  |          | 15        | <b>8</b> | 9        | 12        |          |          |          |
| 6   | e       | $\infty$  |          | 14        |          | <b>9</b> | 12        |          |          |          |
| 7   | f       | 21        |          | 13        |          |          | <b>12</b> |          |          |          |
| 8   | c       | 17        |          | <b>13</b> |          |          |           |          |          |          |
| 9   | a       | <b>17</b> |          |           |          |          |           |          |          |          |

| nach              | Entfernung | Reihenfolge | Pfad                                                        |
|-------------------|------------|-------------|-------------------------------------------------------------|
| $s \rightarrow a$ | 17         | 9           | $s \rightarrow d \rightarrow f \rightarrow c \rightarrow a$ |
| $s \rightarrow b$ | 3          | 2           | $s \rightarrow b$                                           |
| $s \rightarrow c$ | 13         | 8           | $s \rightarrow d \rightarrow f \rightarrow c$               |
| $s \rightarrow d$ | 8          | 5           | $s \rightarrow d$                                           |
| $s \rightarrow e$ | 9          | 6           | $s \rightarrow b \rightarrow e$                             |
| $s \rightarrow f$ | 12         | 7           | $s \rightarrow d \rightarrow f$                             |
| $s \rightarrow g$ | 7          | 4           | $s \rightarrow b \rightarrow g$                             |
| $s \rightarrow h$ | 5          | 3           | $s \rightarrow h$                                           |
| $s \rightarrow s$ | 0          | 1           |                                                             |

(b) Der Dijkstra-Algorithmus liefert bekanntlich auf Graphen mit negativen Kantengewichten unter Umständen ein falsches Ergebnis.

- (i) Geben Sie einen Graphen mit negativen Kantengewichten an, sodass der Dijkstra-Algorithmus ausgehend von einem von Ihnen ausgezeichneten Startknoten ein korrektes Ergebnis liefert.

Lösungsvorschlag

Startknoten a

```

graph TD
 a[a] ---|10| b[b]
 b[b] ---|-1| c[c]
 a[a] ---|2| c[c]

```

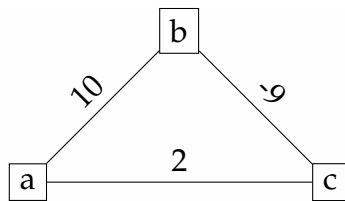
| Nr. | besucht | a | b        | c        |
|-----|---------|---|----------|----------|
| 0   |         | 0 | $\infty$ | $\infty$ |
| 1   | a       | 0 | 10       | 2        |
| 2   | c       |   | 10       | 2        |
| 3   | b       |   | 10       |          |

Richtig: a - c: 2

- (ii) Geben Sie einen Graphen mit negativen Kantengewichten an, sodass der Dijkstra-Algorithmus ausgehend von einem von Ihnen ausgezeichneten Startknoten ein falsches Ergebnis liefert.

Lösungsvorschlag

Startknoten a



| Nr. | besucht | a        | b         | c        |
|-----|---------|----------|-----------|----------|
| 0   |         | 0        | $\infty$  | $\infty$ |
| 1   | a       | <b>0</b> | 10        | 2        |
| 2   | c       |          | 10        | <b>2</b> |
| 3   | b       |          | <b>10</b> |          |

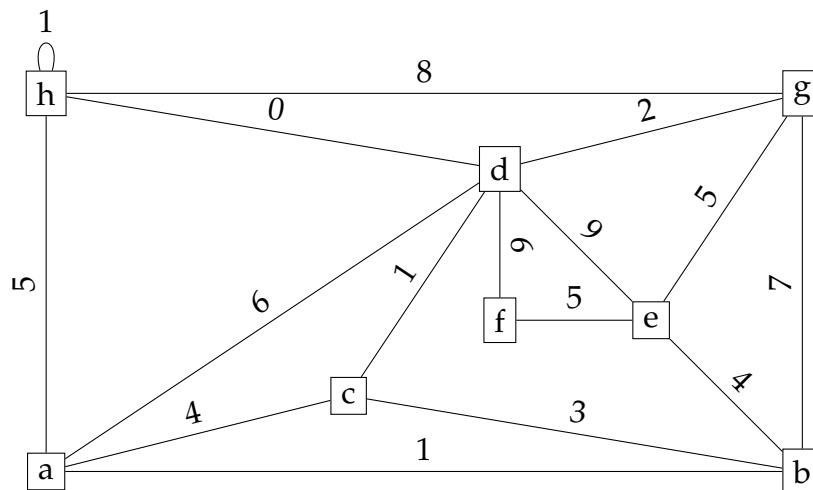
falsch: a - c: müsste 1 (10 - 9) sein.

Ein Beweis oder eine Begründung ist jeweils nicht erforderlich.

## 66115 / 2018 / Frühjahr / Thema 2 / Aufgabe 10

- (a) Berechnen Sie mithilfe des Algorithmus von Prim ausgehend vom Knoten  $a$  einen minimalen Spannbaum des ungerichteten Graphen  $G$ , der durch folgende Adjazenzmatrix gegeben ist:

$$\begin{array}{c}
 \begin{array}{cccccccc}
 & a & b & c & d & e & f & g & h \\
 \begin{array}{l} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{array} & \begin{pmatrix} * & 1 & 4 & 6 & - & - & - & 5 \\
 1 & * & 3 & - & 4 & - & 7 & - \\
 4 & 3 & * & 1 & - & - & - & - \\
 6 & - & 1 & * & 9 & 6 & 2 & 0 \\
 - & 4 & - & 9 & * & 5 & 5 & - \\
 - & - & - & 6 & 5 & * & - & - \\
 - & 7 & - & 2 & 5 & - & * & 8 \\
 5 & - & - & 0 & - & - & 8 & 1 \end{pmatrix}
 \end{array}
 \end{array}$$

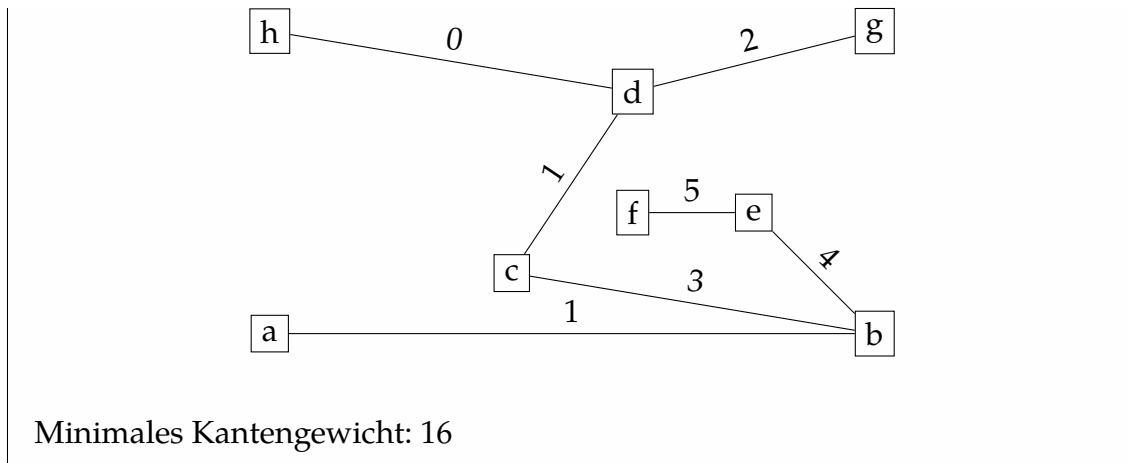


Erstellen Sie dazu eine Tabelle mit zwei Spalten und stellen Sie jeden einzelnen Schritt des Verfahrens in einer eigenen Zeile dar. Geben Sie in der ersten Spalte denjenigen Knoten  $v$ , der vom Algorithmus als nächstes in den Ergebnisbaum aufgenommen wird (dieser sog. „schwarze“ Knoten ist damit fertiggestellt), als Tripel  $(v, p, \delta)$  mit  $v$  als Knotenname,  $p$  als aktueller Vorgängerknoten und  $\delta$  als aktuelle Distanz von  $v$  zu  $p$  an. Führen Sie in der zweiten Spalte alle anderen vom aktuellen Spannbaum direkt erreichbaren Knoten  $v$  (sog. „graue Randknoten“) ebenfalls als Tripel  $(v, p, \delta)$  auf.

Zeichnen Sie anschließend den entstandenen Spannbaum und geben sein Gewicht an.

Lösungsvorschlag

| „schwarze“                 | „graue“ Randknoten                                  |
|----------------------------|-----------------------------------------------------|
| $(a, \text{NULL}, \infty)$ | $(b, a, 1) (c, a, 4) (h, a, 5) (d, a, 6)$           |
| $(b, a, 1)$                | $(c, b, 3) (e, b, 4) (h, a, 5) (d, a, 6) (g, b, 7)$ |
| $(c, b, 3)$                | $(d, c, 1) (e, b, 4) (h, a, 5) (g, b, 7)$           |
| $(d, c, 1)$                | $(h, d, 0) (g, d, 2) (e, b, 4) (f, d, 6)$           |
| $(h, d, 0)$                | $(g, d, 2) (e, b, 4) (f, d, 6)$                     |
| $(g, d, 2)$                | $(e, b, 4) (f, d, 6)$                               |
| $(e, b, 4)$                | $(f, e, 5)$                                         |
| $(f, e, 5)$                |                                                     |



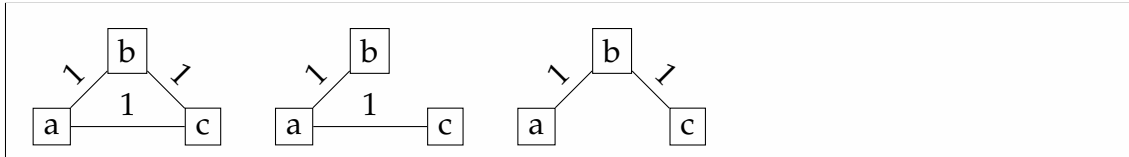
- (b) Welche Worst-Case-Laufzeitkomplexität hat der Algorithmus von Prim, wenn die grauen Knoten in einem Heap (= Halde) nach Distanz verwaltet werden? Sei dabei  $n$  die Anzahl an Knoten und  $m$  die Anzahl an Kanten des Graphen. Eine Begründung ist nicht erforderlich.

Lösungsvorschlag

$$\mathcal{O}(n \cdot \log(n) + m)$$

- (c) Zeigen Sie durch ein kleines Beispiel, dass ein minimaler Spannbaum eines ungerichteten Graphen nicht immer eindeutig ist.

Lösungsvorschlag



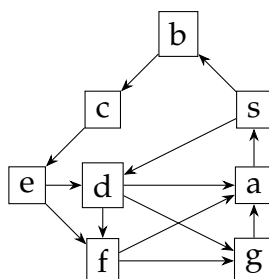
- (d) Skizzieren Sie eine Methode, mit der ein maximaler Spannbaum mit einem beliebigen Algorithmus für minimale Spannbäume berechnet werden kann. In welcher Laufzeitkomplexität kann ein maximaler Spannbaum berechnet werden?

Lösungsvorschlag

Alle Kantengewichte negieren. In  $\mathcal{O}(n \cdot \log(n) + m)$  wie der Algorithmus von Prim.

## 66115 / 2018 / Frühjahr / Thema 2 / Aufgabe 11

Gegeben sei der folgende gerichtete Graph  $G$ :



Traversieren Sie  $G$  ausgehend vom Knoten  $s$  mittels

- (a) Tiefensuche (DFS),

Lösungsvorschlag

Rekursiv ohne Keller:

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| s | b | c | e | d | a | f | g |

- (b) Breitensuche (BFS)

Lösungsvorschlag

mit Warteschlange:

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| s | b | d | c | a | f | g | e |

und geben Sie jeweils die erhaltene Nummerierung der Knoten an. Besuchen Sie die Nachbarn eines Knotens bei Wahlmöglichkeiten immer in alphabetisch aufsteigender Reihenfolge.

### 66115 / 2018 / Herbst / Thema 1 / Aufgabe 3

- (a) Entwerfen Sie eine kontextfreie Grammatik für die folgende kontextfreie Sprache über dem Alphabet  $\Sigma = \{a, b, c\}$ :

$$L = \{wb^{3k}c^{2k+1}v \mid k \in \mathbb{N}, |w|_c = |u|_a\}$$

(Hierbei bezeichnet  $|u|_x$  die Anzahl des Zeichens  $x$  in dem Wort  $u$ , und es gilt  $0 \in \mathbb{N}$ .) Erklären Sie den Zweck der einzelnen Nichtterminale (Variablen) und der Grammatikregeln Ihrer Grammatik.

- (b) Betrachten Sie die folgende kontextfreie Grammatik

$$G = (\{S, X, Y, Z\}, \{z, y\}, P, S)$$

mit den Produktionen

$$P = \left\{ \begin{array}{l} S \rightarrow ZX \mid y \\ X \rightarrow ZS \mid SS \mid x \\ Y \rightarrow SX \mid YZ \\ Z \rightarrow XX \mid XS \end{array} \right.$$

Benutzen Sie den Algorithmus von Cocke-Younger-Kasami (CYK) um zu zeigen, dass das Wort  $xxxyx$  zu der von  $G$  erzeugten Sprache  $L(G)$  gehört.

Lösungsvorschlag

|       |   |   |   |   |
|-------|---|---|---|---|
| x     | x | x | y | x |
| X     | X | X | S | X |
| Z     | Z | Z | Y |   |
| S     | X | S |   |   |
| Z,X   | Z |   |   |   |
| X,S,Z |   |   |   |   |

$\Rightarrow xxxyx \in L(G)$

(c) Geben Sie eine Ableitung des Wortes  $xxxyx$  mit  $G$  an.

## 66115 / 2018 / Herbst / Thema 1 / Aufgabe 5

Hinweis: Wir betrachten in dieser Aufgabe binäre Suchbäume, bei denen jeder innere Knoten genau zwei Kinder hat. Schlüssel werden nur in den inneren Knoten gespeichert - die Blätter speichern keinerlei Informationen.

(a) Welche Eigenschaften muss ein binärer Suchbaum haben, damit er ein AVL-Baum ist?

Lösungsvorschlag

Er muss die zusätzliche Eigenschaft haben, dass sich an jedem Knoten die Höhe der beiden Teilbäume um höchstens eins unterscheidet

(b) Mit  $n(h)$  bezeichnen wir die minimale Anzahl innerer Knoten eines AVL-Baums der Höhe  $h$ .

- (i) Begründen Sie, dass  $n(1) = 1$  und  $n(2) = 2$ .
- (ii) Begründen Sie, dass  $n(h) = 1 + n(h-1) + n(h-2)$ .
- (iii) Folgern Sie, dass  $n(h) > 2^{\frac{h}{2}-1}$ .

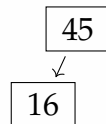
(c) Warum ist die Höhe jedes AVL-Baums mit  $n$  inneren Knoten  $O(\log n)$ ?

(d) Fügen Sie die Elemente (45, 16, 79, 31, 51, 87, 49, 61) in der angegebenen Reihenfolge in einen anfangs leeren binären Suchbaum ein (ohne Rebalancierungen). Zeichnen Sie den resultierenden Suchbaum nach jeder Einfügeoperation.

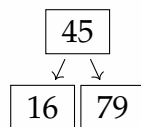
Einfügen von „45“:



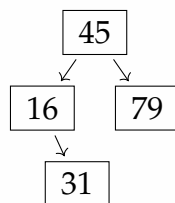
Einfügen von „16“:



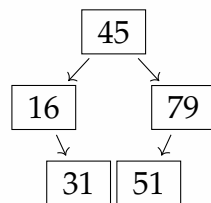
Einfügen von „79“:



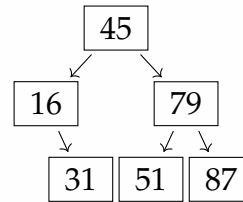
Einfügen von „31“:



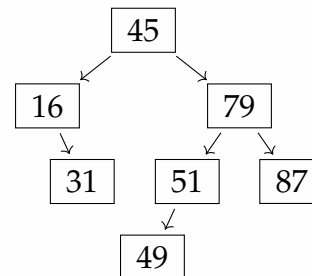
Einfügen von „51“:



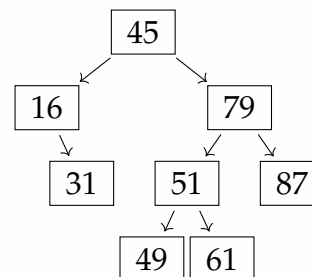
Einfügen von „87“:



Einfügen von „49“:



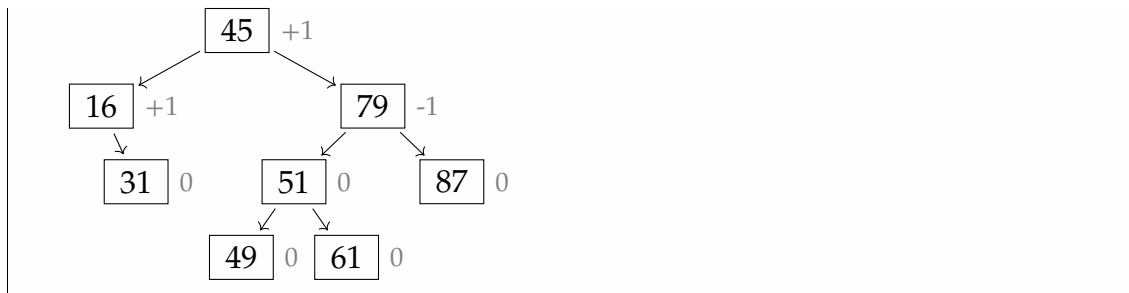
Einfügen von „61“:



- (e) Ist der resultierende Suchbaum aus Teilaufgabe 5.4 ein AVL-Baum? Begründen Sie Ihre Antwort.

Ja, wie in der untenstehenden Grafik zu sehen ist, unterscheiden sich die Höhe der Teilbäume von allen Knoten nur um höchstens eins.





- (f) Das Einfügen in einen AVL-Baum funktioniert (zunächst) wie beim binären Suchbaum durch Erweitern eines äußeren Knotens w:

vor dem Einfügen von 54 nach dem Einfügen von 54

Anschließend wird die AVL-Baum Eigenschaft (falls notwendig) durch eine (Doppel-)Rotation wiederhergestellt: Wenn  $z$  der erste Knoten auf dem Pfad  $P$  von  $w$  zur Wurzel ist, der nicht balanciert ist,  $y$  das Kind von  $z$  auf  $P$  und  $r$  das Kind von  $y$  auf  $P$ , und wenn  $(a, b, c)$  die Inorder-Reihenfolge von  $x, y, z$  ist, dann führen wir die Rotation aus, die benötigt wird, um  $b$  zum obersten Knoten der drei zu machen.

Die folgende Illustration zeigt den Fall, dass  $\text{key}(y) < \text{key}(x) < \text{key}(z)$ ,  $\delta(a, b, c) = (y, x, z)$ , wobei  $w$  ein Knoten in  $T_y$  ist.

Sei  $h$  die Höhe des Teilbaums  $T_z$ . Für  $i = 0, 1, 2$  sei  $h_i$  die Höhe des Teilbaums  $T_i$  und für  $v = 2, y, z$  sei  $h_v$  die Höhe des Teilbaums mit der Wurzel  $v$  vor der Restrukturierung. Begründen Sie, dass

- (i)  $h_0 = h$
  - (ii)  $h_1 = h - 1$
  - (iii)  $h_2 = h$
  - (iv)  $h_x = h + 1$
  - (v)  $h_y = h + 2$
  - (vi)  $h_z = h + 3$
- (g) Welche Höhe haben die Teilbäume mit den Wurzeln  $x, y, z$  nach der Restrukturierung? Begründen Sie Ihre Antworten.
- (h) Begründen Sie, dass die oben gezeigte Doppelrotation die AVL-Baum-Eigenschaft wiederherstellt.
- (i) Beschreiben Sie, wie ein binärer Baum der Höhe  $h$  in einem Array repräsentiert werden kann. Wie viel Speicherplatz ist für so eine Darstellung erforderlich?
- (j) Warum verwendet man bei der Implementierung von AVL-Bäumen eine verzeigte Struktur und nicht eine Array-basierte Repräsentation?

**66115 / 2018 / Herbst / Thema 2 / Aufgabe 4**

Wir betrachten ungerichtete Graphen  $G = (V, E)$ , wo  $E$  eine Teilmenge  $E$  hat, die wir exklusive Kanten nennen. Eine beschränkte Überdeckung von  $G$  ist eine Teilmenge  $U$  von  $V$ , so dass

- (a) jeder Knoten einen Nachbarknoten in  $U$  hat oder selbst in  $U$  liegt (für jeden Knoten  $u \in V \setminus U$  gibt es einen Knoten  $v \in U$  mit  $(u, v) \in E$ ) und
- (b) für jede exklusive Kante  $(u, v) \in E$  genau einer der Knoten  $u, v$  in  $U$  liegt.

Betrachten Sie nun die folgenden Entscheidungsprobleme:

**3SAT**

**Gegeben:** Aussagenlogische Formel  $p$  in 3KNF

**Frage:** Hat  $p$  eine erfüllende Belegung?

**BÜ**

**Gegeben:** Graph  $G = (V, E)$ , exklusive Kantenmenge  $E$ .  $C \subseteq E$  und  $k \in \mathbb{N}$

**Frage:** Hat  $G$  eine beschränkte Überdeckung  $U$  mit  $|U| \leq k$ ?

Beweisen Sie, dass BÜ NP-vollständig ist. Sie dürfen dabei annehmen, dass 3SAT NP-vollständig ist.

**66115 / 2018 / Herbst / Thema 2 / Aufgabe 6**

Ein sehr bekanntes Optimierungsproblem ist das sogenannte Rucksackproblem: Gegeben ist ein Rucksack mit der Tragfähigkeit  $B$ . Weiterhin ist eine endliche Menge von Gegenständen mit Werten und Gewichten gegeben. Nun soll eine Teilmenge der Gegenstände so ausgewählt werden, dass ihr Gesamtwert maximal ist, aber ihr Gesamtgewicht die Tragfähigkeit des Rucksacks nicht überschreitet.

Mathematisch exakt kann das Rucksackproblem wie folgt formuliert werden:

Gegeben ist eine endliche Menge von Objekten  $U$ . Durch eine Gewichtsfunktion  $w: U \rightarrow \mathbb{R}^+$  wird den Objekten ein Gewicht und durch eine Nutzenfunktion  $v: U \rightarrow \mathbb{R}^+$  ein festgelegter Nutzwert zugeordnet.

Des Weiteren gibt es eine vorgegebene Gewichtsschranke  $B \in \mathbb{R}^+$ . Gesucht ist eine Teilmenge  $K \subseteq U$ , die die Bedingung  $\sum_{u \in K} w(u) \leq B$  einhält und die Zielfunktion  $\sum_{u \in K} v(u)$  maximiert.

Das Rucksackproblem ist NP-vollständig (Problemgröße ist die Anzahl der Objekte), sodass es an dieser Stelle wenig Sinn macht, über eine effiziente Lösung nachzudenken. Lösen Sie das Rucksackproblem daher mittels Backtracking und formulieren Sie

einen entsprechenden Algorithmus. Gehen Sie davon aus, dass die Gewichtsschranke  $B$  sowie die Anzahl an Objekten  $N$  beliebig, aber fest vorgegeben sind.

Das Programm soll folgende Ausgaben liefern:

- (a) Maximaler Nutzwert, der durch eine Objektauswahl unter Einhaltung der Gewichtsschranke  $B$  erreicht werden kann.
- (b) Das durch die maximierende Objektmenge erreichte Gesamtgewicht.
- (c) Diejenigen Objekte (Objektnummern) aus  $U$ , die zur Maximierung des Nutzwerts beigetragen haben.

Lösungsvorschlag

```
/**
 * https://stackoverflow.com/a/14186622
 */
public class Rucksack {
 // static int[] werte = new int[] { 894, 260, 392, 281, 27 };
 // static int[] gewichte = new int[] { 8, 6, 4, 0, 21 };
 // static int[] werte = new int[] { 4, 2, 10, 1, 2 };
 // static int[] gewichte = new int[] { 12, 1, 4, 1, 2 };
 static int werte[] = new int[] { 60, 100, 120 };
 static int gewichte[] = new int[] { 10, 20, 30 };

 /**
 * Gewichtsschranke
 */
 //static int B = 30;
 //static int B = 15;
 static int B = 50;

 /**
 * Diejenigen Objekte aus U, die zur Maximierung des Nutzwerts beigetragen
 * haben.
 */
 static boolean[] auswahl = new boolean[werte.length];

 private static int berechne(int i, int W) {
 if (i < 0) {
 return 0;
 }
 int alt = berechne(i - 1, W);
 if (gewichte[i] > W) {
 // Backtracking!
 auswahl[i] = false;
 return alt;
 } else {
 int neu = berechne(i - 1, W - gewichte[i]) + werte[i];
 if (alt >= neu) {
 // Backtracking!
 }
 }
 }
}
```

```

 auswahl[i] = false;
 return alt;
 } else {
 auswahl[i] = true;
 return neu;
 }
}
}

static void werteAus() {
 System.out.println(berechne(werte.length - 1, B));

 int gesamtGewicht = 0;
 int gesamtWert = 0;

 for (int i = 0; i < auswahl.length; i++) {
 if (auswahl[i]) {
 gesamtGewicht += gewichte[i];
 gesamtWert += werte[i];
 System.out.println("Objekt-Nr. " + i + " Gewicht: " + gewichte[i] +
 " Wert: " + werte[i]);
 }
 }

 System.out.println("Gesamtgewicht: " + gesamtGewicht);
 System.out.println("Gesamtwert: " + gesamtWert);
}

public static void main(String[] args) {
 werteAus();
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2018/herbst/Rucksack.java](https://github.com/org/bschlangaul/examen/examen_66115/jahr_2018/herbst/Rucksack.java)

## 66115 / 2018 / Herbst / Thema 2 / Aufgabe 8

Gegeben sei das folgende Feld  $A$  mit 7 Schlüsseln:

$[15, 4, 10, 7, 1, 8, 10]$

- (a) Sortieren Sie das Feld mittels des Sortierverfahrens *Bubblesort*. Markieren Sie jeweils, welche zwei Feldwerte verglichen werden und geben Sie den Zustand des gesamten Feldes jeweils neu an, wenn Sie eine Vertauschung durchgeführt haben.

Lösungsvorschlag

|     |     |     |     |    |   |    |                     |
|-----|-----|-----|-----|----|---|----|---------------------|
| 15  | 4   | 10  | 7   | 1  | 8 | 10 | Eingabe             |
| 15  | 4   | 10  | 7   | 1  | 8 | 10 | Durchlauf Nr. 1     |
| >15 | 4<  | 10  | 7   | 1  | 8 | 10 | vertausche (i 0<>1) |
| 4   | >15 | 10< | 7   | 1  | 8 | 10 | vertausche (i 1<>2) |
| 4   | 10  | >15 | 7<  | 1  | 8 | 10 | vertausche (i 2<>3) |
| 4   | 10  | 7   | >15 | 1< | 8 | 10 | vertausche (i 3<>4) |

|    |     |     |     |     |     |     |                     |
|----|-----|-----|-----|-----|-----|-----|---------------------|
| 4  | 10  | 7   | 1   | >15 | 8<  | 10  | vertausche (i 4<>5) |
| 4  | 10  | 7   | 1   | 8   | >15 | 10< | vertausche (i 5<>6) |
| 4  | 10  | 7   | 1   | 8   | 10  | 15  | Durchlauf Nr. 2     |
| 4  | >10 | 7<  | 1   | 8   | 10  | 15  | vertausche (i 1<>2) |
| 4  | 7   | >10 | 1<  | 8   | 10  | 15  | vertausche (i 2<>3) |
| 4  | 7   | 1   | >10 | 8<  | 10  | 15  | vertausche (i 3<>4) |
| 4  | 7   | 1   | 8   | 10  | 10  | 15  | Durchlauf Nr. 3     |
| 4  | >7  | 1<  | 8   | 10  | 10  | 15  | vertausche (i 1<>2) |
| 4  | 1   | 7   | 8   | 10  | 10  | 15  | Durchlauf Nr. 4     |
| >4 | 1<  | 7   | 8   | 10  | 10  | 15  | vertausche (i 0<>1) |
| 1  | 4   | 7   | 8   | 10  | 10  | 15  | Durchlauf Nr. 5     |
| 1  | 4   | 7   | 8   | 10  | 10  | 15  | Ausgabe             |

Selectionsort  
Algorithmische Komplexität  
(O-Notation)

- (b) Sortieren Sie das Feld mittels des Sortierverfahrens *Selectionsort*. Markieren Sie jeweils, welche zwei Feldwerte verglichen werden und geben Sie den Zustand des gesamten Feldes jeweils neu an, wenn Sie eine Vertauschung durchgeführt haben.

Lösungsvorschlag

|     |    |     |    |    |    |     |                     |
|-----|----|-----|----|----|----|-----|---------------------|
| 15  | 4  | 10  | 7  | 1  | 8  | 10  | Eingabe             |
| 15  | 4  | 10  | 7  | 1  | 8  | 10* | markiere (i 6)      |
| >15 | 4  | 10  | 7  | 1  | 8  | 10< | vertausche (i 0<>6) |
| 10  | 4  | 10  | 7  | 1  | 8* | 15  | markiere (i 5)      |
| >10 | 4  | 10  | 7  | 1  | 8< | 15  | vertausche (i 0<>5) |
| 8   | 4  | 10  | 7  | 1* | 10 | 15  | markiere (i 4)      |
| 8   | 4  | >10 | 7  | 1< | 10 | 15  | vertausche (i 2<>4) |
| 8   | 4  | 1   | 7* | 10 | 10 | 15  | markiere (i 3)      |
| >8  | 4  | 1   | 7< | 10 | 10 | 15  | vertausche (i 0<>3) |
| 7   | 4  | 1*  | 8  | 10 | 10 | 15  | markiere (i 2)      |
| >7  | 4  | 1<  | 8  | 10 | 10 | 15  | vertausche (i 0<>2) |
| 1   | 4* | 7   | 8  | 10 | 10 | 15  | markiere (i 1)      |
| 1   | >4 | 7   | 8  | 10 | 10 | 15  | vertausche (i 1<>1) |
| 1*  | 4  | 7   | 8  | 10 | 10 | 15  | markiere (i 0)      |
| >1  | 4  | 7   | 8  | 10 | 10 | 15  | vertausche (i 0<>0) |
| 1   | 4  | 7   | 8  | 10 | 10 | 15  | Ausgabe             |

- (c) Vergleichen Sie beide Sortierverfahren hinsichtlich ihres Laufzeitverhaltens im *best case*. Welches Verfahren ist in dieser Hinsicht besser, wenn das zu sortierende Feld anfangs bereits sortiert ist? Begründen Sie Ihre Antwort.

Der Bubblesort-Algorithmus hat im *best case* eine Laufzeit von  $\mathcal{O}(n)$ , der Selectionsort-Algorithmus  $\mathcal{O}(n^2)$ .

Bubblesort steuert seine äußere bedingte Wiederholung in vielen Implementationen über eine boolesche Hilfsvariable `getauscht`, die beim Betreten der Schleife erstmals auf falsch gesetzt wird. Erst wenn Vertauschungen vorgenommen werden müssen, wird diese Variable auf wahr gesetzt und die äußere

re Schleife läuft ein weiteres Mal ab. Ist das zu sortierende Feld bereits sortiert, durchsucht der Algorithmus des Bubblesort das Feld einmal und terminiert dann.

Der Selectionsort-Algorithmus hingegen ist mit zwei ineinander verschränkten Schleifen umgesetzt, deren Wiederholungsanzahl sich starr nach der Anzahl der Elemente im Feld richtet.

## Bubblesort

```
int durchlaufNr = 0;
boolean getauscht;
do {
 durchlaufNr++;
 berichte.feld("Durchlauf Nr. " + durchlaufNr);
 getauscht = false;
 for (int i = 0; i < zahlen.length - 1; i++) {
 if (zahlen[i] > zahlen[i + 1]) {
 // Elemente vertauschen
 vertausche(i, i + 1);
 getauscht = true;
 }
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/sortier/BubbleIterativ.java](https://github.com/bschlangaul/sortier/BubbleIterativ.java)

## Selectionsort

```
// Am Anfang ist die Markierung das letzte Element im Zahlen-Array.
int markierung = zahlen.length - 1;
while (markierung >= 0) {
 berichte.feldMarkierung(markierung);
 // Bestimme das größtes Element.
 // max ist der Index des größten Elements.
 int max = 0;
 // Wir vergleichen zuerst die Zahlen mit der Index-Number
 // 0 und 1, dann 1 und 2, etc. bis zur Markierung
 for (int i = 1; i <= markierung; i++) {
 if (zahlen[i] > zahlen[max]) {
 max = i;
 }
 }

 // Tausche zahlen[markierung] mit dem gefundenem Element.
 vertausche(markierung, max);
 // Die Markierung um eins nach vorne verlegen.
 markierung--;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/sortier/SelectionRechtsIterativ.java](https://github.com/bschlangaul/sortier/SelectionRechtsIterativ.java)

**66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 1**

Antworten Sie auf die folgenden Behauptungen mit Wahr/Falsch und geben Sie eine kurze Begründung an.

- (a) Wenn  $L_2$  regulär ist und  $L_1 \subseteq L_2$  gilt, dann ist  $L_1$  auch regulär.

Lösungsvorschlag

Falsch

Nein. Wähle  $L = \Sigma^*$ . Wähle eine beliebige nicht-reguläre Sprache  $L'$ .  $L'$  ist Teilmenge von  $L$ .<sup>a</sup>

$L_2 = \{a^*b^*\}$  ...Regulär

$L_1 = \{a^n b^n\}$  ...nicht regulär,

da man sich das  $n$  merken muss, das bedeutet, dass man beispielsweise einen Automaten bräuchte, der unendlich viele Zustände besitzt. Das wiederum ist aber bei DEA's nicht möglich. (Da sie nur endlich viele Zustände haben können.) Für beide Sprachen gilt somit:  $L_1 \subseteq L_2$ , aber nicht beide sind regulär.  
b

<sup>a</sup><https://www.c-plusplus.net/forum/topic/287036/theoretische-informatik-teilmenge-einer-reguläre-sprache>

<sup>b</sup>[https://vowi.fsinf.at/images/3/3a/TU\\_Wien-Theoretische\\_Informatik\\_und\\_Logik\\_VU\\_\(Fermüller,\\_Freund\)-Übungen\\_SS13\\_-\\_Übungsblatt\\_1.pdf](https://vowi.fsinf.at/images/3/3a/TU_Wien-Theoretische_Informatik_und_Logik_VU_(Fermüller,_Freund)-Übungen_SS13_-_Übungsblatt_1.pdf)

- (b)  $L = \{a^q \mid \exists i \in \mathbb{N}. q = i^2\}$  ist bekanntlich nicht regulär. Behauptung:  $Q^*$  ist ebenfalls nicht regulär.

Lösungsvorschlag

Wahr.

Siehe Abschlusseigenschaften der Formalen Sprachen unter dem Kleene-Stern.

- (c) Wenn  $L \subseteq \Sigma^*$  entscheidbar ist, dann ist auch das Komplement  $\bar{L} = \Sigma^* \setminus L$  entscheidbar.

Lösungsvorschlag

Wahr

Zu jeder entscheidbaren Menge ist auch ihr Komplement entscheidbar.

$L$  entscheidbar, dann auch  $\bar{L}$  entscheidbar. Wir benutzen eine DTM  $\tau'$ , die die DTM  $\tau$  simuliert. Diese vertauscht die beiden Ausgaben „JA“ und „NEIN“.<sup>a</sup>

<sup>a</sup><http://www.informatikseite.de/theorie/node14.php#SECTION00044100000000000000>

- (d) Jedes  $\mathcal{NP}$ -vollständige Problem ist entscheidbar.



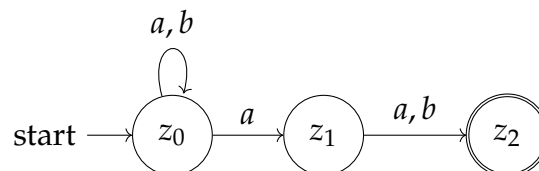
Wahr

Definition von  $\mathcal{NP}$ -Vollständigkeit:

vollständig für die Klasse der Probleme, die sich nichtdeterministisch in Polynomialzeit lösen lassen, wenn es zu den schwierigsten Problemen in der Klasse NP gehört, also sowohl in NP liegt als auch NP-schwer ist.

## 66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 2

- (a) Gegeben sei der nichtdeterministische endliche Automat  $A$  über dem Alphabet  $\Sigma = \{a, b\}$  wie folgt:



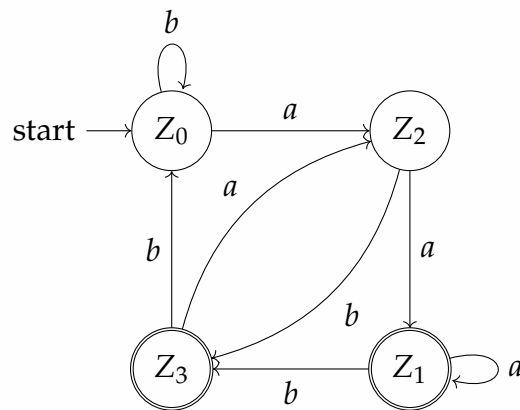
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Arozq4rm2](http://flaci.com/Arozq4rm2)

Konstruieren Sie einen deterministischen endlichen Automaten, der das Komplement  $\bar{L}(A) = \{w \in \Sigma^* \mid w \notin L(A)\}$  der von  $A$  akzeptierten Sprache  $L(A)$  akzeptiert.

Lösungsvorschlag

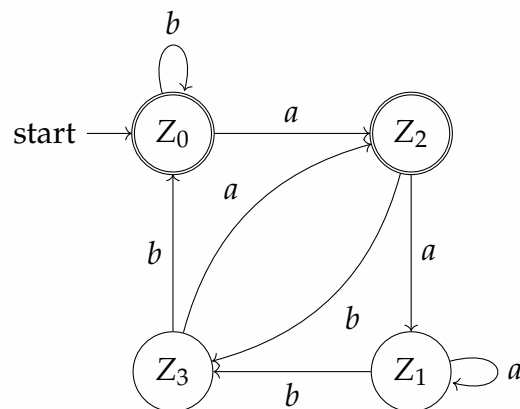
Wir konvertieren zuerst den nichtdeterministischen endlichen Automaten in einen deterministischen endlichen Automaten mit Hilfe des Potenzmengenalgorithmus.

| Zustandsmenge           | Eingabe $a$             | Eingabe $b$        |
|-------------------------|-------------------------|--------------------|
| $Z_0 \{z_0\}$           | $Z_1 \{z_0, z_1\}$      | $Z_0 \{z_0\}$      |
| $Z_1 \{z_0, z_1\}$      | $Z_2 \{z_0, z_1, z_2\}$ | $Z_3 \{z_0, z_2\}$ |
| $Z_2 \{z_0, z_1, z_2\}$ | $Z_2 \{z_0, z_1, z_2\}$ | $Z_3 \{z_0, z_2\}$ |
| $Z_3 \{z_0, z_2\}$      | $Z_1 \{z_0, z_1\}$      | $Z_0 \{z_0\}$      |



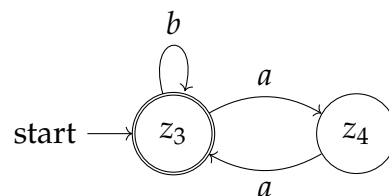
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Arxujcbdg](http://flaci.com/Arxujcbdg)

Wir vertauschen die End- und Nicht-End-Zustände, um das Komplement zu erhalten:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/A5zqsonq2](http://flaci.com/A5zqsonq2)

- (b) Gegeben sei zudem der nichtdeterministische Automat  $B$  über dem Alphabet  $\Sigma = \{a, b\}$ :

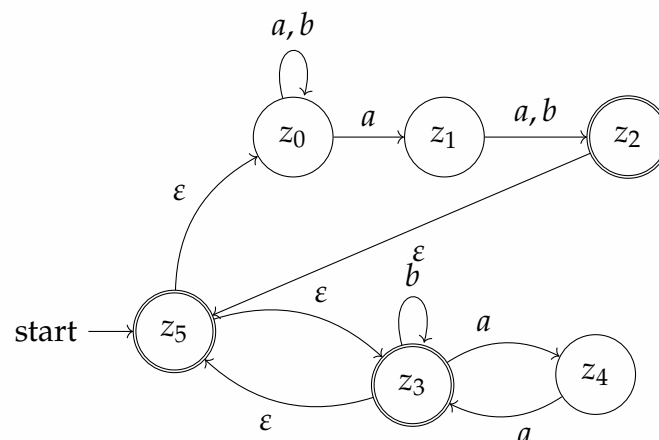


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Arafgk0h2](http://flaci.com/Arafgk0h2)

Konstruieren Sie einen endlichen Automaten (möglicherweise mit  $\varepsilon$ -Übergängen), der die Sprache  $(L(A)L(B))^* \subseteq \Sigma^*$  akzeptiert ( $A$  aus der vorigen Aufgabe). Erläutern Sie auch Ihre Konstruktionsidee.

Lösungsvorschlag

$L(A)L(B)^*$  ist die beliebige Konkatenation (Verknüpfung/Verkettung) der Sprachen  $L(A)$  und  $L(B)$  mit dem leeren Wort. Wir führen einen neuen Startzustand ( $z_5$ ) ein, der zugleich Endzustand ist. Dadurch wird das leere Wort akzeptiert. Dieser neue Startzustand führt über  $\varepsilon$ -Übergängen zu den ehemaligen Startzuständen der Automaten  $A$  und  $B$ . Die Endzustände der Automaten  $A$  und  $B$  führen über  $\varepsilon$ -Übergängen zu  $z_5$ . Dadurch sind beliebige Konkatenationen möglich.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Aro3uhzjz](https://flaci.com/Aro3uhzjz)

### 66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 3

Gegeben sei die kontextfreie Grammatik  $G = (V, \Sigma, P, S)$  mit Sprache  $L(G)$ , wobei  $V = \{S, T, U\}$  und  $\Sigma = \{a, b\}$ .  $P$  bestehe aus den folgenden Produktionen:

$P = \{$

$S \rightarrow TUUT$

$T \rightarrow aT \mid \varepsilon$

$U \rightarrow bUb \mid a$

$\}$

1

(a) Geben Sie fünf verschiedene Wörter  $w \in \Sigma^*$  mit  $w \in L(G)$  an.

<sup>1</sup><https://flaci.com/Gjpsin26a>

- aa
- aaaa
- ababbaba
- aababbabaa
- abbabbbabba

(b) Geben Sie eine explizite Beschreibung der Sprache  $L(G)$  an.

Lösungsvorschlag

$$L = \{ a^* b^n a b^{2n} a b^n a^* \mid n \in \mathbb{N}_0 \}$$

(c) Bringen Sie  $G$  in Chomsky-Normalform und erklären Sie Ihre Vorgehensweise.

Lösungsvorschlag

(i) **Elimination der  $\varepsilon$ -Regeln**

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen. —

$$P = \{$$

$$S \rightarrow TUUT \mid TUU \mid UUT \mid UU$$

$$T \rightarrow aT \mid a$$

$$U \rightarrow bUb \mid a$$

$$\}$$

(ii) **Elimination von Kettenregeln**

— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren. —

∅ Nichts zu tun

(iii) **Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt. —

$$P = \{$$

$$S \rightarrow TUUT \mid TUU \mid UUT \mid UU$$

$$T \rightarrow AT \mid A$$

$$U \rightarrow BUB \mid A$$

$$A \rightarrow a$$

$$B \rightarrow b$$

$$\}$$

(iv) **Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1 B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1} B_n, A_{n-1} \rightarrow A_{n-2} B_{n-1}, \dots, A_2 \rightarrow B_1 B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. —

$$P = \{$$

$$S \rightarrow TS_1 \mid TS_3 \mid US_2 \mid UU$$

$$S_1 \rightarrow US_2$$

$$S_2 \rightarrow UT$$

$$S_3 \rightarrow UU$$

$$T \rightarrow AT \mid a$$

$$U \rightarrow BU_1 \mid a$$

$$U_1 \rightarrow UB$$

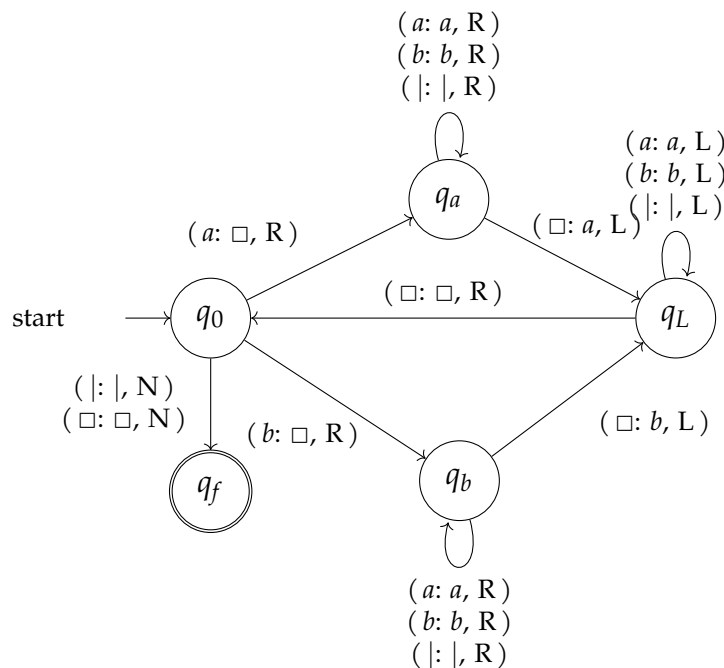
$$A \rightarrow a$$

$$B \rightarrow b$$

$$\}$$
**66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 4**

Wir betrachten die Turingmaschine  $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ . Hierbei ist die Zustandsmenge  $Q = \{q_0, q_a, q_b, q_L, q_f\}$  mit Startzustand  $q_0$  und akzeptierenden Zuständen  $F = \{q_f\}$ . Das Eingabealphabet ist  $\Sigma = \{a, b, |\}$ <sup>2</sup> das Bandalphabet ist  $\Gamma = \Sigma \cup \{\square\}$  mit Blank-Zeichen  $\square$  für leeres Feld. Die Übergangsfunktion  $\delta : Z \times \Gamma \rightarrow Z \times \Gamma \times \{L, R, N\}$ , wobei der Schreib-Lese-Kopf mit L nach links, mit N nicht und mit R nach rechts bewegt wird, ist durch folgende Tabelle gegeben (bspw. ist  $\delta(q_0, a) = (q_a, \square, R)$ ):

<sup>2</sup>In der Angabe ist das Trennzeichen ein „\$“. Wir verwenden stattdessen ein „|“, denn „\$“ ist eine Tex-Sonderzeichen und müsste deshalb ständig besonders behandelt werden.



Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Aj54q4rd9](http://flaci.com/Aj54q4rd9)

- (a) Die Notation  $(v, q, aw)$  beschreibt eine Konfiguration der Turingmaschine: der interne Zustand ist  $q$ , der Schreib-Lesekopf steht auf einem Feld mit  $a \in \Gamma$ , rechts vom Schreib-Lesekopf steht  $w \in \Gamma^*$ , links vom Schreib-Lesekopf steht  $v \in \Gamma^*$ .

Vervollständigen Sie die Folge von Konfigurationen, die die Turingmaschine bei Eingabe  $ab|$  bis zum Erreichen des Zustands  $q_f$  durchläuft. Sie können auch Ihre eigene Notation zur Darstellung von Konfigurationen verwenden.

$(\square, q_0, ab|) \vdash$   
 $(\square, q_a, \square b|) \vdash$   
 $(\square, q_a, b|) \vdash$   
 $\dots$

Lösungsvorschlag

$(\square, q_0, ab|) \vdash$   
 $(\square, q_a, \square b|) \vdash$   
 $(\square, q_a, b|) \vdash$   
 $(b, q_a, |) \vdash$   
 $(b|, q_L, a) \vdash$   
 $(b, q_L, |a) \vdash$   
 $(\square, q_L, \square b|a) \vdash$   
 $(\square, q_b, \square b|a) \vdash$   
 $(\square, q_b, \square|a) \vdash$   
 $(\square, q_b, |a) \vdash$   
 $(|, q_b, a) \vdash$   
 $(|a, q_L, b) \vdash$   
 $(|, q_L, ab) \vdash$

$$\begin{array}{l} (\square, q_L, |ab) \vdash \\ (\square, q_0, \square |ab) \vdash \\ (\square, q_f, |ab) \end{array}$$

- (b) Sei  $w \in \{a, b\}^*$  beliebig. Mit welchem Bandinhalt terminiert die Turingmaschine bei Eingabe von  $w$ ? Geben Sie auch eine kurze Begründung an.

Lösungsvorschlag

Die Turingmaschine terminiert bei alle möglichen Wörtern  $w \in \{a, b\}^*$ , auch bei dem leeren Wort vor  $|$ . Die Turing-Maschine verschiebt alle  $a$ 's und  $b$ 's vor dem Trennzeichen  $|$  nach rechts. Ist das Trennzeichen  $|$  schließlich das erste Zeichen von links gesehen, dann terminiert die Maschine.

## 66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 5

Wir betrachten das Behälterproblem BEHAELTER. Gegeben ist eine Menge von  $k \in \mathbb{N}$  Behältern, die jeweils ein Fassungsvermögen der Größe  $b \in \mathbb{N}$  haben. Gegeben sind weiterhin  $n$  Objekte mit jeweiligen Größen  $a_1, \dots, a_n$ . Gesucht ist eine Zuordnung der  $n$  Objekte auf die  $k$  Behälter, sodass keiner der Behälter überläuft.

Formal sind Instanzen des Behälterproblems BEHAELTER durch Tupel  $(k, a_1, \dots, a_n)$  gegeben, die wie folgt zu interpretieren sind:

- $k \in \mathbb{N}$  steht für eine Anzahl von Behältern.
- Jeder Behälter hat ein Fassungsvermögen von  $b \in \mathbb{N}$ .
- Die  $a_i$  stehen für die jeweiligen Größen von  $n$  Objekten.

Zuordnungen von Objekten zu Behältern geben wir durch eine Funktion  $v$  an, wobei  $v(j) = i$  wenn das  $j$ -te Objekt (mit Größe  $a_j$ ) dem  $i$ -ten Behälter zugeordnet wird.

$(k, b, a_1, \dots, a_n)$  ist eine JA-Instanz von BEHAELTER, wenn es eine Zuordnung  $v$  von Objekten auf Behälter ( $v: [1; n] \rightarrow [1; k]$ ) gibt, die sicherstellt, dass kein Behälter überläuft:

$$(k, b, a_1, \dots, a_n) \in \text{BEHAELTER} \iff (\exists v: [1; n] \rightarrow [1; k]. \forall i. S(a_i) \leq b) \iff \exists v. \sum_{j: v(j)=i} a_j \leq b$$

Wir betrachten auch das modifizierte Problem GERADEBEHAELTER. Instanzen von GERADEBEHAELTER tragen die zusätzliche Einschränkung, dass alle  $a_i$  gerade (durch zwei teilbar) sein müssen.

- (a) Warum ist sowohl BEHAELTER  $\in$  NP als auch GERADEBEHAELTER  $\in$  NP?
- (b) Beweisen Sie, dass das Problem BEHAELTER auf das Problem GERADEBEHAELTER in polynomieller Zeit reduzierbar ist.
- (c) BEHAELTER ist NP-vollständig. Begründen Sie, was obige Reduktion für die Komplexität von GERADEBEHAELTER bedeutet. BEHAELTER ist NP-vollständig. Begründen Sie, was obige Reduktion für die Komplexität von GERADEBEHAELTER bedeutet.

**66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 6**

Aus dem Känguru-Wettbewerb 2017 — Klassenstufen 3 und 4.

Luna hat für den Kuchenbasar Muffins mitgebracht: 10 Apfelmuffins, 18 Nussmuffins, 12 Schokomuffins und 9 Blaubeermuffins. Sie nimmt immer 3 verschiedene Muffins und legt sie auf einen Teller. Welches ist die kleinste Zahl von Muffins, die dabei übrig bleiben können?

A: 1, B: 3, C: 4, D: 7, E: 8

- (a) Geben Sie die richtige Antwort auf die im Känguru-Wettbewerb gestellte Frage und begründen Sie sie.

Lösungsvorschlag

4<sup>a</sup>

<sup>a</sup><https://www.youtube.com/watch?v=ceJW9kAp1VY>

- (b) Lunas Freundin empfiehlt den jeweils nächsten Teller immer aus den drei aktuell häufigsten Muffinsorten zusammenzustellen. Leiten Sie aus dieser Idee einen effizienten GreedyAlgorithmus her, der die Fragestellung für beliebige Anzahlen von Muffins löst (nach wie vor soll es nur vier Sorten und je drei pro Teller geben). Skizzieren Sie in geeigneter Form, wie Ihr Algorithmus die Beispielinstantz von oben richtig löst.

Lösungsvorschlag

```
public static int berechneRest4Sorten3ProTeller() {
 int[] muffins = new int[] { 10, 18, 12, 9 };
 int n = muffins.length;
 sortiere(muffins);

 // Wir nehmen uns 3 verschiedene Muffins solange, wie die dritthäufigste
 // Muffinsorte noch Muffins hat.
 while (muffins[n - 3] > 0) {
 muffins[n - 1]--;
 muffins[n - 2]--;
 muffins[n - 3]--;
 sortiere(muffins);
 }
 return berechneGesamtzahl(muffins);
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2019/fruehjahr/Muffin.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2019/fruehjahr/Muffin.java)

- (c) Beschreiben Sie eine mögliche und sinnvolle Verallgemeinerung Ihrer Lösung auf  $n$  Muffinsorten und  $k$  Muffins pro Teller für  $n > 4$  und  $k > 3$ .



```

public static int berechneRestAllgemein(int[] muffins, int k) {
 int n = muffins.length;
 sortiere(muffins);
 while (muffins[n - k] > 0) {
 for (int i = 1; i <= k; i++) {
 muffins[n - i]--;
 }
 sortiere(muffins);
 }
 return berechneGesamtzahl(muffins);
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2019/fruehjahr/Muffin.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2019/fruehjahr/Muffin.java)

- (d) Diskutieren Sie, wie man die Korrektheit des Greedy-Algorithmus zeigen könnte, also dass er tatsächlich immer eine optimale Lösung findet. Ein kompletter, rigoroser Beweis ist nicht verlangt.

## 66115 / 2019 / Frühjahr / Thema 2 / Aufgabe 1

Gegeben sei eine unsortierte Liste von  $n$  verschiedenen natürlichen Zahlen. Das  $k$ -kleinste Element ist das Element, das größer als genau  $k - 1$  Elemente der Liste ist.

- (a) Geben Sie einen Algorithmus mit Laufzeit  $\mathcal{O}(n \cdot \log n)$  an, um das  $k$ -kleinste Element zu berechnen.

Lösungsvorschlag

*a*

---

<sup>a</sup><https://en.wikipedia.org/wiki/Quickselect>

- (b) Gegeben sei nun ein Algorithmus  $A$ , der den Median einer unsortierten Liste von  $n$  Zahlen in  $\mathcal{O}(n)$  Schritten berechnet. Nutzen Sie Algorithmus  $A$  um einen Algorithmus  $B$  anzugeben, welcher das  $k$ -kleinste Element in  $\mathcal{O}(n)$  Schritten berechnet. Argumentieren Sie auch, dass der Algorithmus die gewünschte Laufzeit besitzt.

Lösungsvorschlag

*a*

---

<sup>a</sup>[https://en.wikipedia.org/wiki/Median\\_of\\_medians](https://en.wikipedia.org/wiki/Median_of_medians)

- (c) Geben Sie einen Algorithmus an, der für alle  $i = 1 \dots, \lfloor n/k \rfloor$  das  $i \cdot k$ -kleinste Element berechnet. Die Laufzeit Ihres Algorithmus sollte  $\mathcal{O}(n \cdot \log(n/k))$  sein. Sie dürfen weiterhin Algorithmus  $A$ , wie in Teilaufgabe (b) beschrieben, nutzen.

## 66115 / 2019 / Herbst / Thema 1 / Aufgabe 4

Betrachten Sie die folgenden Probleme:

### SAT

**Gegeben:** Aussagenlogische Formel  $F$  in KNF

**Frage:** Gibt es mindestens eine erfüllende Belegung für  $F$ ?

### DOPPELSAT

**Gegeben:** Aussagenlogische Formel  $F'$  in KNF

**Frage:** Gibt es mindestens eine erfüllende Belegung für  $F$ , in der mindestens zwei Literale pro Klausel wahr sind?

(a) Führen Sie eine polynomielle Reduktion von SAT auf DOPPELSAT durch.

Lösungsvorschlag

<https://courses.cs.washington.edu/courses/csep531/09wi/handouts/sol4.pdf>

DOUBLE-SAT is in NP. The polynomial size certificate consists of two assignments  $f_1$  and  $f_2$ . First, the verifier verifies if  $f_1 \neq f_2$ . Then, it verifies if both assignments satisfy  $\phi$  by substituting the values for the variables and evaluate the clauses of  $\phi$ . Both checks can be done in linear time. DOUBLE-SAT is NP-hard. We give a reduction from SAT. Given an instance  $\phi$  of SAT which is a CNF formula of  $n$  variables  $x_1, x_2, \dots, x_n$ , we construct a new variable  $x_{n+1}$  and let  $\psi = \phi \wedge (x_{n+1} \vee \neg x_{n+1})$  be the corresponding instance of DOUBLE-SAT. We claim that  $\phi$  has a satisfying assignment iff  $\psi$  has at least two satisfying assignments. On one hand, if  $\phi$  has a satisfying assignment  $f$ , we can obtain two distinct satisfying assignments of  $\psi$  by extending  $f$  with  $x_{n+1} = T$  and  $x_{n+1} = F$  respectively. On the other hand, if  $\psi$  has at least two satisfying assignments then the restriction of any of them to the set  $x_1, x_2, \dots, x_n$  is a satisfying assignment for  $\phi$ . Thus, DOUBLE-SAT is NP-complete.

<https://cs.stackexchange.com/questions/6371/proving-double-sat-is-np-complete>

Here is one solution:

Clearly Double-SAT belongs to NP, since a NTM can decide Double-SAT as follows: On a Boolean input formula  $\phi(x_1, \dots, x_n)$ , nondeterministically guess 2 assignments and verify whether both satisfy  $\phi$ .

To show that Double-SAT is NP-Complete, we give a reduction from SAT to Double-SAT, as follows:

On input  $\phi(x_1, \dots, x_n)$ :

1. Introduce a new variable  $y$ . 2. Output formula  $\phi'(x_1, \dots, x_n, y) = \phi(x_1, \dots, x_n) \wedge (y \vee \bar{y})$ .

If  $\phi(x_1, \dots, x_n)$  belongs to SAT, then  $\phi$  has at least 1 satisfying assignment, and therefore  $\phi'(x_1, \dots, x_n, y)$  has at least 2 satisfying assignments as we can

satisfy the new clause  $(y \vee \bar{y})$  by assigning either  $y = 1$  or  $y = 0$  to the new variable  $y$ , so  $\phi'(x_1, \dots, x_n, y) \in \text{Double-SAT}$ .

On the other hand, if  $\phi(x_1, \dots, x_n) \notin \text{SAT}$ , then clearly  $\phi'(x_1, \dots, x_n, y) = \phi(x_1, \dots, x_n) \wedge (y \vee \bar{y})$  has no satisfying assignment either, so  $\phi'(x_1, \dots, x_n, y) \notin \text{Double-SAT}$ .

Therefore,  $\text{SAT} \leq_p \text{Double-SAT}$ , and hence Double-SAT is NP-Complete.

(b) Zeigen Sie, dass DOPPELSAT NP-vollständig ist.

## 66115 / 2019 / Herbst / Thema 1 / Aufgabe 5

In der folgenden Aufgabe soll ein Feld A von ganzen Zahlen aufsteigend sortiert werden. Das Feld habe n Elemente A[0] bis A[n-1]. Der folgende Algorithmus (in der Notation des Informatik-Duden) sei gegeben:

```

procedure quicksort(links, rechts : integer)
var i, j, x : integer;
begin
 i := links;
 j := rechts;
 if j > i then begin
 x := A[links];
 repeat
 while A[i] < x do i := i+1;
 while A[j] > x do j := j-1;
 if i < j then begin
 tmp := A[i]; A[i] := A[j]; A[j] := tmp;
 i := i+1; j := j-1;
 end
 until i > j;
 quicksort(links, j);
 quicksort(i, rechts);
 end
end

```

### Umsetzung in Java:

```

public static void quicksort(int[] A, int links, int rechts) {
 System.out.println("quick");
 int i = links;
 int j = rechts;
 if (j > i) {
 int x = A[links];
 do {
 while (A[i] < x) {
 i = i + 1;
 }
 while (A[j] > x) {
 j = j - 1;
 }
 if (i <= j) {

```

```
 int tmp = A[i];
 A[i] = A[j];
 A[j] = tmp;
 i = i + 1;
 j = j - 1;
}
// Java verfügt über keine do-until Schleife.
// Wir verwenden eine do-while-Schleife mit einem umgedrehten Test
// unit i > j -> while (i <= j)
} while (i <= j);
quicksort(A, links, j);
quicksort(A, i, rechts);
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2019/herbst/QuickSort.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66115/jahr_2019/herbst/QuickSort.java)

Der initiale Aufruf der Prozedur lautet:

`quicksort(0,n-1)`

- (a) Sortieren Sie das folgende Feld der Länge 7 mittels des Algorithmus. Notieren Sie jeweils alle Aufrufe der Prozedur `quicksort` mit den konkreten Parameterwerten. Geben Sie zudem für jeden Aufruf der Prozedur den Wert des in Zeile 7 gewählten Elements an.

27 13 21 3 6 17 44 42

Lösungsvorschlag

```
quicksort(0, 6)
27 32 3 6 17 44 42
x: 27
quicksort(0, 2)
17 6 3 32 27 44 42
x: 17
quicksort(0, 1)
3 6 17 32 27 44 42
x: 3
quicksort(0, -1)
3 6 17 32 27 44 42
quicksort(1, 1)
3 6 17 32 27 44 42
quicksort(2, 2)
3 6 17 32 27 44 42
quicksort(3, 6)
3 6 17 32 27 44 42
x: 32
quicksort(3, 3)
3 6 17 27 32 44 42
quicksort(4, 6)
3 6 17 27 32 44 42
x: 32
quicksort(4, 3)
3 6 17 27 32 44 42
quicksort(5, 6)
3 6 17 27 32 44 42
x: 44
```

```
quicksort(5, 5)
3 6 17 27 32 42 44
quicksort(6, 6)
3 6 17 27 32 42 44
3 6 17 27 32 42 44
```

- (b) Angenommen, die Bedingung  $j > i$  in Zeile 6 des Algorithmus wird ersetzt durch die Bedingung  $j \geq i$ . Ist der Algorithmus weiterhin korrekt? Begründen Sie Ihre Antwort.

geht, dauert aber länger

```
quicksort(0, 6)
27 32 3 6 17 44 42
x: 27
quicksort(0, 2)
17 6 3 32 27 44 42
x: 17
quicksort(0, 1)
3 6 17 32 27 44 42
x: 3
quicksort(0, -1)
3 6 17 32 27 44 42
quicksort(1, 1)
3 6 17 32 27 44 42
x: 6
quicksort(1, 0)
3 6 17 32 27 44 42
quicksort(2, 1)
3 6 17 32 27 44 42
quicksort(2, 2)
3 6 17 32 27 44 42
x: 17
quicksort(2, 1)
3 6 17 32 27 44 42
quicksort(3, 2)
3 6 17 32 27 44 42
quicksort(3, 6)
3 6 17 32 27 44 42
x: 32
quicksort(3, 3)
3 6 17 27 32 44 42
x: 27
quicksort(3, 2)
3 6 17 27 32 44 42
quicksort(4, 3)
3 6 17 27 32 44 42
quicksort(4, 6)
3 6 17 27 32 44 42
x: 32
quicksort(4, 3)
3 6 17 27 32 44 42
quicksort(5, 6)
3 6 17 27 32 44 42
```

```

x: 44
quicksort(5, 5)
3 6 17 27 32 42 44
x: 42
quicksort(5, 4)
3 6 17 27 32 42 44
quicksort(6, 5)
3 6 17 27 32 42 44
quicksort(6, 6)
3 6 17 27 32 42 44
x: 44
quicksort(6, 5)
3 6 17 27 32 42 44
quicksort(7, 6)
3 6 17 27 32 42 44
3 6 17 27 32 42 44

```

- (c) Angenommen, die Bedingung  $i \leq j$  in Zeile 11 des Algorithmus wird ersetzt durch die Bedingung  $i < j$ . Ist der Algorithmus weiterhin korrekt? Begründen Sie Ihre Antwort.

Lösungsvorschlag

bleibt hängen

```

quicksort(0, 6)
27 32 3 6 17 44 42
x: 27
quicksort(0, 2)
17 6 3 32 27 44 42
x: 17
quicksort(0, 1)
3 6 17 32 27 44 42
x: 3

```

- (d) Wie muss das Feld A gestaltet sein, damit der Algorithmus mit der geringsten Anzahl von Schritten terminiert? Betrachten Sie dazu vor allem Zeile 7. Begründen Sie Ihre Antwort und geben Sie ein Beispiel.

Lösungsvorschlag

Im Worst Case (schlechtesten Fall) wird das Pivotelement stets so gewählt, dass es das größte oder das kleinste Element der Liste ist. Dies ist etwa der Fall, wenn als Pivotelement stets das Element am Ende der Liste gewählt wird und die zu sortierende Liste bereits sortiert vorliegt. Die zu untersuchende Liste wird dann in jedem Rekursionsschritt nur um eins kleiner und die Zeitkomplexität wird beschrieben durch  $\mathcal{O}(n^2)$ . Die Anzahl der Vergleiche ist in diesem Fall  $\frac{n \cdot (n+1)}{2} - 1 = \frac{n^2}{2} + \frac{n}{2} - 1$ .

Die Länge der jeweils längeren Teilliste beim rekursiven Aufrufe ist nämlich im Schnitt  $\frac{2}{n} \sum_{i=\frac{n}{2}}^{n-1} i = \frac{3}{4}n - \frac{2}{4}$  und die Tiefe der Rekursion damit in  $\mathcal{O}(\log(n))$ .

Im Average Case ist die Anzahl der Vergleiche etwa  $2 \cdot \log(2) \cdot (n+1) \cdot \log_2(n) \approx$

$$1,39 \cdot (n+1) \cdot \log_2(n).$$

- (e) Die rekursiven Aufrufe in den Zeilen 16 und 17 des Algorithmus werden zur Laufzeit des Computers auf dem Stack verwaltet. Die Anzahl der Aufrufe von quicksort auf dem Stack abhängig von der Eingabegröße  $n$  sei mit  $s(n)$  bezeichnet. Geben Sie die Komplexitätsklasse von  $s(n)$  für den schlimmsten möglichen Fall an. Begründen Sie Ihre Antwort.

## 66115 / 2019 / Herbst / Thema 1 / Aufgabe 6

- (a) Sortieren Sie die unten angegebenen Funktionen der O-Klassen  $\mathcal{O}(a(n))$ ,  $\mathcal{O}(b(n))$ ,  $\mathcal{O}(c(n))$ ,  $\mathcal{O}(d(n))$  und  $\mathcal{O}(e(n))$  bezüglich ihrer Teilmengenbeziehungen. Nutzen Sie ausschließlich die echte Teilmenge  $\subset$  sowie die Gleichheit  $=$  für die Beziehung zwischen den Mengen. Folgendes Beispiel illustriert diese Schreibweise für einige Funktionen  $f_1$  bis  $f_5$  (diese haben nichts mit den unten angegebenen Funktionen zu tun):<sup>3</sup>

$$\mathcal{O}(f_4(n)) \subset \mathcal{O}(f_3(n)) = \mathcal{O}(f_5(n)) \subset \mathcal{O}(f_1(n)) = \mathcal{O}(f_2(n))$$

Die angegebenen Beziehungen müssen weder bewiesen noch begründet werden.

- $a(n) = n^2 \cdot \log_2(n) + 42$
- $b(n) = 2^n + n^4$
- $c(n) = 2^{2 \cdot n}$
- $d(n) = 2^{n+3}$
- $e(n) = \sqrt{n^5}$

Lösungsvorschlag

$$\begin{array}{ll} a(n) = n^2 \cdot \log_2(n) + 42 & = n \\ b(n) = 2^n + n^4 & = 2^n \\ c(n) = 2^{2 \cdot n} & = 2^{2 \cdot n} \\ d(n) = 2^{n+3} & = 2^n \\ e(n) = \sqrt{n^5} & \end{array}$$

$$\mathcal{O}(a(n)) \subset \mathcal{O}(e(n)) \subset \mathcal{O}(b(n)) = \mathcal{O}(d(n)) \subset \mathcal{O}(c(n))$$

<sup>3</sup>[http://www.s-inf.de/Skripte/DaStru.2012-SS-Katoen.\(KK\).Klausur1MitLoesung.pdf](http://www.s-inf.de/Skripte/DaStru.2012-SS-Katoen.(KK).Klausur1MitLoesung.pdf)



$$\mathcal{O}(n^2 \cdot \log_2(n) + 42) \subset \mathcal{O}(\sqrt{n^5}) \subset \mathcal{O}(2^n + n^4) = \mathcal{O}(2^{n+3}) \subset \mathcal{O}(2^{2 \cdot n})$$

- (b) Beweisen Sie die folgenden Aussagen formal nach den Definitionen der O-Notation oder widerlegen Sie sie.

(i)  $\mathcal{O}(n \cdot \log_2 n) \subseteq \mathcal{O}(n \cdot (\log_2 n)^2)$

Lösungsvorschlag

Die Aussage gilt. Für  $n \geq 16$  haben wir

$$(\log_2 n)^2 \leq n \Leftrightarrow \log_2 n \leq \sqrt{n}$$

und dies ist eine wahre Aussage für  $n \geq 16$ . Also gilt die Aussage mit  $n_0 = 16$  und  $c = 1$ .

(ii)  $2^{(n+1)} \in \mathcal{O}(n \cdot \log_2 n)$

- (c) Bestimmen Sie eine asymptotische Lösung (in  $\Theta$ -Schreibweise) für die folgende Rekursionsgleichung:

(i)  $T(n) = 4 \cdot T(\frac{n}{2}) + n^2$

(ii)  $T(n) = T(\frac{n}{2}) + \frac{n}{2}n^2 + n$

## 66115 / 2019 / Herbst / Thema 1 / Aufgabe 7

Gegeben sei die folgende Realisierung von binären Bäumen (in einer an Java angelehnten Notation):

```
class Node {
 Node left, right;
 int value;
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2019/herbst/Node.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2019/herbst/Node.java)

- (a) Beschreiben Sie in möglichst wenigen Worten, was die folgende Methode `foo` auf einem nicht-leeren binären Baum berechnet.

```
int foo(Node node) {
 int b = node.value;
 if (b < 0) {
 b = -1 * b;
 }
 if (node.left != null) {
 b = b + foo(node.left);
 }
 if (node.right != null) {
 b = b + foo(node.right);
 }
 return b;
}
```

## Lösungsvorschlag

Die Methode `foo(Node node)` berechnet die Summe aller Knoten des Unterbaums des als Parameter übergebenen Knotens `node`. Der Schlüsselwert des Knotens `node` selbst wird in die Summenberechnung mit einbezogen.

- (b) Die Laufzeit der Methode `foo(tree)` ist linear in  $n$ , der Anzahl von Knoten im übergebenen Baum `tree`. Begründen Sie kurz, warum `foo(tree)` eine lineare Laufzeit hat.

## Lösungsvorschlag

Die Methode `foo(Node node)` wird pro Knoten des Baums `tree` genau einmal aufgerufen. Es handelt sich um eine rekursive Methode, die auf den linken und rechten Kindknoten aufgerufen wird. Hat ein Knoten keine Kinder mehr, dann wird die Methode nicht aufgerufen.

- (c) Betrachten Sie den folgenden Algorithmus für nicht-leere, binäre Bäume. Beschreiben Sie in möglichst wenigen Worten die Wirkung der Methode `magic(tree)`. Welche Rolle spielt dabei die Methode `max`?

```
void magic(Node node) {
 Node m = max(node);
 if (m.value > node.value) {
 // Werte von m und node vertauschen
 int tmp = m.value;
 m.value = node.value;
 node.value = tmp;
 }
 if (node.left != null)
 magic(node.left);
 if (node.right != null)
 magic(node.right);
}

Node max(Node node) {
 Node max = node;
 if (node.left != null) {
 Node tmp = max(node.left);
 if (tmp.value > max.value)
 max = tmp;
 }
 if (node.right != null) {
 Node tmp = max(node.right);
 if (tmp.value > max.value)
 max = tmp;
 }
 return max;
}
```

Methode `magic(tree)` vertauscht für jeden Knoten des Unterbaums den Wert mit dem Maximal-Knoten. Die Methode `max` liefert dabei den Knoten mit der größten Schlüsselwert im Unterbaum des übergebenen Knoten (sich selbst eingeschlossen).

- (d) Geben Sie in Abhängigkeit von  $n$ , der Anzahl von Knoten im übergebenen Baum `tree`, jeweils eine Rekursionsgleichung für die asymptotische Best-Case-Laufzeit ( $B(n)$ ) und Worst-Case-Laufzeit ( $W(n)$ ) des Aufrufs `magic(tree)` sowie die entsprechende Komplexitätsklasse ( $\Theta$ ) an. Begründen Sie Ihre Antwort.

Hinweis: Überlegen Sie, ob die Struktur des übergebenen Baumes Einfluss auf die Laufzeit hat. Die lineare Laufzeit von `max(t)` in der Anzahl der Knoten des Baumes `t` darf vorausgesetzt werden.

## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 1

Gesucht ist eine Turing-Maschine mit genau einem beidseitig unendlichen Band, die die Funktion  $f : \mathbb{N} \rightarrow \mathbb{N}$  mit  $f(x) = 3x$  berechnet. Zu Beginn der Berechnung steht die Eingabe binär codiert auf dem Band, wobei der Kopf auf die linkeste Ziffer (most significant bit) zeigt. Am Ende der Berechnung soll der Funktionswert binär codiert auf dem Band stehen, wobei der Kopf auf ein beliebiges Feld zeigen darf.

- (a) Beschreiben Sie zunächst in Worten die Arbeitsweise Ihrer Maschine.

Lösungsvorschlag

$$13 \cdot 3 = 0b1101 \cdot 0b11 = 39 = 0b100111:$$

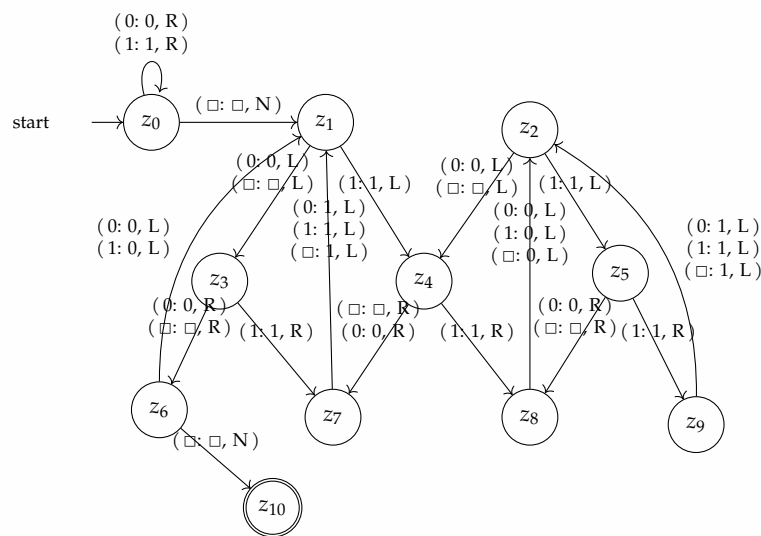
$$\begin{array}{r}
 1 \ 1 \ 0 \ 1 \\
 1 \ 1 \ 0 \ 1 \\
 \hline
 1 \ 1 \\
 1 \ 0 \ 0 \ 1 \ 1 \ 1
 \end{array}$$

Die entworfene Turingmaschine imitierte der Vorgehensweise beim schriftlichen Multiplizieren. Die Maschine geht zunächst an das Leerzeichen am rechten Ende des Eingabewortes. Die Maschine bewegt sich nun zwei Schritte nach links und liest die Zahlen ein und addiert sie. Schließlich bewegt sich die Maschine einen Schritt nach rechts und schreibt das Ergebnis der Addition. Dabei wird das Eingabewort überschrieben allmählich überschrieben.

- (b) Geben Sie dann das kommentierte Programm der Turing-Maschine an und erklären Sie die Bedeutung der verwendeten Zustände.

- $z_0$  An das rechte Ende des Eingabewortes gehen
- $z_1$  Übertrag 0
- $z_2$  Übertrag 1
- $z_3$  1. Additionsschritt: +0
- $z_4$  1. Additionsschritt: +1
- $z_5$  1. Additionsschritt: +2
- $z_6$  2. Additionsschritt: +0
- $z_7$  2. Additionsschritt: +1
- $z_8$  2. Additionsschritt: +2
- $z_9$  2. Additionsschritt: +3
- $z_{10}$  Endzustand

Nicht sehr übersichtlich hier: Im Anhang findet sich die JSON-Datei für flaci.com



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/AhjkW50kg](http://flaci.com/AhjkW50kg)

## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 6

Der Hauptsatz der Laufzeitfunktionen ist bekanntlich folgendermaßen definiert:

**1. Fall:**  $T(n) \in \Theta(n^{\log_b a})$

falls  $f(n) \in \mathcal{O}(n^{\log_b a - \varepsilon})$  für  $\varepsilon > 0$

**2. Fall:**  $T(n) \in \Theta(n^{\log_b a} \cdot \log n)$

$$\text{falls } f(n) \in \Theta(n^{\log_b a})$$

**3. Fall:**  $T(n) \in \Theta(f(n))$

falls  $f(n) \in \Omega(n^{\log_b a + \varepsilon})$  für  $\varepsilon > 0$  und ebenfalls für ein  $c$  mit  $0 < c < 1$  und alle hinreichend großen  $n$  gilt:  $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$

Bestimmen und begründen Sie formal mit Hilfe dieses Satzes welche Komplexität folgende Laufzeitfunktionen haben.

(a)  $T(n) = 8 \cdot T(\frac{n}{2}) + 5n^2$

Lösungsvorschlag

**Allgemeine Rekursionsgleichung:**

$$T(n) = a \cdot T(\frac{n}{b}) + f(n)$$

**Anzahl der rekursiven Aufrufe (a):**

8

**Anteil Verkleinerung des Problems (b):**

um  $\frac{1}{2}$  also  $b = 2$

**Laufzeit der rekursiven Funktion (f(n)):**

$$5n^2$$

**Ergibt folgende Rekursionsgleichung:**

$$T(n) = 8 \cdot T(\frac{n}{2}) + 5n^2$$

**1. Fall:**  $f(n) \in \mathcal{O}(n^{\log_b a - \varepsilon})$ :

für  $\varepsilon = 4$ :

$$f(n) = 5n^2 \in \mathcal{O}(n^{\log_2 8 - 4}) = \mathcal{O}(n^{\log_2 4}) = \mathcal{O}(n^2)$$

**2. Fall:**  $f(n) \in \Theta(n^{\log_b a})$ :

$$f(n) = 5n^2 \notin \Theta(n^{\log_2 8}) = \Theta(n^3)$$

**3. Fall:**  $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ :

$$f(n) = 5n^2 \notin \mathcal{O}(n^{\log_2 8 + \varepsilon})$$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

(b)  $T(n) = 9 \cdot T(\frac{n}{3}) + 5n^2$

**Allgemeine Rekursionsgleichung:**

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

**Anzahl der rekursiven Aufrufe ( $a$ ):**

9

**Anteil Verkleinerung des Problems ( $b$ ):**

um  $\frac{1}{3}$  also  $b = 3$

**Laufzeit der rekursiven Funktion  $(f(n))$ :**

$$5n^2$$

**Ergibt folgende Rekursionsgleichung:**

$$T(n) = 9 \cdot T\left(\frac{n}{3}\right) + 5n^2$$

**1. Fall:**  $f(n) \in \mathcal{O}(n^{\log_b a - \varepsilon})$ :

$$f(n) = 5n^2 \notin \mathcal{O}(n^{\log_3 9 - \varepsilon}) \text{ für } \varepsilon > 0$$

**2. Fall:**  $f(n) \in \Theta(n^{\log_b a})$ :

$$f(n) = 5n^2 \in \Theta(n^{\log_3 9}) = \Theta(n^2)$$

**3. Fall:**  $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ :

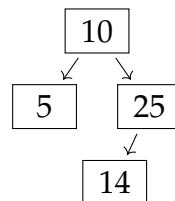
$$f(n) = 5n^2 \notin \mathcal{O}(n^{\log_3 9 + \varepsilon}) \text{ für } \varepsilon > 0$$

$$\Rightarrow T(n) \in \Theta(n^2 \cdot \log n)$$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 7

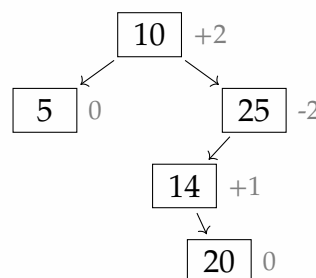
Fügen Sie (manuell) nacheinander die Zahlen 20, 31, 2, 17, 7 in folgenden AVL-Baum ein. Löschen Sie anschließend aus dem entstandenen Baum nacheinander 14 und 25.



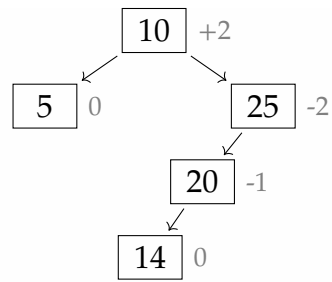
Zeichnen Sie jeweils direkt nach jeder einzelnen Operation zum Einfügen oder Löschen eines Knotens, sowie nach jeder elementaren Rotation den entstehenden Baum. Insbesondere sind evtl. anfallende Doppelrotationen in zwei Schritten darzustellen. Geben Sie zudem an jedem Knoten die Balancewerte an.

Lösungsvorschlag

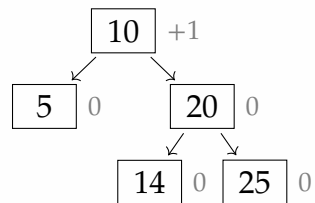
Nach dem Einfügen von „20“:



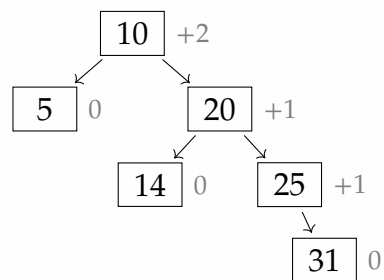
Nach der Linksrotation:



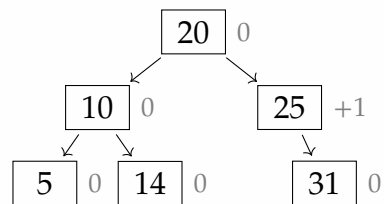
*Nach der Rechtsrotation:*



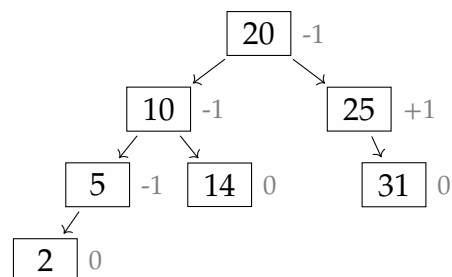
*Nach dem Einfügen von „31“:*



*Nach der Linksrotation:*

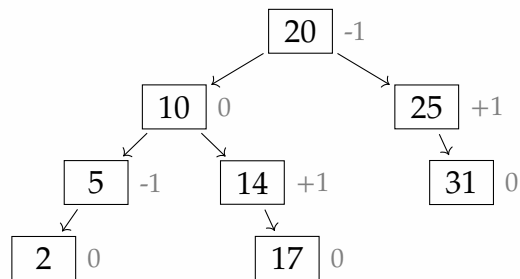


*Nach dem Einfügen von „2“:*

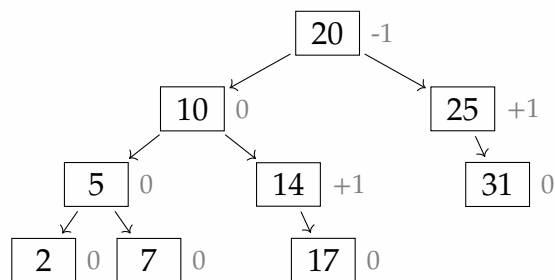




Nach dem Einfügen von „17“:

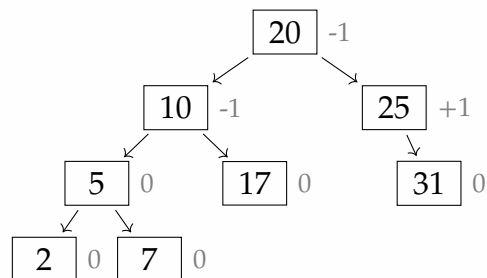


Nach dem Einfügen von „7“:

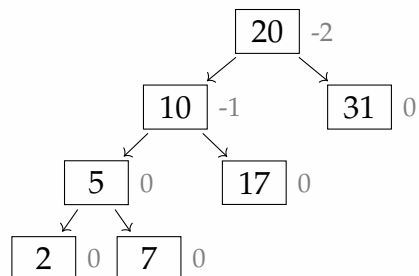


## Löschen

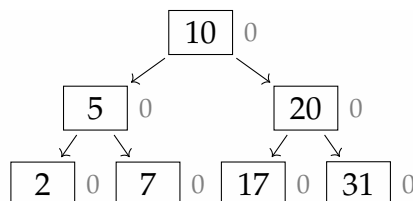
Nach dem Löschen von „14“:



Nach dem Löschen von „25“:

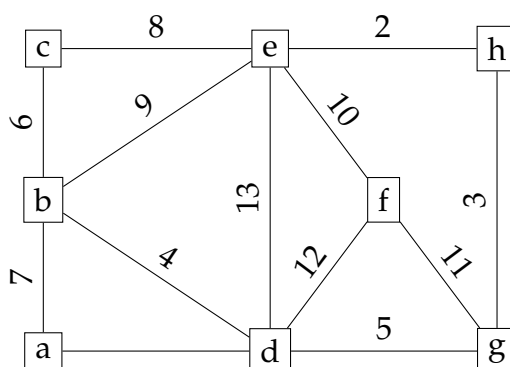


Nach der Rechtsrotation:



## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 8

Gegeben Sei der folgende ungerichtete Graph mit Kantengewichten.



- (a) Zeichnen Sie den (hier eindeutigen) minimalen Spannbaum.
- (b) Geben Sie sowohl für den Algorithmus von Jarník-Prim als auch für den Algorithmus von Kruskal die Reihenfolge an, in der die Kanten hinzugefügt werden. Starten Sie für den Algorithmus von Jarník-Prim beim Knoten  $a$ .

Übernehmen Sie den Graph auf Ihre Bearbeitung und füllen Sie hierzu das Tupel jeder Kante - aus dem MST in der Form  $(n, m)$  aus, wobei die Kante  $e$  vom Algorithmus von Jarník-Prim als  $n$ 'te Kante und vom Algorithmus von Kruskal als  $m$ 'te Kante hinzugefügt wird. Lassen Sie andere Tupel unausgefüllt.

## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 9

Verwenden Sie die Hashfunktion  $h(k, i) = (h'(k) + i^2) \bmod 11$  mit  $h'(k) = k \bmod 13$ , um die Werte 12, 29 und 17 in die folgende Hashtabelle einzufügen. Geben Sie zudem jeweils an, auf welche Zellen der Hashtabelle zugegriffen wird.

|   |   |   |    |   |   |   |   |   |    |    |
|---|---|---|----|---|---|---|---|---|----|----|
| 0 | 1 | 2 | 3  | 4 | 5 | 6 | 7 | 8 | 9  | 10 |
|   |   |   | 16 |   | 5 |   |   |   | 22 |    |

Lösungsvorschlag

### Einfügen des Wertes 12

$$h'(12) = 12 \bmod 13 = 12$$

$$h(12, 0) = 12 + 0^2 \bmod 11 = 1$$

|   |    |   |    |   |   |   |   |   |    |    |
|---|----|---|----|---|---|---|---|---|----|----|
| 0 | 1  | 2 | 3  | 4 | 5 | 6 | 7 | 8 | 9  | 10 |
|   | 12 |   | 16 |   | 5 |   |   |   | 22 |    |

**Einfügen des Wertes 29**

$$h'(29) = 29 \bmod 13 = 3$$

$$h(29, 0) = 3 + 0^2 \bmod 11 = 3 \text{ (belegt von 16)}$$

$$h(29, 1) = 3 + 1^2 \bmod 11 = 4$$

|   |    |   |    |    |   |   |   |   |    |    |
|---|----|---|----|----|---|---|---|---|----|----|
| 0 | 1  | 2 | 3  | 4  | 5 | 6 | 7 | 8 | 9  | 10 |
|   | 12 |   | 16 | 29 | 5 |   |   |   | 22 |    |

**Einfügen des Wertes 17**

$$h'(17) = 17 \bmod 13 = 4$$

$$h(17, 0) = 4 + 0^2 \bmod 11 = 4 \text{ (belegt von 29)}$$

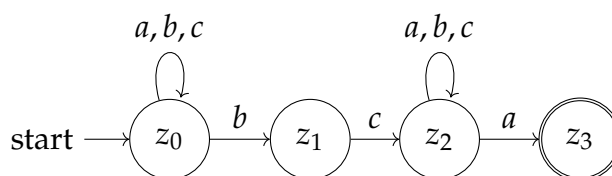
$$h(17, 1) = 4 + 1^2 \bmod 11 = 5 \text{ (belegt von 5)}$$

$$h(17, 2) = 4 + 2^2 \bmod 11 = 8$$

|   |    |   |    |    |   |   |   |    |    |    |
|---|----|---|----|----|---|---|---|----|----|----|
| 0 | 1  | 2 | 3  | 4  | 5 | 6 | 7 | 8  | 9  | 10 |
|   | 12 |   | 16 | 29 | 5 |   |   | 17 | 22 |    |

**66115 / 2020 / Frühjahr / Thema 1 / Aufgabe 2**

- (a) Es sei  $L \subseteq \{a, b, c\}^*$  die von dem folgenden nichtdeterministischen Automaten akzeptierte Sprache:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Apmac9bwc](https://flaci.com/Apmac9bwc)

Beschreiben Sie (in Worten) wie die Wörter aus der Sprache  $L$  aussehen.

Lösungsvorschlag

Alle Wörter der Sprache  $L$  enthalten die Symbolfolge  $bc$  und enden auf  $a$ . Am Anfang der Wörter und vor dem letzten  $a$  können beliebige Kombination aus  $a, b, c$  vorkommen.

- (b) Benutzen Sie die Potenzmengenkonstruktion, um einen deterministischen Automaten zu konstruieren, der zu dem Automaten aus Teil (a) äquivalent ist. (Berechnen Sie nur erreichbare Zustände.)

| Zustandsmenge           | Eingabe $a$             | Eingabe $b$             | Eingabe $c$        |
|-------------------------|-------------------------|-------------------------|--------------------|
| $Z_0 \{z_0\}$           | $Z_0 \{z_0\}$           | $Z_1 \{z_0, z_1\}$      | $Z_0 \{z_0\}$      |
| $Z_1 \{z_0, z_1\}$      | $Z_0 \{z_0\}$           | $Z_1 \{z_0, z_1\}$      | $Z_2 \{z_0, z_2\}$ |
| $Z_2 \{z_0, z_2\}$      | $Z_3 \{z_0, z_2, z_3\}$ | $Z_4 \{z_0, z_1, z_2\}$ | $Z_2 \{z_0, z_2\}$ |
| $Z_3 \{z_0, z_2, z_3\}$ | $Z_3 \{z_0, z_2, z_3\}$ | $Z_4 \{z_0, z_1, z_2\}$ | $Z_2 \{z_0, z_2\}$ |
| $Z_4 \{z_0, z_1, z_2\}$ | $Z_3 \{z_0, z_2, z_3\}$ | $Z_4 \{z_0, z_1, z_2\}$ | $Z_2 \{z_0, z_2\}$ |

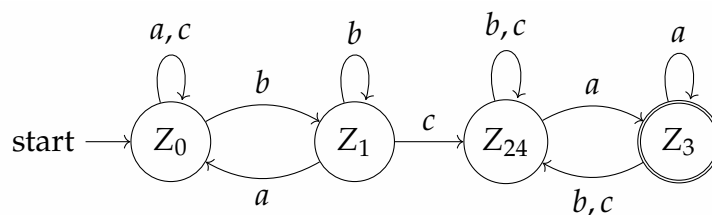
  

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/A5o6pho8c](http://flaci.com/A5o6pho8c)

- (c) Ist der resultierende deterministische Automat schon minimal? Begründen Sie Ihre Antwort.

Nein.  $Z_2 \{z_0, z_2\}$  und  $Z_4 \{z_0, z_1, z_2\}$  können vereinigt werden, da sie bei denselben Eingaben auf die selben Potenzenmengen übergehen.

| Zustandsmenge           | Eingabe $a$             | Eingabe $b$             | Eingabe $c$        |
|-------------------------|-------------------------|-------------------------|--------------------|
| $Z_2 \{z_0, z_2\}$      | $Z_3 \{z_0, z_2, z_3\}$ | $Z_4 \{z_0, z_1, z_2\}$ | $Z_2 \{z_0, z_2\}$ |
| $Z_3 \{z_0, z_2, z_3\}$ | $Z_3 \{z_0, z_2, z_3\}$ | $Z_4 \{z_0, z_1, z_2\}$ | $Z_2 \{z_0, z_2\}$ |
| $Z_4 \{z_0, z_1, z_2\}$ | $Z_3 \{z_0, z_2, z_3\}$ | $Z_4 \{z_0, z_1, z_2\}$ | $Z_2 \{z_0, z_2\}$ |



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ai1hox2b7

(d) Minimieren Sie den folgenden deterministischen Automaten:

### 66115 / 2020 / Frühjahr / Thema 1 / Aufgabe 3

(a) Entwerfen Sie eine kontextfreie Grammatik für die folgende kontextfreie Sprache über dem Alphabet  $\Sigma = \{a, b, c\}$ :

$$L = \{ a^{3n+2} w v c^n \mid n \in \mathbb{N}_0, 2 \cdot |w|_b = |v|_a \}$$

(Hierbei bezeichnet  $|u|_x$ , die Anzahl des Zeichens  $x$  in dem Wort  $u$ .)

Erklären Sie den Zweck der einzelnen Nichtterminale (Variablen) und der Grammatikregeln Ihrer Grammatik.

Lösungsvorschlag

$P = \{$

$$S \rightarrow aaaSc \mid aaaAc$$

$$A \rightarrow aaB$$

$$B \rightarrow bBaa \mid baa$$

$\}$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ghhs1xexw

(b) Betrachten Sie die folgende kontextfreie Grammatik

$$G = (\{A, B, C, D\}, \{a, b, c\}, P, A)$$

mit den Produktionen

$P = \{$

$$A \rightarrow AB \mid CD \mid a$$

$$B \rightarrow CC \mid c$$

$$C \rightarrow DC \mid CB \mid b$$

$$D \rightarrow DB \mid a$$

CYK-Algorithmus  
Ableitung (Kontextfreie  
Sprache)  
Kontextfreie Grammatik  
Kontextfreie Sprache

}

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gf7556jn2

Benutzen Sie den Algorithmus von Cocke-Younger-Kasami (CYK), um zu zeigen, dass das Wort *abcab* zu der von *G* erzeugten Sprache  $L(G)$  gehört.

Lösungsvorschlag

| a       | b | c | a   | b |
|---------|---|---|-----|---|
| A,D     | C | B | A,D | C |
| C       | C | - | C   |   |
| C,C     | A | - |     |   |
| A,A     | B |   |     |   |
| A,D,B,B |   |   |     |   |

$\Rightarrow abcab \in L(G)$

- (c) Finden Sie nun ein größtmögliches Teilwort von *abcab*, dass von keinem der vier Nichtterminale von *G* ableitbar ist.
- (d) Geben Sie eine Ableitung des Wortes *abcab* mit *G* an.

Lösungsvorschlag

$A \vdash AB \vdash ACC \vdash ACBC \vdash ACBDC \vdash aCBDC \vdash abBDC \vdash abcDC \vdash abcaC \vdash abcab$

- (e) Beweisen Sie, dass die folgende formale Sprache über  $Z = a,b$  nicht kontextfrei ist:  $L = a^n b^n$ .

## 66115 / 2020 / Frühjahr / Thema 2 / Aufgabe 3

- (a) Ist die folgende Sprache  $L_1 = \{ a^{n+2} b^{2n+1} \mid n \geq 2 \}$  über dem Alphabet  $\Sigma = \{a,b\}$  kontextfrei?

Falls ja, geben Sie eine kontextfreie Grammatik für  $L_1$ , an, falls nein, eine kurze Begründung (ein vollständiger Beweis ist hier nicht gefordert).

$L_1$  ist kontextfrei

$$G = (\{S, A, B\}, \{a, b\}, P, S)$$

$$P = \left\{ \right.$$

$$S \rightarrow aAbb$$

$$A \rightarrow aAbb \mid aBbb$$

$$B \rightarrow aab$$

$$\left. \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Grxk1oczg

$$n = 2 \quad 4a \ 5b: \text{aaaabbbbbb}$$

$$n = 3 \quad 5a \ 7b: \text{aaaaabbbbbbb}$$

$$n = 4 \quad 6a \ 9b: \text{aaaaaabbbbbbbb}$$

(b) Geben Sie einen Kellerautomaten (PDA) formal an, der die Sprache

$$L_1 = \{ w_1 w_2 w_3 \mid w_1, w_2, w_3 \in \Sigma^* \setminus \{\lambda\} \text{ und } w_1 = w_3^{\text{rev}} \} \in \text{CFL} \text{ über dem Alphabet } \Sigma = \{0, 1\} \text{ akzeptiert.}$$

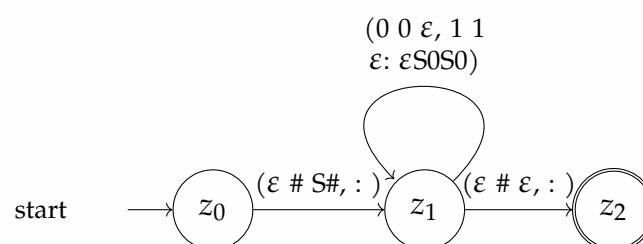
Dabei bezeichnet  $\lambda$  das leere Wort und  $w_3^{\text{rev}}$  bezeichnet das Wort  $w_3$  rückwärts gelesen. Bei Akzeptanz einer Eingabe soll sich der PDA in einem Endzustand befinden und der Keller geleert sein.

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gpkctmk3g

$$P = \left\{ \right.$$

$$S \rightarrow 0S0 \mid 1S1 \mid 0A0 \mid 1A1$$

$$A \rightarrow 0A \mid 1A \mid 0 \mid 1$$

$$\left. \right\}$$


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A5z2zfkdw

Berechenbarkeit  
Algorithmische Komplexität  
(O-Notation)

- (c) Beschreiben Sie in Worten die Arbeitsweise Ihres PDA aus Aufgabenteil (b).

## 66115 / 2020 / Frühjahr / Thema 2 / Aufgabe 4

$A = \{ (M) \mid M \text{ ist Turingmaschine, die bei Eingabe } 101 \text{ hält} \}$ . Dabei bezeichnet  $(M)$  die Gödelnummer der Turingmaschine  $M$ .

- (a) Zeigen Sie, dass  $A$  unentscheidbar ist.

Lösungsvorschlag

Reduktionsbeweis von  $H_0 \leq A$ : TM  $U$

- (i) Die zu  $\square' \in \square^0$  passende TM  $\square^* \in \square^*$  aus  $A$  suchen mit  $\langle \square' \rangle = \langle \square^* \rangle$
- (ii) 101 auf das Band schreiben
- (iii)  $\square^*$  auf 101 starten

Damit könnte  $U$   $H_0$  entscheiden, was aber ein Widerspruch zu  $H_0$  semi-entscheidbar ist. Damit ist  $A$  ebenfalls semi-entscheidbar.

- (b) Zeigen Sie, dass  $A$  semi-entscheidbar ist.

Lösungsvorschlag

siehe a)

- (c) Ist das Komplement  $A^c$  von  $A$  entscheidbar? Ist es semi-entscheidbar? Begründen Sie Ihre Antworten.

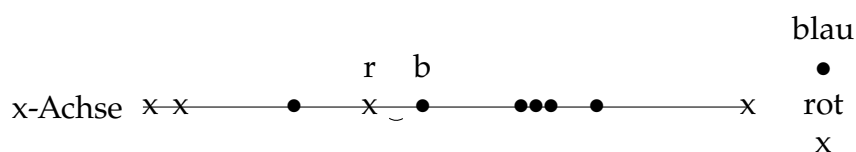
Hinweis: Sie können die Aussagen aus Teilaufgabe a) und b) verwenden, auch wenn Sie sie nicht bewiesen haben.

Lösungsvorschlag

Wenn  $A$  unentscheidbar ist, dann kann entweder  $A$  oder  $A^c$  semi-entscheidbar sein. Wären beide semi-entscheidbar, dann wäre  $A$  aber ebenfalls entscheidbar, was aber nach Voraussetzung ausgeschlossen ist.

## 66115 / 2020 / Frühjahr / Thema 2 / Aufgabe 8

Gegeben seien zwei nichtleere Mengen  $R$  und  $B$  von roten bzw. blauen Punkten auf der  $x$ -Achse. Gesucht ist der minimale euklidische Abstand  $d(r, b)$  über alle Punktpaare  $(r, b)$  mit  $r \in R$  und  $b \in B$ . Hier ist eine Beispielinstantz:





Die Eingabe wird in einem Feld  $A$  übergeben. Jeder Punkt  $A[i]$  mit  $1 \leq i \leq n$  hat eine x-Koordinate  $A[i].x$  und eine Farbe  $A[i].color \in \{\text{rot}, \text{blau}\}$ . Das Feld  $A$  ist nach x-Koordinate sortiert, d.h. gilt  $A[1].x < A[2].x < \dots < A[n].x$ , wobei  $n = |R| + |B|$ .

- (a) Geben Sie in Worten einen Algorithmus an, der den gesuchten Abstand in  $\mathcal{O}(n)$  Zeit berechnet.

Lösungsvorschlag

### Pseudo-Code

#### Algorithmus 2: Minimaler Euklidischer Abstand

```

 $d_{min} := \max ;$ // Setze d_{min} zuerst auf einen maximalen Wert.
for i in $0 \dots \text{vorletzter Index}$ do ; // Iteriere über die Indizes des Punkte-Arrays
 P bis zum vorletzten Index $P[n-1]$

 if $P[n].color \neq P[n+1].color$ then ; // Berechne den Abstand nur, wenn die
 Punkte unterschiedliche Farben haben

 $d = P[n+1].x - P[n].x$
 if $d < d_{min}$ then
 $d_{min} = d$
 end
 end
end

```

### Java

```

public double findMinimalDistance() {
 double distanceMin = Double.MAX_VALUE;
 for (int i = 0; i < latestIndex - 1; i++) {
 if (points[i].color != points[i + 1].color) {
 double distance = points[i + 1].x - points[i].x;
 if (distance < distanceMin) {
 distanceMin = distance;
 }
 }
 }
 return distanceMin;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/fruehjahr/RedBluePairCollection.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/fruehjahr/RedBluePairCollection.java)

- (b) Begründen Sie kurz die Laufzeit Ihres Algorithmus.

Lösungsvorschlag

Da das Array der Länge  $n$  nur einmal durchlaufen wird, ist die Laufzeit  $\mathcal{O}(n)$  sichergestellt.

- (c) Begründen Sie die Korrektheit Ihres Algorithmus.

In  $d_{min}$  steht am Ende der gesuchte Wert (sofern nicht  $d_{min} = Integer.MAX\_VALUE$  geblieben ist)

- (d) Wir betrachten nun den Spezialfall, dass alle blauen Punkte links von allen roten Punkten liegen. Beschreiben Sie in Worten, wie man in dieser Situation den gesuchten Abstand in  $o(n)$  Zeit berechnen kann. (Ihr Algorithmus darf also insbesondere nicht Laufzeit  $\Theta(n)$  haben.)

Zuerst müssen wir den letzten blauen Punkt finden. Das ist mit einer binären Suche möglich. Wir beginnen mit dem ganzen Feld als Suchbereich und betrachten den mittleren Punkt. Wenn er blau ist, wiederholen wir die Suche in der zweiten Hälfte des Suchbereichs, sonst in der ersten, bis wir einen blauen Punkt gefolgt von einem roten Punkt gefunden haben.

Der gesuchte minimale Abstand ist dann der Abstand zwischen dem gefundenen blauen und dem nachfolgenden roten Punkt. Die Binärsuche hat eine Worst-case-Laufzeit von  $\mathcal{O}(\log n)$ .

## 66115 / 2020 / Frühjahr / Thema 2 / Aufgabe 10

Sei  $B$  ein binärer Suchbaum. In jedem Knoten  $v$  von  $B$  wird ein Schlüssel  $v.key \in \mathbb{N}$  gespeichert sowie Zeiger  $v.left$ ,  $v.right$  und  $v.parent$  auf sein linkes Kind, auf sein rechtes Kind und auf seinen Elternknoten. Die Zeiger sind *nil*, wenn der entsprechende Nachbar nicht existiert. Für zwei Knoten  $u$  und  $v$  ist wie üblich der *Abstand* die Anzahl der Kanten auf dem kürzesten Pfad von  $u$  nach  $v$ .

Für einen Knoten  $w$  von  $B$  sei  $B(w)$  der Teilbaum von  $B$  mit Wurzel  $w$ . Für zwei Knoten  $u$  und  $v$  von  $B$  ist  $w$  ein *gemeinsamer Vorfahre*, wenn  $u$  und  $v$  in  $B(w)$  liegen. Wir suchen den niedrigsten gemeinsamen Vorfahren  $ngV(u, v)$  von  $u$  und  $v$ , also einen gemeinsamen Vorfahren  $w$ , so dass für jeden Vorfahren  $w$  von  $u$  und  $v$  gilt, dass  $w$  in  $B(w)$  liegt. Wir betrachten verschiedene Szenarien, in denen Sie jeweils den niedrigsten gemeinsamen Vorfahren von  $u$  und  $v$  berechnen sollen.

### Exkurs: Lowest Common Ancestor

Als Lowest Common Ancestor (LCA) oder „letzter gemeinsamer Vorfahre“ wird in der Informatik und Graphentheorie ein Ermittlungskonzept bezeichnet, das einen gegebenen gewurzelten Baum von Datenstrukturen effizient vorverarbeitet, sodass anschließend Anfragen nach dem letzten gemeinsamen Vorfahren für beliebige Knotenpaare in konstanter Zeit beantwortet werden können.

<sup>a</sup>

<sup>a</sup>[https://de.wikipedia.org/wiki/Lowest\\_Common\\_Ancestor](https://de.wikipedia.org/wiki/Lowest_Common_Ancestor)

- (a) Wir bekommen  $u$  und  $v$  als Zeiger auf die entsprechenden Knoten in  $B$  geliefert. Beschreiben Sie in Worten und in Pseudocode einen Algorithmus, der den niedrigsten gemeinsamen Vorfahren von  $u$  und  $v$  berechnet. Analysieren Sie die Laufzeit Ihres Algorithmus.

```
/**
 * NBV = Niedrigster gemeinsamer Vorfahre.
 *
 * https://afteracademy.com/blog/lowest-common-ancestor-of-a-binary-tree
 */
public class NGV {
 static class Knoten {
 int schlüssel;
 Knoten links;
 Knoten rechts;

 public Knoten(int schlüssel) {
 this.schlüssel = schlüssel;
 }
 }

 /**
 * ngV = niedrigster gemeinsamer Vorfahre
 *
 * @param wurzel Der Wurzelknoten des Binärbaums.
 * @param knoten1 Der erste Knoten, dessen niedrigster gemeinsamer Vorfahre
 * gesucht werden soll.
 * @param knoten2 Der zweite Knoten, dessen niedrigster gemeinsamer Vorfahre
 * gesucht werden soll.
 *
 * @return Der niedrigste gemeinsame Vorfahre der Knoten 1 und 2.
 */
 public static Knoten ngVRekursiv(Knoten wurzel, Knoten knoten1, Knoten
 ↪ knoten2) {
 if (wurzel == null)
 return null;
 if (wurzel.equals(knoten1) || wurzel.equals(knoten2))
 return wurzel;
 Knoten links = ngVRekursiv(wurzel.links, knoten1, knoten2);
 Knoten rechts = ngVRekursiv(wurzel.rechts, knoten1, knoten2);
 if (links == null)
 return rechts;
 else if (rechts == null)
 return links;
 else
 return wurzel;
 }

 /**
 * <pre>
 * {@code
 * 20
 * / \
 * 8 22
 * / \
 * 4 12
 * / \
 * 10 14
 * }
 * </pre>
 */
}
```

```

*
* Beispiele von
* https://www.geeksforgeeks.org/lowest-common-ancestor-in-a-binary-search-
↪ tree/
*
* @param args Kommandozeilen-Argumente
*/
public static void main(String[] args) {
 Knoten wurzel = new Knoten(20);

 Knoten knoten8 = new Knoten(8);
 Knoten knoten22 = new Knoten(22);
 Knoten knoten4 = new Knoten(4);
 Knoten knoten12 = new Knoten(12);
 Knoten knoten10 = new Knoten(10);
 Knoten knoten14 = new Knoten(14);

 wurzel.links = knoten8;
 wurzel.rechts = knoten22;
 wurzel.links.links = knoten4;
 wurzel.links.rechts = knoten12;
 wurzel.links.rechts.links = knoten10;
 wurzel.links.rechts.rechts = knoten14;

 System.out.println(ngVRekursiv(wurzel, knoten10, knoten14).schlüssel); // 12
 System.out.println(ngVRekursiv(wurzel, knoten14, knoten8).schlüssel); // 8
 System.out.println(ngVRekursiv(wurzel, knoten10, knoten22).schlüssel); // 20
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/fruehjahr/NGV.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/fruehjahr/NGV.java)

- (b) Wir bekommen  $u$  und  $v$  wieder als Zeiger auf die entsprechenden Knoten in  $B$  geliefert. Seien  $d_u$  und  $d_v$ , die Abstände von  $u$  bzw.  $v$  zum niedrigsten gemeinsamen Vorfahren von  $u$  und  $v$ . Die Laufzeit Ihres Algorithmus soll  $O(\max\{d_u, d_v\})$  sein. Dabei kann Ihr Algorithmus in jedem Knoten  $v$  eine Information  $v.info$  speichern. Skizzieren Sie Ihren Algorithmus in Worten.
- (c) Wir bekommen die Schlüssel  $u.key$  und  $v.key$ . Die Laufzeit Ihres Algorithmus soll proportional zum Abstand der Wurzel von  $B$  zum niedrigsten gemeinsamen Vorfahren von  $u$  und  $v$  sein. Skizzieren Sie Ihren Algorithmus in Worten.

## 66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 1

Antworten Sie mit „Stimmt“ oder „Stimmt nicht“. Begründen Sie Ihr Urteil kurz.

- (a) Eine Sprache ist genau dann regulär, wenn sie unendlich viele Wörter enthält.

Lösungsvorschlag

Stimmt nicht. Sprachen mit endlicher Mächtigkeit sind immer regulär. Endliche Sprachen sind in obenstehender Aussage ausgeschlossen.

- (b) Zu jedem nichtdeterministischen endlichen Automaten mit  $n$  Zuständen gibt es einen deterministischen endlichen Automaten, der die gleiche Sprache erkennt und höchstens  $n^2$  Zustände hat.

Lösungsvorschlag

Stimmt nicht. Müsste  $2^n$  heißen.

- (c) Das Komplement einer kontextfreien Sprache ist wieder kontextfrei.

Lösungsvorschlag

Stimmt nicht. Kontextfreie Sprachen sind nicht abgeschlossen unter dem Komplement. Das Komplement einer kontextfreien Sprache kann regulär, kontextfrei oder kontextsensitiv sein.

- (d) Wenn ein Problem unentscheidbar ist, dann ist es nicht semientscheidbar.

Lösungsvorschlag

Stimmt nicht. Semientscheidbarkeit ist eine typische Form der Unentscheidbarkeit. Unentscheidbarkeit ist das Gegenteil von Entscheidbarkeit. Unentscheidbar kann entweder völlig unentscheidbar sein oder semientscheidbar.

- (e) Sei  $f$  eine totale Funktion. Dann gibt es ein WHILE-Programm, das diese berechnet.

Lösungsvorschlag

Stimmt nicht. Wir wissen nicht, ob die totale Funktion  $f$  berechenbar ist. Wenn  $f$  berechenbar ist, dann wäre die Aussage richtig.

- (f) Das Halteproblem für LOOP-Programme ist entscheidbar.

Lösungsvorschlag

Stimmt. Alle LOOP-Programme terminieren (halten). Es gibt für jede Eingabe eine Ausgabe.

- (g) Die Komplexitätsklasse  $\mathcal{NP}$  enthält genau die Entscheidungsprobleme, die in nicht-polynomieller Zeit entscheidbar sind.

Lösungsvorschlag

Stimmt. Die Aussage entspricht der Definition der Komplexitätsklasse  $\mathcal{NP}$ .

- (h) Falls  $P \geq NP$ , dann gibt es keine  $\mathcal{NP}$ -vollständigen Probleme, die in  $P$  liegen.

Lösungsvorschlag

Stimmt. Entspricht der Definition.

## 66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2

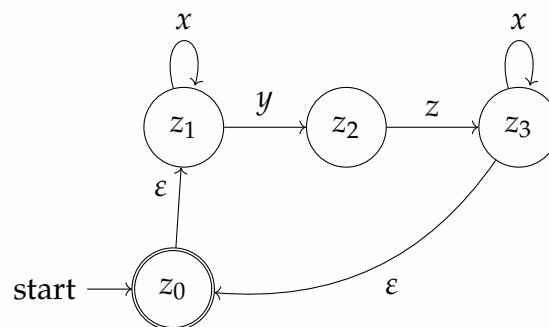
Sei  $\Sigma = \{x, y, z\}$ . Sei  $L = (x^* y z x^*)^* \subseteq \Sigma^*$ .

- (a) Geben Sie einen endlichen (deterministischen oder nichtdeterministischen) Automaten  $A$  an, der  $L$  erkennt bzw. akzeptiert.

Lösungsvorschlag

**Nichtdeterministischer endlicher Automat**

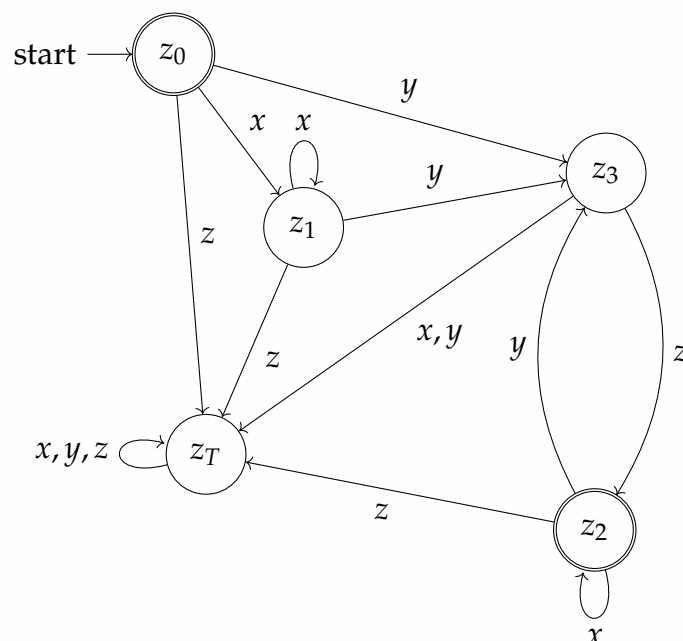
$$A_{\text{NEA}} = (\{z_0, z_1, z_2, z_3, z_T\}, \{x, y, z\}, \delta, \{z_0, z_2\}, z_0)$$



Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ajpmxqvh9](http://flaci.com/Ajpmxqvh9)

**Deterministischer endlicher Automat**

$$A_{\text{DEA}} = (\{z_0, z_1, z_2, z_3, z_T\}, \{x, y, z\}, \delta, \{z_0, z_2\}, z_0)$$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A5xo470g9

Kontextfreie Sprache  
Pumping-Lemma (Reguläre Sprache)

(b) Geben Sie eine reguläre und eindeutige Grammatik  $G$  an, die  $L$  erzeugt.

Lösungsvorschlag

$$P = \left\{ \begin{array}{l} Z_0 \rightarrow xZ_1 \mid yZ_3 \mid \varepsilon \\ Z_1 \rightarrow yZ_3 \mid xZ_1 \\ Z_2 \rightarrow xZ_2 \mid x \mid yZ_3 \\ Z_3 \rightarrow zZ_2 \mid z \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gjfc3c2d2

## 66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Seien  $\Sigma = \{a, b, c\}$  und  $L = \{wc\hat{w} \mid w \in \{a, b\}^*\}$ . Dabei ist  $\hat{w}$  das zu  $w$  gespiegelte Wort.

(a) Zeigen Sie, dass  $L$  nicht regulär ist.

### Exkurs: Pumping-Lemma für Reguläre Sprachen

Es sei  $L$  eine reguläre Sprache. Dann gibt es eine Zahl  $j$ , sodass für alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  (jedes Wort  $\omega$  in  $L$  mit Mindestlänge  $j$ ) jeweils eine Zerlegung  $\omega = uvw$  existiert, sodass die folgenden Eigenschaften erfüllt sind:

- (i)  $|v| \geq 1$  (Das Wort  $v$  ist nicht leer.)
- (ii)  $|uv| \leq j$  (Die beiden Wörter  $u$  und  $v$  haben zusammen höchstens die Länge  $j$ .)
- (iii) Für alle  $i = 0, 1, 2, \dots$  gilt  $uv^i w \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^i w$  in der Sprache  $L$ )

Die kleinste Zahl  $j$ , die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache  $L$  genannt.

Lösungsvorschlag

$L$  ist regulär. Dann gilt für  $L$  das Pumping-Lemma. Sei  $j$  die Zahl aus dem Pumping-Lemma. Dann muss sich das Wort  $a^j b c b a^j \in L$  aufpumpen lassen (da  $|a^j b c b a^j| \geq j$ ).  $a^j b c b a^j = uvw$  ist eine passende Zerlegung laut Lemma. Da  $|uv| < j$ , ist  $u = a^x$ ,  $v = a^y$ ,  $w = a^z b c b a^j$ , wobei  $y > 0$  und  $x + y + z = j$ . Aber dann  $uv^0 w = a^{x+z} b c b a^j \notin L$ , da  $x + z < j$ . Widerspruch. <sup>a</sup>

<sup>a</sup><https://userpages.uni-koblenz.de/~sofronie/gti-ss-2015/slides/>

- (b) Zeigen Sie, dass  $L$  kontextfrei ist, indem Sie eine geeignete Grammatik angeben und anschließend begründen, dass diese die Sprache  $L$  erzeugt.

Lösungsvorschlag

$$P = \left\{ \begin{array}{l} S \rightarrow aSa \mid aCa \mid bSb \mid bCb \\ C \rightarrow c \end{array} \right\}$$

$$S \vdash aSa \vdash abCba \vdash abcba$$

$$S \vdash bSb \vdash bbSbb \vdash bbaSabb \vdash bbacabb$$

### 66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4

Geben Sie für jede der folgenden Mengen an, ob sie entscheidbar ist oder nicht. Dabei ist  $\sigma_w$ , die Funktion, die von der Turingmaschine berechnet wird, die durch das Wort  $w$  kodiert wird. Beweisen Sie Ihre Behauptungen.

- (a)  $L_1 = \{ w \in \Sigma^* \mid \sigma_w(0) = 0 \}$

Lösungsvorschlag

Nicht entscheidbar wegen dem Halteproblem.

- (b)  $L_2 = \{ w \in \Sigma^* \mid \sigma_w(w) = w \}$

Lösungsvorschlag

Nicht entscheidbar wegen dem Halteproblem.

- (c)  $L_3 = \{ w \in \Sigma^* \mid \sigma_0(0) = w \}$

Lösungsvorschlag

Entscheidbar wegen  $\sigma_0$ .

### 66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 5

Sei  $\mathcal{IF}$  die Menge aller aussagenlogischen Formeln, die ausschließlich mit den Konstanten 0 und 1, logischen Variablen  $x_i$  mit  $i \in \mathbb{N}$  und der Implikation  $\Rightarrow$  als Operationszeichen aufgebaut sind, wobei natürlich Klammern zugelassen sind. Beachten Sie, dass  $x_i \Rightarrow x_j$  die gleiche Wahrheitstabelle wie  $\neg x_i \vee x_j$  hat.



Wir betrachten das Problem  $\text{ISAT}$ . Eine Formel  $F \in \mathcal{IF}$  ist genau dann in  $\text{ISAT}$  enthalten, wenn sie erfüllbar ist, das heißt, falls es eine Belegung der Variablen mit Konstanten 0 oder 1 gibt, sodass  $F'$  den Wert 1 annimmt.

Zeigen Sie:  $\text{ISAT}$  ist NP-vollständig. Sie dürfen benutzen, dass das  $\text{SAT}$ -Problem NP-vollständig ist.

Lösungsvorschlag

## 66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 1

Betrachten Sie die folgende Prozedur `countup`, die aus zwei ganzzahligen Eingabewerten  $n$  und  $m$  einen ganzzahligen Ausgabewert berechnet:

```
procedure countup(n, m : integer): integer
var x, y : integer;
begin
 x := n;
 y := 0;
 while (y < m) do
 x := x - 1;
 y := y + 1;
 end while
 return x;
end
```

- (a) Führen Sie `countup(3,2)` aus. Geben Sie für jeden Schleifendurchlauf jeweils den Wert der Variablen  $n$ ,  $m$ ,  $x$  und  $y$  zu Beginn der `while`-Schleife und den Rückgabewert der Prozedur an.

Lösungsvorschlag

| n               | m | x | y | ausgeführter Code, der Änderung bewirkte |
|-----------------|---|---|---|------------------------------------------|
| 3               | 2 | 3 | 0 |                                          |
| 3               | 2 | 2 | 1 | x := x - 1; y := y + 1;                  |
| Rückgabewert: 1 |   |   |   |                                          |

### Java-Implementation der Prozedur

```
public class CountUp {

 public static int countup(int n, int m) {
 int x = n;
 int y = 0;
 while (y < m) {
 System.out.println(String.format("%s %s %s %s", n, m, x, y));
 x = x - 1;
 y = y + 1;
 }
 return x;
 }
}
```

```
public static void main(String[] args) {
 System.out.println(countup(3, 2));
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/counter/CountUp.java](https://src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/counter/CountUp.java)

- (b) Gibt es Eingabewerte von  $n$  und  $m$ , für die die Prozedur `countup` nicht terminiert? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Nein. Mit jedem Schleifendurchlauf wird der Wert der Variablen  $y$  um eins hochgezählt. Die Werte, die  $y$  annimmt, sind streng monoton steigend.  $y$  nähert sich  $m$  an, bis  $y$  nicht mehr kleiner ist als  $m$  und die Prozedur terminiert. An diesem Sachverhalt ändern auch sehr große Zahlen, die über die Variable  $m$  der Prozedur übergeben werden, nichts.

- (c) Geben Sie die asymptotische worst-case Laufzeit der Prozedur `countup` in der  $\Theta$ -Notation in Abhängigkeit von den Eingabewerten  $n$  und/oder  $m$  an. Begründen Sie Ihre Antwort.

Lösungsvorschlag

Die Laufzeit der Prozedur ist immer  $\Theta(m)$ . Die Laufzeit hängt nur von  $m$  ab. Es kann nicht zwischen best-, average and worst-case unterschieden werden.

- (d) Betrachten Sie nun die folgende Prozedur `countdown`, die aus zwei ganzzahligen Eingabewerten  $n$  und  $m$  einen ganzzahligen Ausgabewert berechnet:

```
procedure countdown(n, m : integer) : integer
var x, y : integer;
begin
 x := n;
 y := 0;
 while (n > 0) do
 if (y < m) then
 x := x - 1;
 y := y + 1;
 else
 y := 0;
 n := n / 2; /* Ganzzahldivision */
 end if
 end while
 return x;
end
```

Führen Sie `countdown(3, 2)` aus. Geben Sie für jeden Schleifendurchlauf jeweils den Wert der Variablen  $n$ ,  $m$ ,  $x$  und  $y$  zu Beginn der `while`-Schleife und den Rückgabewert der Prozedur an.

| n | m | x  | y | ausgeführter Code, der Änderung bewirkte |
|---|---|----|---|------------------------------------------|
| 3 | 2 | 3  | 0 |                                          |
| 3 | 2 | 2  | 1 | <code>x := x - 1; y := y + 1;</code>     |
| 3 | 2 | 1  | 2 | <code>x := x - 1; y := y + 1;</code>     |
| 1 | 2 | 1  | 0 | <code>y := 0; n := n / 2;</code>         |
| 1 | 2 | 0  | 1 | <code>x := x - 1; y := y + 1;</code>     |
| 1 | 2 | -1 | 2 | <code>x := x - 1; y := y + 1;</code>     |

Rückgabewert: -1

### Java-Implementation der Prozedur

```
public class Countdown {

 public static int countdown(int n, int m) {
 int x = n;
 int y = 0;
 while (n > 0) {
 System.out.println(String.format("%s %s %s %s", n, m, x, y));
 if (y < m) {
 x = x - 1;
 y = y + 1;
 } else {
 y = 0;
 n = n / 2; /* Ganzzahldivision */
 }
 }
 return x;
 }

 public static void main(String[] args) {
 System.out.println(countdown(3, 2));
 }

}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/counter/CountDown.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/counter/CountDown.java)

(e) Gibt es Eingabewerte von  $n$  und  $m$ , für die die Prozedur `countdown` nicht terminiert?

Begründen Sie Ihre Antwort.

Nein.

$n \leq 0$  terminiert sofort

$m \leq 0$  Der Falsch-Block der Wenn-Dann-Bedingung erniedrigt  $n$   $n := n / 2$ ;  
bis  $0$  erreicht ist. Dann terminiert die Prozedur.

$m > 0$  Der erste Wahr-Block der Wenn-Dann-Bedingung erhöht  $y$  streng monoton bis  $y \geq m$ . 2. Falsch-Block der Wenn-Dann-Bedingung halbiert  $n$  bis 0. 1. und 2. solange bis  $n = 0$

- (f) Geben Sie die asymptotische Laufzeit der Prozedur `countdown` in der  $\Theta$ -Notation in Abhängigkeit von den Eingabewerten  $n$  und/oder  $m$  an unter der Annahme, dass  $m \geq 0$  und  $n > 0$ . Begründen Sie Ihre Antwort.

Lösungsvorschlag

Anzahl der Wiederholungen der while-Schleife:  $m + 1$ :

- $m$  oft: bis  $y < m$
- +1 Halbierung von  $n$  und  $y$  auf 0 setzen

wegen dem  $n/2$  ist die Laufzeit logarithmisch, ähnlich wie der worst case bei der Binären Suche.

| $n$ | $m$ | $x$ | $y$ | ausgeführter Code, der Änderung bewirkte |
|-----|-----|-----|-----|------------------------------------------|
| 16  | 3   | 16  | 0   |                                          |
| 16  | 3   | 15  | 1   |                                          |
| 16  | 3   | 14  | 2   |                                          |
| 16  | 3   | 13  | 3   |                                          |
| 8   | 3   | 13  | 0   | $y := 0; n := n / 2;$                    |
| 8   | 3   | 12  | 1   |                                          |
| 8   | 3   | 11  | 2   |                                          |
| 8   | 3   | 10  | 3   |                                          |
| 4   | 3   | 10  | 0   | $y := 0; n := n / 2;$                    |
| 4   | 3   | 9   | 1   |                                          |
| 4   | 3   | 8   | 2   |                                          |
| 4   | 3   | 7   | 3   |                                          |
| 2   | 3   | 7   | 0   | $y := 0; n := n / 2;$                    |
| 2   | 3   | 6   | 1   |                                          |
| 2   | 3   | 5   | 2   |                                          |
| 2   | 3   | 4   | 3   |                                          |
| 1   | 3   | 4   | 0   | $y := 0; n := n / 2;$                    |
| 1   | 3   | 3   | 1   |                                          |
| 1   | 3   | 2   | 2   |                                          |
| 1   | 3   | 1   | 3   |                                          |

$$\Theta((m+1) \log_2 n)$$

Wegkürzen der Konstanten:

$$\Rightarrow \Theta(m \log n)$$

Binärbaum

## 66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 2

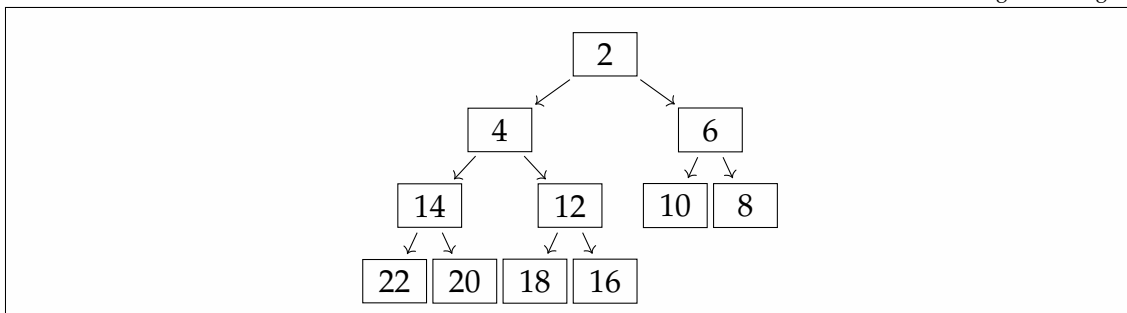
Wir betrachten ein Feld  $A$  von ganzen Zahlen mit  $n$  Elementen, die über die Indizes  $A[0]$  bis  $A[n-1]$  angesprochen werden können. In dieses Feld ist ein binärer Baum nach den folgenden Regeln eingebettet: Für das Feldelement mit Index  $i$  befindet sich

- der Elternknoten im Feldelement mit Index  $\lfloor \frac{i-1}{2} \rfloor$ ,
- der linke Kindknoten im Feldelement mit Index  $2 \cdot i + 1$ , und
- der rechte Kindknoten im Feldelement mit Index  $2 \cdot i + 2$ .

(a) Zeichnen Sie den durch das folgende Feld repräsentierten binären Baum.

| i    | 0 | 1 | 2 | 3  | 4  | 5  | 6 | 7  | 8  | 9  | 10 |
|------|---|---|---|----|----|----|---|----|----|----|----|
| A[i] | 2 | 4 | 6 | 14 | 12 | 10 | 8 | 22 | 20 | 18 | 16 |

Lösungsvorschlag



(b) Der folgende rekursive Algorithmus sei gegeben:

### Pseudo-Code / Pascal

```

procedure magic(i, n : integer) : boolean
begin
 if (i > (n - 2) / 2) then
 return true;
 endif
 if (A[i] <= A[2 * i + 1] and A[i] <= A[2 * i + 2] and
 magic(2 * i + 1, n) and magic(2 * i + 2, n)) then
 return true;
 endif
 return false;
end

```

## Java-Implementation

```
public static boolean magic(int i, int n) {
 System.out.println(String.format("i: %s n: %s", i, n));
 if (i > (n - 2) / 2) {
 return true;
 }
 if (A[i] <= A[2 * i + 1] && A[i] <= A[2 * i + 2] && magic(2 * i + 1, n) &&
 ↪ magic(2 * i + 2, n)) {
 return true;
 }
 return false;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/Baum.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java)

Gegeben sei folgendes Feld:

|      |   |   |   |    |
|------|---|---|---|----|
| i    | 0 | 1 | 2 | 3  |
| A[i] | 2 | 4 | 6 | 14 |

Führen Sie `magic(0,3)` auf dem Feld aus. Welches Resultat liefert der Algorithmus zurück?

Lösungsvorschlag

true

- (c) Wie nennt man die Eigenschaft, die der Algorithmus `magic` auf dem Feld A prüft? Wie lässt sich diese Eigenschaft formal beschreiben?

Lösungsvorschlag

Die sogenannte „Haldeneigenschaft“ bzw. „Heap-Eigenschaft“ einer Min-Halde. Der Schlüssel eines jeden Knotens ist kleiner (oder gleich) als die Schlüssel seiner Kinder.

Ein Baum erfüllt die Heap-Eigenschaft bezüglich einer Vergleichsrelation „ $>$ “ auf den Schlüsselwerten genau dann, wenn für jeden Knoten  $u$  des Baums gilt, dass  $u_{\text{wert}} > v_{\text{wert}}$  für alle Knoten  $v$  aus den Unterbäumen von  $u$ .

- (d) Welche Ausgaben sind durch den Algorithmus `magic` möglich, wenn das Eingabefeld aufsteigend sortiert ist? Begründen Sie Ihre Antwort.

Lösungsvorschlag

**true**. Eine sortierte aufsteigende Zahlenfolge entspricht den Haldeneigenschaften einer Min-Heap.

- (e) Geben Sie zwei dreielementige Zahlenfolgen (bzw. Felder) an, eine für die `magic(0,2)` den Wert **true** liefert und eine, für die `magic(0,2)` den Wert **false** liefert.

```
A = new int[] { 1, 2, 3 };
System.out.println(magic(0, 2)); // true

A = new int[] { 2, 1, 3 };
System.out.println(magic(0, 2)); // false
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/Baum.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java)

- (f) Betrachten Sie folgende Variante `almostmagic` der oben bereits erwähnten Prozedur `magic`, bei der die Anweisungen in Zeilen 3 bis 5 entfernt wurden:

### Pseudo-Code / Pascal

```
procedure almostmagic(i, n : integer) : boolean
begin
 // leer
 // leer
 // leer
 if (A[i] <= A[2 * i + 1] and A[i] <= A[2 * i + 2] and
 magic(2 * i + 1, n) and magic(2 * i + 2, n)) then
 return true;
 endif
 return false;
end
```

### Java-Implementation

```
public static boolean almostmagic(int i, int n) {
 System.out.println(String.format("i: %s n: %s", i, n));
 if (A[i] <= A[2 * i + 1] && A[i] <= A[2 * i + 2] && magic(2 * i + 1, n) &&
 magic(2 * i + 2, n)) {
 return true;
 }
 return false;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/Baum.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java)

Beschreiben Sie die Umstände, die auftreten können, wenn `almostmagic` auf einem Feld der Größe `n` aufgerufen wird. Welchen Zweck erfüllt die entfernte bedingte Anweisung?

Wird die Prozedur zum Beispiel mit `almostmagic(0, n + 1)` aufgerufen, kommst es zu einem sogenannten „Array-Index-Out-of-Bounds“ Fehler, d. h. die Prozedur will auf Index des Feldes zugreifen, der im Feld gar nicht existiert. Die drei zusätzlichen Zeilen in der Methode `magic` bieten dafür einen Schutz, indem sie vor den Index-Zugriffen auf das Feld `true` zurückgeben.

```
// A = new int[] { 1, 2, 3 };
```



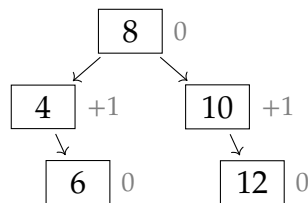
```
// System.out.println(almostmagic(0, 4)); // Exception in thread "main"
↪ java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for
↪ length 3
```

AVL-Baum

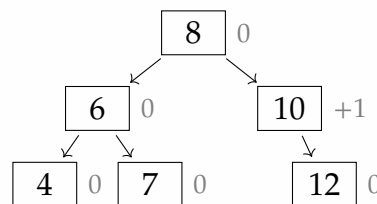
Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/Baum.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java)

- (g) Fügen Sie jeweils den angegebenen Wert in den jeweils angegebenen AVL-Baum mit aufsteigender Sortierung ein und zeichnen Sie den resultierenden Baum vor möglicherweise erforderlichen Rotationen. Führen Sie danach bei Bedarf die erforderliche(n) Rotation(en) aus und zeichnen Sie dann den resultierenden Baum. Sollten keine Rotationen erforderlich sein, so geben Sie dies durch einen Text wie „keine Rotationen nötig“ an.

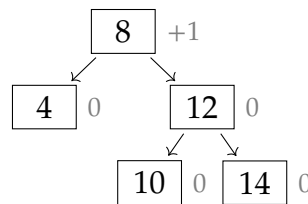
- (i) Wert 7 einfügen



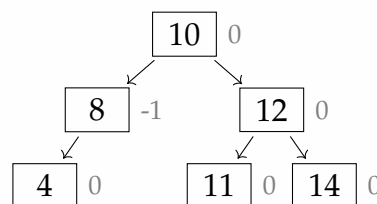
Lösungsvorschlag



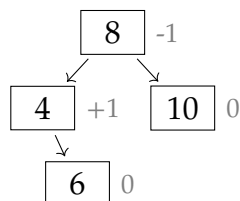
- (ii) Wert 11 einfügen



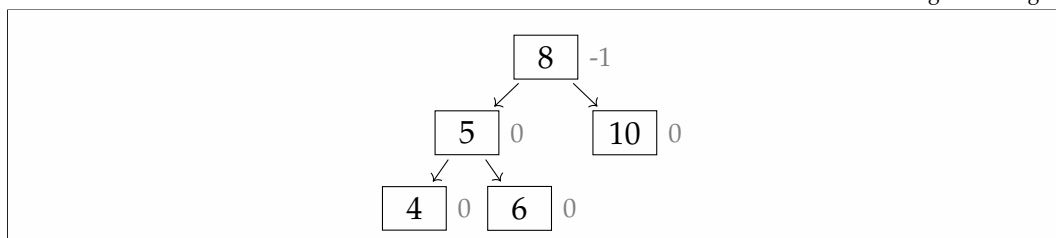
Lösungsvorschlag



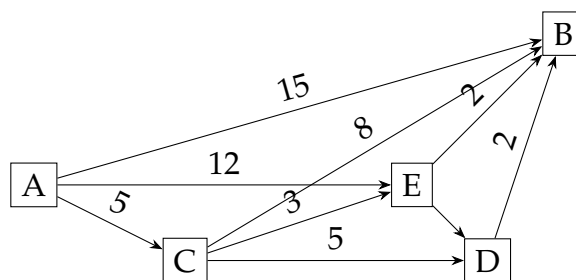
- (iii) Wert 5 einfügen



Lösungsvorschlag



## 66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 3



- (a) Ermitteln Sie mit dem Algorithmus von Dijkstra den kürzesten Weg vom Knoten A zu allen erreichbaren Knoten in G. Verwenden Sie zur Lösung eine Tabelle der folgenden Form. Markieren Sie in jeder Zeile den jeweils als nächstes zu betrachtenden Knoten und führen Sie die Prioritätswarteschlange der noch zu betrachtenden Knoten (aufsteigend sortiert).

| Nr. | besucht | A | B        | C        | D        | E        |
|-----|---------|---|----------|----------|----------|----------|
| 0   |         | 0 | $\infty$ | $\infty$ | $\infty$ | $\infty$ |

Lösungsvorschlag

| Nr. | besucht | A        | B         | C        | D        | E        |
|-----|---------|----------|-----------|----------|----------|----------|
| 0   |         | 0        | $\infty$  | $\infty$ | $\infty$ | $\infty$ |
| 1   | A       | <b>0</b> | 15        | 5        | $\infty$ | 12       |
| 2   | C       |          | 13        | <b>5</b> | 10       | 8        |
| 3   | E       |          | 10        |          | 9        | <b>8</b> |
| 4   | D       |          | 10        |          | <b>9</b> |          |
| 5   | B       |          | <b>10</b> |          |          |          |

- (b) Geben Sie den kürzesten Pfad vom Knoten A zum Knoten B an.

$A \rightarrow C \rightarrow E \rightarrow B: 10$ 

| nach              | Entfernung | Reihenfolge | Pfad                                          |
|-------------------|------------|-------------|-----------------------------------------------|
| $A \rightarrow A$ | 0          | 0           |                                               |
| $A \rightarrow B$ | 10         | 5           | $A \rightarrow C \rightarrow E \rightarrow B$ |
| $A \rightarrow C$ | 5          | 2           | $A \rightarrow C$                             |
| $A \rightarrow D$ | 9          | 4           | $A \rightarrow C \rightarrow E \rightarrow D$ |
| $A \rightarrow E$ | 8          | 3           | $A \rightarrow C \rightarrow E$               |

**66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 4**

(a) Betrachten Sie das folgende Code-Beispiel (in Java-Notation):

```

int mystery(int n) {
 int a = 0, b = 0;
 int i = 0;
 while (i < n) {
 a = b + i;
 b = a;
 i = i + 1;
 }
 return a;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/o\\_notation/Mystery1.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/o_notation/Mystery1.java)

Bestimmen Sie die asymptotische worst-case Laufzeit des Code-Beispiels in  $\mathcal{O}$ -Notation bezüglich der Problemgröße  $n$ . Begründen Sie Ihre Antwort.

Lösungsvorschlag

Die asymptotische worst-case Laufzeit des Code-Beispiels in  $\mathcal{O}$ -Notation ist  $\mathcal{O}(n)$ .

Die **while**-Schleife wird genau  $n$  mal ausgeführt. In der Schleife wird die Variable **i** in der Zeile **i = i + 1**; inkrementiert. **i** wird mit 0 initialisiert. Die **while**-Schleife endet, wenn **i** gleich groß ist als **n**.

(b) Betrachten Sie das folgende Code-Beispiel (in Java-Notation):

```

int mystery(int n) {
 int r = 0;
 while (n > 0) {
 int y = n;
 int x = n;
 for (int i = 0; i < y; i++) {
 for (int j = 0; j < i; j++) {
 r = r + 1;
 }
 }
 }
}

```

```

 }
 r = r - 1;
 }
 n = n - 1;
}
return r;

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/o\\_notation/Mystery2.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/o_notation/Mystery2.java)

Bestimmen Sie für das Code-Beispiel die asymptotische worst-case Laufzeit in  $\mathcal{O}$ -Notation bezüglich der Problemgröße  $n$ . Begründen Sie Ihre Antwort.

Lösungsvorschlag

```

while: n -mal
1. for: $n, n-1, \dots, 2, 1$
2. for: $1, 2, \dots, n-1, n$
 $n \times n \times n = \mathcal{O}(n^3)$

```

- (c) Bestimmen Sie eine asymptotische Lösung (in  $\Theta$ -Schreibweise) für die folgende Rekursionsgleichung:

$$T(n) = T\left(\frac{n}{2}\right) + \frac{1}{2}n^2 + n$$

### Exkurs: Master-Theorem

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

$a$  = Anzahl der rekursiven Aufrufe, Anzahl der Unterprobleme in der Rekursion ( $a \geq 1$ ).

$\frac{1}{b}$  = Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird, Anteil an der Verkleinerung des Problems ( $b > 1$ ).

$f(n)$  = Kosten (Aufwand, Nebenkosten), die durch die Division des Problems und die Kombination der Teillösungen entstehen. Eine von  $T(n)$  unabhängige und nicht negative Funktion.

Dann gilt:

1. Fall:  $T(n) \in \Theta\left(n^{\log_b a}\right)$

falls  $f(n) \in \mathcal{O}\left(n^{\log_b a - \epsilon}\right)$  für  $\epsilon > 0$

2. Fall:  $T(n) \in \Theta\left(n^{\log_b a} \cdot \log n\right)$

falls  $f(n) \in \Theta\left(n^{\log_b a}\right)$

**3. Fall:**  $T(n) \in \Theta(f(n))$

falls  $f(n) \in \Omega(n^{\log_b a + \varepsilon})$  für  $\varepsilon > 0$  und ebenfalls für ein  $c$  mit  $0 < c < 1$  und alle hinreichend großen  $n$  gilt:  $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$

Lösungsvorschlag

**Allgemeine Rekursionsgleichung:**

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

**Anzahl der rekursiven Aufrufe ( $a$ ):**

$$1$$

**Anteil Verkleinerung des Problems ( $b$ ):**

$$\text{um } \frac{1}{2} \text{ also } b = 2$$

**Laufzeit der rekursiven Funktion ( $f(n)$ ):**

$$\frac{1}{2}n^2 + n$$

**Ergibt folgende Rekursionsgleichung:**

$$T(n) = 1 \cdot T\left(\frac{n}{2}\right) + \frac{1}{2}n^2 + n$$

Nebenrechnung:  $\log_b a = \log_2 1 = 0$

**1. Fall:**  $f(n) \in \mathcal{O}(n^{\log_b a - \varepsilon})$ :

$$\frac{1}{2}n^2 + n \notin \mathcal{O}(n^{-1})$$

**2. Fall:**  $f(n) \in \Theta(n^{\log_b a})$ :

$$\frac{1}{2}n^2 + n \notin \Theta(1)$$

**3. Fall:**  $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ :

$$\varepsilon = 2$$

$$\frac{1}{2}n^2 + n \in \Omega(n^2)$$

Für eine Abschätzung suchen wir eine Konstante, damit gilt:

$$1 \cdot f\left(\frac{n}{2}\right) \leq c \cdot f(n)$$

$$\frac{1}{2} \cdot \frac{1}{4}n^2 + \frac{1}{2}n \leq c \cdot \left(\frac{1}{2} \cdot n^2 + n\right)$$

$$\text{Damit folgt } c = \frac{1}{4}$$

$$\text{und } 0 < c < 1$$

$$\Rightarrow \Theta\left(\frac{1}{2}n^2 + n\right)$$

$$\Rightarrow \Theta(n^2)$$

Berechne die Rekursionsgleichung auf WolframAlpha: WolframAlpha

## 66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 5

Gegeben seien die folgenden Schlüssel  $k$  zusammen mit ihren Streuwerten  $h(k)$ :

|        |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|
| $k$    | B | Y | E | ! | A | U | D | ? |
| $h(k)$ | 5 | 4 | 0 | 4 | 4 | 0 | 7 | 2 |

- (a) Fügen Sie die Schlüssel in der angegebenen Reihenfolge (von links nach rechts) in eine Streutabelle der Größe 8 ein und lösen Sie Kollisionen durch verkettete Listen auf.

Stellen Sie die Streutabelle in folgender Art und Weise dar:

Lösungsvorschlag

| Fach | Schlüssel $k$ (verkettete Liste, zuletzt eingetragener Schlüssel rechts) |
|------|--------------------------------------------------------------------------|
| 0    | E, U,                                                                    |
| 1    |                                                                          |
| 2    | ?                                                                        |
| 3    |                                                                          |
| 4    | Y, !, A                                                                  |
| 5    | B,                                                                       |
| 6    |                                                                          |
| 7    | D                                                                        |

- (b) Fügen Sie die gleichen Schlüssel in der gleichen Reihenfolge und mit der gleichen Streufunktion in eine neue Streutabelle der Größe 8 ein. Lösen Sie Kollisionen diesmal aber durch lineares Sondieren mit Schrittweite +1 auf.

Geben Sie für jeden Schlüssel jeweils an, welche Fächer Sie in welcher Reihenfolge sondiert haben und wo der Schlüssel schlussendlich gespeichert wird.

Lösungsvorschlag

| Fach | Schlüssel $k$ |
|------|---------------|
| 0    | E             |
| 1    | U             |
| 2    | D             |
| 3    | ?             |
| 4    | Y             |
| 5    | B             |
| 6    | !             |
| 7    | A             |

| Schlüssel | Sondierung | Speicherung |
|-----------|------------|-------------|
| B         |            | 5           |
| Y         |            | 4           |
| E         |            | 0           |
| !         | 4, 5       | 6           |
| A         | 4, 5, 6    | 7           |
| U         | 0          | 1           |
| D         | 7, 0, 1    | 2           |
| ?         | 2          | 3           |

- (c) Bei der doppelten Streuadressierung verwendet man eine Funktionsschar  $h_i$ , die sich aus einer primären Streufunktion  $h_0$  und einer Folge von sekundären Streufunktionen  $h_1, h_2, \dots$  zusammensetzt. Die folgenden Werte der Streufunktionen sind gegeben:

| $k$      | B | Y | E | ! | A | U | D | ? |
|----------|---|---|---|---|---|---|---|---|
| $h_0(k)$ | 5 | 4 | 0 | 4 | 4 | 0 | 7 | 2 |
| $h_1(k)$ | 6 | 3 | 3 | 3 | 1 | 2 | 6 | 0 |
| $h_2(k)$ | 7 | 2 | 6 | 2 | 6 | 4 | 5 | 6 |
| $h_3(k)$ | 0 | 1 | 1 | 1 | 3 | 6 | 4 | 4 |

Fügen Sie die Schlüssel in der angegebenen Reihenfolge (von links nach rechts) in eine Streutabelle der Größe 8 ein und geben Sie für jeden Schlüssel jeweils an, welche Streufunktion  $h_i$  zur letztendlichen Einsortierung verwendet wurde.

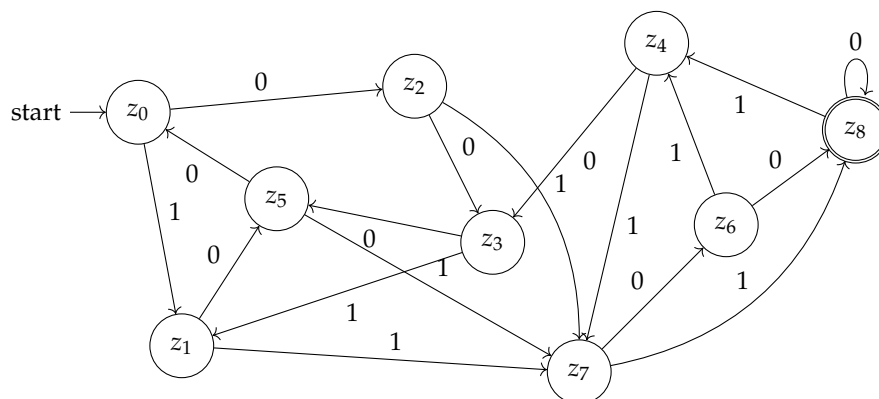
Lösungsvorschlag

| Fach | Schlüssel $k$ |
|------|---------------|
| 0    | E             |
| 1    | A             |
| 2    | U             |
| 3    | !             |
| 4    | Y             |
| 5    | B             |
| 6    | ?             |
| 7    | D             |

| Schlüssel | Streufunction |
|-----------|---------------|
| B         | $h_0(k)$      |
| Y         | $h_0(k)$      |
| E         | $h_0(k)$      |
| !         | $h_1(k)$      |
| A         | $h_1(k)$      |
| U         | $h_1(k)$      |
| D         | $h_0(k)$      |
| ?         | $h_2(k)$      |

## 66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1

- (a) Geben Sie einen deterministischen endlichen Automaten (DEA) mit minimaler Anzahl an Zuständen an, der dieselbe Sprache akzeptiert wie folgender deterministischer endlicher Automat. Dokumentieren Sie Ihr Vorgehen geeignet.



Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Aj5aei652](http://flaci.com/Aj5aei652)

Lösungsvorschlag

### Minimierungstabelle (Table filling)

— Der Minimierungs-Algorithmus (auch Table-Filling-Algorithmus genannt) trägt in seinem Verlauf eine Markierung in alle diejenigen Zellen der Tabelle ein, die zueinander nicht äquivalente Zustände bezeichnen. Die Markierung „ $x_n$ “ in einer Tabellenzelle  $(i, j)$  bedeutet dabei, dass das Zustandspaar  $(i, j)$  in der  $k$ -ten Iteration des Algorithmus markiert wurde und die Zustände  $i$  und  $j$  somit zueinander  $(k - 1)$ -äquivalent, aber nicht  $k$ -äquivalent und somit insbesondere nicht äquivalent sind. Bleibt eine Zelle bis zum Ende unmarkiert, sind die entsprechenden Zustände zueinander äquivalent.



|       |             |             |             |             |             |             |             |             |             |
|-------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| $z_0$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $z_1$ | $x_3$       | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $z_2$ | $x_3$       | $x_4$       | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $z_3$ |             | $x_3$       | $x_3$       | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $z_4$ | $x_3$       | $x_4$       |             | $x_3$       | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $z_5$ | $x_3$       | $x_4$       |             | $x_3$       |             | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $z_6$ | $x_2$       | $x_2$       | $x_2$       | $x_2$       | $x_2$       | $x_2$       | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $z_7$ | $x_2$       | $x_2$       | $x_2$       | $x_2$       | $x_2$       | $x_2$       | $x_2$       | $\emptyset$ | $\emptyset$ |
| $z_8$ | $x_1$       | $x_1$       | $x_1$       | $x_1$       | $x_1$       | $x_1$       | $x_1$       | $x_1$       | $\emptyset$ |
|       | $z_0$       | $z_1$       | $z_2$       | $z_3$       | $z_4$       | $z_5$       | $z_6$       | $z_7$       | $z_8$       |

$x_1$  Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.

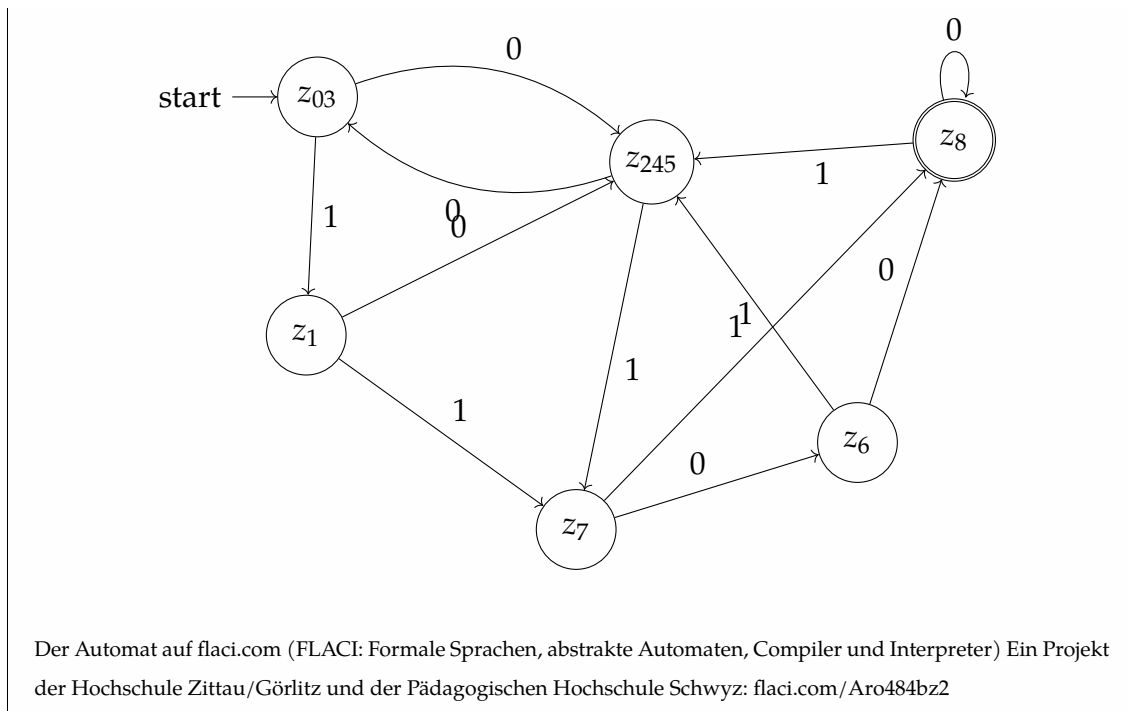
$x_2$  Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

$x_3$  In weiteren Iterationen markierte Zustände.

$x_4$  ...

## Übergangstabelle

| Zustandspaar | 0            | 1                        |
|--------------|--------------|--------------------------|
| $(z_0, z_1)$ | $(z_2, z_5)$ | $(z_1, z_7) \ x_3 \ x_3$ |
| $(z_0, z_2)$ | $(z_2, z_3)$ | $(z_1, z_7) \ x_3$       |
| $(z_0, z_3)$ | $(z_2, z_5)$ | $(z_1, z_1)$             |
| $(z_0, z_4)$ | $(z_2, z_3)$ | $(z_1, z_7) \ x_3$       |
| $(z_0, z_5)$ | $(z_2, z_0)$ | $(z_1, z_7) \ x_3$       |
| $(z_0, z_6)$ | $(z_2, z_8)$ | $(z_1, z_4) \ x_2$       |
| $(z_0, z_7)$ | $(z_2, z_6)$ | $(z_1, z_8) \ x_2$       |
| $(z_1, z_2)$ | $(z_5, z_3)$ | $(z_7, z_7) \ x_4$       |
| $(z_1, z_3)$ | $(z_5, z_5)$ | $(z_7, z_1) \ x_3$       |
| $(z_1, z_4)$ | $(z_5, z_3)$ | $(z_7, z_7) \ x_4$       |
| $(z_1, z_5)$ | $(z_5, z_0)$ | $(z_7, z_7) \ x_4$       |
| $(z_1, z_6)$ | $(z_5, z_8)$ | $(z_7, z_4) \ x_2$       |
| $(z_1, z_7)$ | $(z_5, z_6)$ | $(z_7, z_8) \ x_2$       |
| $(z_2, z_3)$ | $(z_3, z_5)$ | $(z_7, z_1) \ x_3$       |
| $(z_2, z_4)$ | $(z_3, z_3)$ | $(z_7, z_7)$             |
| $(z_2, z_5)$ | $(z_3, z_0)$ | $(z_7, z_7)$             |
| $(z_2, z_6)$ | $(z_3, z_8)$ | $(z_7, z_4) \ x_2$       |
| $(z_2, z_7)$ | $(z_3, z_6)$ | $(z_7, z_8) \ x_2$       |
| $(z_3, z_4)$ | $(z_5, z_3)$ | $(z_1, z_7) \ x_3$       |
| $(z_3, z_5)$ | $(z_5, z_0)$ | $(z_1, z_7) \ x_3$       |
| $(z_3, z_6)$ | $(z_5, z_8)$ | $(z_1, z_4) \ x_2$       |
| $(z_3, z_7)$ | $(z_5, z_6)$ | $(z_1, z_8) \ x_2$       |
| $(z_4, z_5)$ | $(z_3, z_0)$ | $(z_7, z_7)$             |
| $(z_4, z_6)$ | $(z_3, z_8)$ | $(z_7, z_4) \ x_2$       |
| $(z_4, z_7)$ | $(z_3, z_6)$ | $(z_7, z_8) \ x_2$       |
| $(z_5, z_6)$ | $(z_0, z_8)$ | $(z_7, z_4) \ x_2$       |
| $(z_5, z_7)$ | $(z_0, z_6)$ | $(z_7, z_8) \ x_2$       |
| $(z_6, z_7)$ | $(z_8, z_6)$ | $(z_4, z_8) \ x_2$       |



- (b) Beweisen oder widerlegen Sie für folgende Sprachen über dem Alphabet  $\Sigma = \{a, b, c\}$ , dass sie regulär sind.

(i)  $L_1 = \{a^i c u b^j v a c^k \mid u, v \in \{a, b\}^* \text{ und } i, j, k \in \mathbb{N}_0\}$

Lösungsvorschlag

Die Sprache  $L_1$  ist regulär. Nachweis durch regulären Ausdruck:

$$a^* c (a|b)^* b^* (a|b)^* a c^*$$

(ii)  $L_2 = \{a^i c u b^j v a c^k \mid u, v \in \{a, b\}^* \text{ und } i, j, k \in \mathbb{N}_0 \text{ mit } k = i + j\}$

Lösungsvorschlag

Die Sprache  $L_2$  ist nicht regulär. Widerlegung durch das Pumping-Lemma.  
TODO

- (c) Sei  $L$  eine reguläre Sprache über dem Alphabet  $\Sigma$ . Für ein festes Element  $a \in \Sigma$  betrachten wir die Sprache  $L_a = \{aw \mid w \in \Sigma^*, wa \in L\}$ . Zeigen Sie, dass  $L_a$  regulär ist.

Lösungsvorschlag

Die regulären Sprachen sind unter dem Komplement abgeschlossen.

## 66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2

- (a) Sei  $L = \{0^n 1^m 1^p 0^q \mid n + m = p + q \text{ und } n, m, p, q \in \mathbb{N}_0\}$ . Geben Sie eine kontextfreie Grammatik für  $L$  an. Sie dürfen dabei  $\varepsilon$ -Produktionen der Form  $\{A \rightarrow \varepsilon\}$  verwenden.

$$P = \left\{ \begin{array}{l} S \rightarrow 0S0 \mid 0A0 \mid 0B0 \mid \varepsilon \mid A \mid B \mid C \\ A \rightarrow 0A1 \mid 0C1 \\ B \rightarrow 1B0 \mid 1C0 \\ C \rightarrow 1C1 \mid \varepsilon \end{array} \right\}$$

- (b) Für eine Sprache  $L$  sei  $L^r = \{x^r \mid x \in L\}$  die Umkehrsprache. Dabei bezeichne  $x^r$  das Wort, das aus  $x$  entsteht, indem man die Reihenfolge der Zeichen umkehrt, beispielsweise  $(abb)^r = bba$ .
- (i) Sei  $L$  eine kontextfreie Sprache. Zeigen Sie, dass dann auch  $L^r$  kontextfrei ist.
  - (ii) Geben Sie eine kontextfreie Sprache  $L_1$ , an, sodass  $L_1 \cap L_1^r$  kontextfrei ist.
  - (iii) Geben Sie eine kontextfreie Sprache  $L_2$ , an, sodass  $L_2 \cap L_2^r$  nicht kontextfrei ist.

## 66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Eine Hashfunktion  $h$  wird verwendet, um Vornamen auf die Buchstaben  $\{A, \dots, Z\}$  abzubilden. Dabei bildet  $h$  auf den ersten Buchstaben des Vornamens als Hashwert ab.

Sei  $H$  die folgende Hashtabelle (Ausgangszustand):

| Schlüssel | Inhalt |
|-----------|--------|
| A         |        |
| B         |        |
| C         |        |
| D         | Dirk   |
| E         |        |
| F         |        |
| G         |        |
| H         |        |
| I         | Inge   |
| J         |        |
| K         | Kurt   |
| L         |        |

| Schlüssel | Inhalt |
|-----------|--------|
| N         |        |
| O         |        |
| P         |        |
| Q         |        |
| R         |        |
| S         |        |
| T         |        |
| U         |        |
| V         |        |
| W         |        |
| X         |        |
| Y         |        |
| Z         |        |

(a) Fügen Sie der Hashtabelle  $H$  die Vornamen

Martin, Michael, Norbert, Matthias, Alfons, Bert, Christel, Adalbert, Edith, Emil

in dieser Reihenfolge hinzu, wobei Sie Kollisionen durch lineares Sondieren (mit Inkrementieren zum nächsten Buchstaben) behandeln.

Lösungsvorschlag

| Schlüssel | Inhalt   | Schlüssel | Inhalt   |
|-----------|----------|-----------|----------|
| A         | Alfons   | N         | Michael  |
| B         | Bert     | O         | Norbert  |
| C         | Christel | P         | Matthias |
| D         | Dirk     | Q         |          |
| E         | Adalbert | R         |          |
| F         | Edith    | S         |          |
| G         | Emil     | T         |          |
| H         |          | U         |          |
| I         | Inge     | V         |          |
| J         |          | W         |          |
| K         | Kurt     | X         |          |
| L         |          | Y         |          |
| M         | Martin   | Z         |          |

(b) Fügen Sie der Hashtabelle  $H$  die Vornamen

Brigitte, Elmar, Thomas, Katrin, Diana, Nathan, Emanuel, Sebastian, Torsten,  
Karolin

in dieser Reihenfolge hinzu, wobei Sie Kollisionen durch Verkettung der Überläufer behandeln. (Hinweis: Verwenden Sie die Hashtabelle im Ausgangszustand.)

Lösungsvorschlag

|  |
|--|
|  |
|--|

| Schlüssel | Inhalt                | Schlüssel | Inhalt          |
|-----------|-----------------------|-----------|-----------------|
| A         |                       | N         | Nathan          |
| B         | Brigitte              | O         |                 |
| C         |                       | P         |                 |
| D         | Dirk, Diana           | Q         |                 |
| E         | Elmar, Emanuel        | R         |                 |
| F         |                       | S         | Sebastian       |
| G         |                       | T         | Thomas, Torsten |
| H         |                       | U         |                 |
| I         | Inge                  | V         |                 |
| J         |                       | W         |                 |
| K         | Kurt, Katrin, Karolin | X         |                 |
| L         |                       | Y         |                 |
| M         |                       | Z         |                 |

## 66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Beweisen Sie die folgenden Aussagen:

- (a) Sei  $f(n) = 2 \cdot n^2 + 3 \cdot n^2 + 4 \cdot (\log n) + 7$ . Dann gilt  $f \in O(n^2)$ .
- (b) Sei  $f(n) = 4^n$ . Dann gilt nicht  $f \in O(2^n)$ .

Lösungsvorschlag

- (c) Sei  $f(n) = (n+1)!$  (d. h. die Fakultät von  $n+1$ ). Dann gilt  $f \in O(n^n)$ .

Lösungsvorschlag

z. B.  $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 5^5$

- (d) Sei  $f: \mathbb{N} \leftarrow \mathbb{N}$  definiert durch die folgende Rekursionsgleichung:

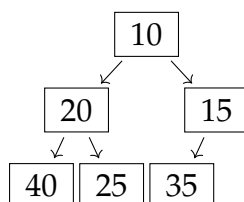
$$f(n) = \begin{cases} 3, & \text{für } n = 1 \\ (n-1)^2 + f(n-1), & \text{für } n > 1 \end{cases}$$

Dann gilt  $f \in \mathcal{O}(n^3)$

$$\begin{aligned}
 f(n) &= (n-1)^2 + f(n-1) + \dots + f(1) \\
 &= (n-1)^2 + (n-1)^2 + f(n-2) + \dots + f(1) \\
 &= \underbrace{(n-1)^2 + \dots + (n-1)^2}_n + 3 \\
 &= \underbrace{(n-1)^2 + \dots + (n-1)^2}_{n-1} + 3 \\
 &= (n-1)^2 \cdot (n-1) + 3 \\
 &= (n-1)^3 + 3
 \end{aligned}$$

### 66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 3

Es sei der folgende Min-Heap gegeben:



- (a) Geben Sie obigen Min-Heap in der Darstellung eines Feldes an, wobei die Knoten in Level-Order abgelegt sind.

Lösungsvorschlag

| 0  | 1  | 2  | 3  | 4  | 5  |
|----|----|----|----|----|----|
| 10 | 20 | 15 | 40 | 25 | 35 |

- (b) Führen Sie wiederholt DeleteMin-Operationen auf dem gegebenen Heap aus, bis der Heap leer ist. Zeichnen Sie dafür den aktuellen Zustand des Heaps als Baum und als Feld nach jeder Änderung des Heaps, wobei Sie nur gültige Bäume zeichnen (d. h. solche die keine Lücken haben). Dokumentieren Sie, was in den einzelnen Schritten geschieht.

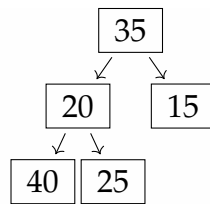
Lösungsvorschlag

#### Löschen von 10

Nach dem Ersetzen von „10“ durch „35“:

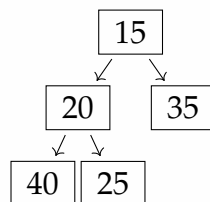
| 0  | 1  | 2  | 3  | 4  |
|----|----|----|----|----|
| 35 | 20 | 15 | 40 | 25 |





Nach dem Vertauschen von „35“ und „15“:

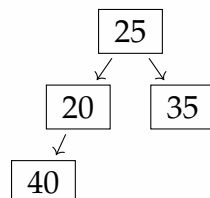
| 0  | 1  | 2  | 3  | 4  |
|----|----|----|----|----|
| 15 | 20 | 35 | 40 | 25 |



## Löschen von 15

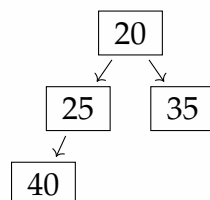
Nach dem Ersetzen von „15“ mit „25“:

| 0  | 1  | 2  | 3  |
|----|----|----|----|
| 25 | 20 | 35 | 40 |



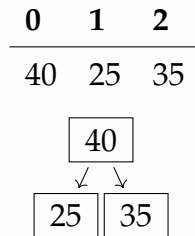
Nach dem Vertauschen von „25“ und „20“:

| 0  | 1  | 2  | 3  |
|----|----|----|----|
| 20 | 25 | 35 | 40 |

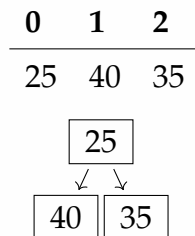


**Löschen von 20**

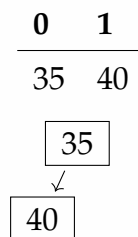
Nach dem Vertauschen von „20“ mit „40“:



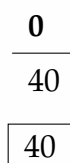
Nach dem Vertauschen von „40“ und „25“:

**Löschen von 25**

Nach dem Ersetzen von „25“ durch „35“:

**Löschen von 35**

Nach dem Ersetzen von „35“ mit „40“:



**Löschen von 40**

Nach dem Löschen von „40“:

**66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 4**

Das GUTSCHEIN-Problem ist gegeben durch eine Folge  $w_1, \dots, w_n$  von Warenwerten (wobei  $w \in \mathbb{N}_0$  für  $i = 1, \dots, n$ ) und einem Gutscheinbetrag  $G \in \mathbb{N}_0$ .

Da Gutscheine nicht in Bargeld ausgezahlt werden können, ist die Frage, ob man eine Teilfolge der Waren findet, sodass man genau den Gutschein ausnutzt. Formal ist dies die Frage, ob es eine Menge von Indizes  $I$  mit  $I \subseteq \{1, \dots, n\}$  gibt, sodass  $\sum_{i \in I} w_i = G$

**Exkurs: Teilsommenproblem**

Das **Teilsommenproblem** (SUBSET SUM oder SSP) ist ein spezielles Rucksackproblem. Gegeben sei eine Menge von ganzen Zahlen  $I = \{w_1, w_2, \dots, w_n\}$ . Gesucht ist eine Untermenge, deren Elementsumme maximal, aber nicht größer als eine gegebene obere Schranke  $c$  ist.

(a) Sei  $w_1 = 10, w_2 = 30, w_3 = 40, w_4 = 20, w_5 = 15$  eine Folge von Warenwerten.

(i) Geben Sie einen Gutscheinbetrag  $40 < G < 115$  an, sodass die GUTSCHEIN-Instanz eine Lösung hat. Geben Sie auch die lösende Menge  $I \subseteq \{1, 2, 3, 4, 5\}$  von Indizes an.

Lösungsvorschlag

50  
 $I = \{1, 3\}$

(ii) Geben Sie einen Gutscheinbetrag  $G$  mit  $40 < G < 115$  an, sodass die GUTSCHEIN-Instanz keine Lösung hat.

Lösungsvorschlag

51

(b) Sei  $table$  eine  $(n \times (G + 1))$ -Tabelle mit Einträgen  $table[i, k]$ , für  $1 \leq i \leq n$  und  $0 \leq k \leq G$ , sodass

$$table[i, k] = \begin{cases} \text{true} & \text{falls es } I \subseteq \{1, \dots, n\} \text{ mit } \sum_{i \in I} w_i = G \text{ gibt} \\ \text{false} & \text{sonst} \end{cases}$$

Geben Sie einen Algorithmus in Pseudo-Code oder Java an, der die Tabelle  $table$  mit *dynamischer Programmierung* in Worst-Case-Laufzeit  $\mathcal{O}(n \times G)$  erzeugt. Begründen Sie die Korrektheit und die Laufzeit Ihres Algorithmus. Welcher Eintrag in  $table$  löst das GUTSCHEIN-Problem?

**Algorithmus 3: Gutschein-Problem**

```

table = boolean array $n + 1 \times (G + 1)$; // Initialisiere ein boolsches Feld
 mit $n + 1$ Zeilen für jeden Warenwert und 0 für keinen Warenwert und mit $G + 1$ Spalten
 für alle Gutscheinbetrag bis G und 0 für keinen Gutscheinbetrag

for k in $1 \dots G$ do ; // Wenn der Gutscheinbetrag größer als 0 ist und es keine
 Warenwerte gibt, $\exists n = 0$, kann der Gutschein nicht eingelöst werden.

 | table[0][k] = false
end

for i in $0 \dots n$ do ; // Ist der Gutscheinbetrag 0, dann kann er immer eingelöst
 werden.

 | table[i][0] = true
end
for i in $1 \dots n$ do ; // Durchlaufe jede Zeile der Warenwerte

 for k in $1 \dots G$ do ; // Durchlaufe jede Spalte der Gutscheinbeträge in dieser
 Zeile

 table[i][k] = table[$i - 1$][k] ; // Übernehme erstmals das Ergebnis der
 Zelle der vorhergehenden Zeile in der gleichen Spalte
 if $k \geq w_i$ und table[i][k] noch nicht markiert then ; // Wenn der
 aktuelle Gutscheinbetrag größer als der aktuelle Warenwert und die aktuelle
 Zelle noch nicht als wahr markiert ist

 | table[i][k] = table[$i - 1$][$k - w_i$] ; // übernimmt das Ergebnis des
 aktuellen Gutscheinbetrags minus des aktuellen Warenwerts
 ;
 end
 end
end

```

```

/**
 * Nach
↳ Subset
 * Sum Problem auf geeksforgeeks.org
 */
public class Gutschein {
 /**
 * @param G Die Indizes der GUTSCHEIN-Beträge.
 *
 * @param W Das GUTSCHEIN-Problem ist gegeben durch eine Folge w_1, \dots, w_n
↳ von
 *
 * Warenwerten.

```

```

*
* @return Wahr, wenn der Gutscheinbetrag vollständig in Warenwerten
→ eingelöst
* werden kann, falsch wenn der Betrag nicht vollständig eingelöst
* werden kann.
*/
public static boolean gutscheinDP(int G, int W[]) {
 // Der Eintrag in der Tabelle tabelle[i][k] ist wahr,
 // wenn es eine Teilsumme der
 // W[0..i-1] gibt, die gleich k ist.
 int n = W.length;
 boolean table[][] = new boolean[n + 1][G + 1];

 // Wenn der Gutschein-Betrag größer als 0 ist und es keine
 // Warenwerte (n = 0) gibt, kann der Gutschein nicht eingelöst
 // werden.
 for (int k = 1; k <= G; k++) {
 table[0][k] = false;
 }

 // Ist der Gutscheinbetrag 0, dann kann er immer eingelöst werden.
 for (int i = 0; i <= n; i++) {
 table[i][0] = true;
 }

 for (int i = 1; i <= n; i++) {
 for (int k = 1; k <= G; k++) {
 table[i][k] = table[i - 1][k];
 // Warenwert
 int w = W[i - 1];
 if (k >= w && !table[i][k]) {
 table[i][k] = table[i - 1][k - w];
 }
 }
 }
 return table[n][G];
}

public static void main(String[] args) {
 System.out.println(gutscheinDP(50, new int[] { 10, 30, 40, 20, 15 }));
 System.out.println(gutscheinDP(41, new int[] { 10, 30, 40, 20, 15 }));

 System.out.println(gutscheinDP(3, new int[] { 1, 2, 3 }));
 System.out.println(gutscheinDP(5, new int[] { 1, 2, 3 }));
 System.out.println(gutscheinDP(6, new int[] { 1, 2, 3 }));
 System.out.println(gutscheinDP(2, new int[] { 1, 2, 3 }));
 System.out.println(gutscheinDP(1, new int[] { 1, 2, 3 }));
 System.out.println(gutscheinDP(7, new int[] { 1, 2, 3 }));
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/Gutschein.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/Gutschein.java)

Die äußere for-Schleife läuft  $n$  mal und die innere for-Schleife  $G$  mal.

Der letzte Eintrag in der Tabelle, also der Wert in der Zelle `table[W.length]` `[G]`, löst das Gutscheinproblem. Steht hier `true`, dann gibt es eine Teilfolge der Waren, die den Gutscheinbetrag genau ausnutzt.

## 66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 5

Eine Folge von Zahlen ist eine *odd-ascending-even-descending*-Folge, wenn gilt:

Zunächst enthält die Folge alle Schlüssel, die *ungerade* Zahlen sind, und diese Schlüssel sind aufsteigend sortiert angeordnet. Im Anschluss daran enthält die Folge alle Schlüssel, die *gerade* Zahlen sind, und diese Schlüssel sind absteigend sortiert angeordnet.

- (a) Geben Sie die Zahlen 10, 3, 11, 20, 8, 4, 9 als *odd-ascending-even-descending*-Folge an.

Lösungsvorschlag

3, 9, 11, 20, 10, 8, 4

- (b) Geben Sie einen Algorithmus (z. B. in Pseudocode oder Java) an, der für eine *odd-ascending-even-descending*-Folge  $F$  gegeben als Feld und einem Schlüsselwert  $S$  prüft, ob  $S$  in  $F$  vorkommt und `true` im Erfolgsfall und ansonsten `false` liefert. Dabei soll der Algorithmus im Worst-Case eine echt bessere Laufzeit als Linearzeit (in der Größe der Arrays) haben. Erläutern Sie Ihren Algorithmus und begründen Sie die Korrektheit.

Lösungsvorschlag

Bei dem Algorithmus handelt es sich um einen leicht abgewandelten Code, der eine „klassische“ binären Suche implementiert.

```
public static boolean suche(int[] feld, int schlüssel) {
 int links = 0, rechts = feld.length - 1;
 boolean istGerade = schlüssel % 2 == 0;
 while (links <= rechts) {
 int mitte = links + (rechts - links) / 2;
 if (feld[mitte] == schlüssel) {
 return true;
 }
 // Verschiebe die linke Grenze nach rechts, wenn die gesuchte
 // Zahl gerade ist und die Zahl in der Mitte größer als die
 // gesuchte Zahl ist oder wenn die gesuchte Zahl ungerade ist
 // und die Zahl in der Mitte kleiner.
 if ((istGerade && feld[mitte] > schlüssel) || (!istGerade &&
 ↪ feld[mitte] < schlüssel)) {
 // nach rechts verschieben
 links = mitte + 1;
 } else {
 // nach links verschieben
 rechts = mitte - 1;
 }
 }
 return false;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/UngeradeGerade.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/UngeradeGerade.java)

- (c) Erläutern Sie schrittweise den Ablauf Ihres Algorithmus für die Folge 1, 5, 11, 8, 4, 2 und den Suchschlüssel 4.

Lösungsvorschlag

Die erste Zeile der Methode `suche` initialisiert die Variable `links` mit 0 und `rechts` mit 5. Da `links` kleiner ist als `rechts`, wird die `while`-Schleife betreten und die Variable `mitte` auf 2 gesetzt. Da der gesuchte Schlüssel gerade ist und `feld[2]` 11 ist, also größer, wird in den `true`-Block der `if`-Bedingung besprungen und die Variable `links` auf 3 gesetzt.

Zu Beginn des 2. Durchlaufs der `while`-Schleife ergeben sich folgende Werte: `links`: 3 `mitte`: 4 `rechts`: 5.

In der anschließenden Bedingten Anweisung wird die `while`-Schleife verlassen und `true` zurückgegeben, da mit `feld[4]` der gewünschte Schlüssel gefunden wurde.

- (d) Analysieren Sie die Laufzeit Ihres Algorithmus für den Worst-Case, geben Sie diese in  $\mathcal{O}$ -Notation an und begründen Sie diese.

Lösungsvorschlag

Die Laufzeit des Algorithmuses ist in der Ordnung  $\mathcal{O}(\log_2 n)$ .

Im schlechtesten Fall muss nicht die gesamte Folge durchsucht werden. Nach dem ersten Teilen der Folge bleiben nur noch  $\frac{n}{2}$  Elemente, nach dem zweiten Schritt  $\frac{n}{4}$ , nach dem dritten  $\frac{n}{8}$  usw. Allgemein bedeutet dies, dass im  $i$ -ten Durchlauf maximal  $\frac{n}{2^i}$  Elemente zu durchsuchen sind. Entsprechend werden  $\log_2 n$  Schritte benötigt.

## Kompletter Code

```
public class UngeradeGerade {

 public static boolean suche(int[] feld, int schlüssel) {
 int links = 0, rechts = feld.length - 1;
 boolean istGerade = schlüssel % 2 == 0;
 while (links <= rechts) {
 int mitte = links + (rechts - links) / 2;
 if (feld[mitte] == schlüssel) {
 return true;
 }
 // Verschiebe die linke Grenze nach rechts, wenn die gesuchte
 // Zahl gerade ist und die Zahl in der Mitte größer als die
 // gesuchte Zahl ist oder wenn die gesuchte Zahl ungerade ist
 // und die Zahl in der Mitte kleiner.
 if ((istGerade && feld[mitte] > schlüssel) || (!istGerade && feld[mitte] <
 ↪ schlüssel)) {
 // nach rechts verschieben
 }
 }
 }
}
```

```

 links = mitte + 1;
 } else {
 // nach links verschieben
 rechts = mitte - 1;
 }
}
return false;
}

public static void main(String[] args) {
 System.out.println(suche(new int[] { 1, 5, 11, 8, 4, 2 }, 4));
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/UngeradeGerade.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/UngeradeGerade.java)

## Test-Code

```

import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class UngeradeGeradeTest {

 private void assertSucheUnGerade(int[] feld, int suche, boolean ergebnis) {
 assertEquals(ergebnis, UngeradeGerade.suche(feld, suche));
 }

 @Test
 public void assertSucheUnGerade() {
 int[] feld = new int[] { 1, 3, 5, 7, 9, 10, 8, 6, 4, 2 };
 assertSucheUnGerade(feld, 4, true);
 assertSucheUnGerade(feld, 11, false);
 assertSucheUnGerade(feld, 0, false);
 assertSucheUnGerade(feld, 3, true);
 }
}

```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2020/herbst/UngeradeGeradeTest.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/UngeradeGeradeTest.java)

## 66115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1

Im Folgenden bezeichnet  $a^i = a \dots a$  und  $\varepsilon$  steht für das leere Wort (öinsbesondere  $a^i = \varepsilon$ ).

Die Menge  $\mathbb{N}_0 = \{0, 1, 2, \dots\}$  ist die Menge aller nicht-negativer Ganzzahlen.

Die Sprachen  $L_1, \dots, L_{12}$  seien definiert als:

- (a) Ordnen Sie jedem der folgenden nichtdeterministischen endlichen Automaten  $N_j, j = 1, \dots, 6$ , (die alle über dem Alphabet  $\Sigma = \{a\}$  arbeiten) **jeweils eine** der Sprachen  $L_i \in \{L_1, \dots, L_{12}\}$  zu, sodass  $L_i$ , genau die von  $N_i$ , **akzeptierte Sprache** ist.



- $N_1 = L_6$  (mindestens ein  $a$ )
- $N_2 = L_8$  (ungerade Anzahl an  $a$ 's:  $1, 5, 7, \dots$ )
- $N_3 = L_2$  (gerade Anzahl an  $a$ 's:  $2, 4, 6, \dots$ )
- $N_4 = L_{12}$  (leeres Wort)
- $N_5 = L_8$  (ungerade Anzahl an  $a$ 's:  $1, 5, 7, \dots$ )
- $N_6 = L_{11}$  (die Sprache akzeptiert nicht)

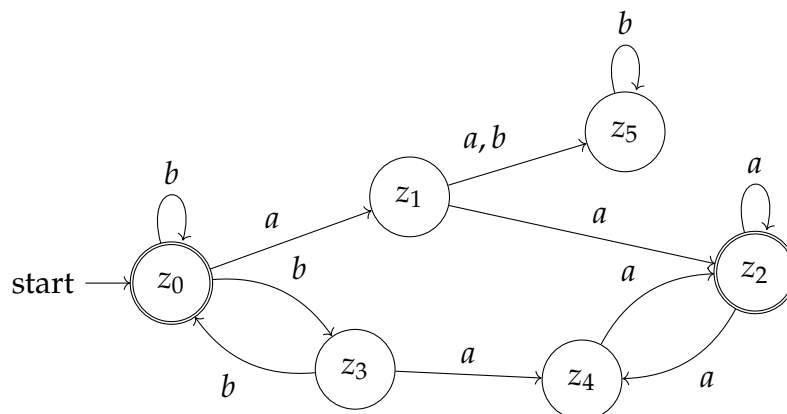
(b) Zeigen Sie für eine der Sprachen  $L_1, \dots, L_{12}$  dass diese **nicht regulär** ist.

$$L_1 0 = \{ a^n \mid n \in \mathbb{N}_0, n \text{ ist Primzahl} \}$$

ist nicht regulär, da sich sonst jede Primzahl  $p$  einer bestimmten Mindestgröße  $j$  als Summe von natürlichen Zahlen  $u + v + w$  darstellen ließe, so dass  $v \geq 1$  und für alle  $i \geq 0$  auch  $u + iv + w = p + (i1)v$  prim ist. Dies ist jedoch für  $i = p + 1$  wegen  $p + (p + 11)v = p(1 + v)$  nicht der Fall.<sup>a</sup>

<sup>a</sup><https://www.informatik.hu-berlin.de/de/forschung/gebiete/algorithmenII/Lehre/ws13/einftheo/einftheo-skript.pdf>

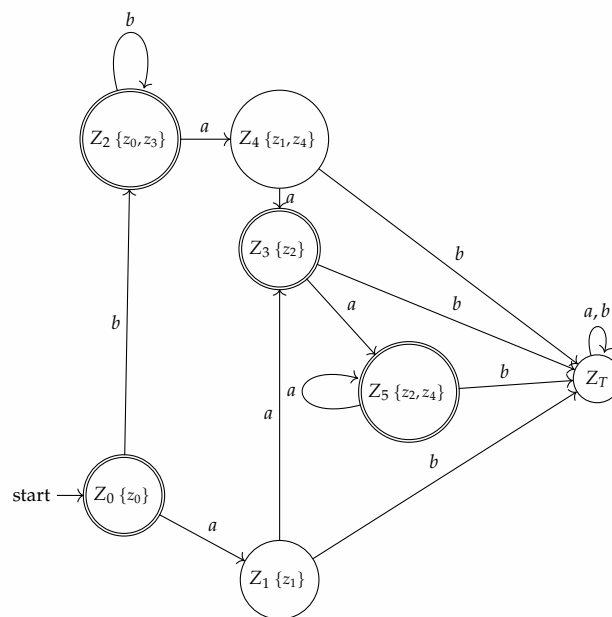
(c) Konstruieren Sie für den folgenden nichtdeterministischen endlichen Automaten (der Worte über dem Alphabet  $\Sigma = \{a, b\}$  verarbeitet) einen äquivalenten deterministischen endlichen Automaten mithilfe der Potenzmengenkonstruktion. Zeichnen Sie dabei nur die vom Startzustand erreichbaren Zustände. Erläutern Sie Ihr Vorgehen.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Af7iooyca](https://flaci.com/Af7iooyca)

| Zustandsmenge      | Eingabe $a$        | Eingabe $b$        |
|--------------------|--------------------|--------------------|
| $Z_0 \{z_0\}$      | $Z_1 \{z_1\}$      | $Z_2 \{z_0, z_3\}$ |
| $Z_1 \{z_1\}$      | $Z_3 \{z_2\}$      | $Z_T$              |
| $Z_2 \{z_0, z_3\}$ | $Z_4 \{z_1, z_4\}$ | $Z_2 \{z_0, z_3\}$ |
| $Z_3 \{z_2\}$      | $Z_5 \{z_2, z_4\}$ | $Z_T$              |
| $Z_4 \{z_1, z_4\}$ | $Z_3 \{z_2\}$      | $Z_T$              |
| $Z_5 \{z_2, z_4\}$ | $Z_5 \{z_2, z_4\}$ | $Z_T$              |

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Apkyuo4ja](http://flaci.com/Apkyuo4ja)

## 66115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Sei  $G = (V, \Sigma, P, S)$  eine kontextfreie Grammatik mit Variablen  $V = \{S, A, B, C, D\}$ , Terminalzeichen  $\Sigma = \{a, b, c\}$ , Produktionen

$$P = \left\{ \begin{array}{l} S \rightarrow AD \mid CC \mid c \\ A \rightarrow a \\ B \rightarrow b \\ C \rightarrow CC \mid c \\ D \rightarrow SB \mid CB \end{array} \right.$$

und Startsymbol  $S$ . Führen Sie den Algorithmus von Cocke, Younger und Kasami (CYK-Algorithmus) für  $G$  und das Wort  $aaacbbbb$  aus. Liegt  $aaacbbbb$  in der durch  $G$  erzeugten Sprache? Erläutern Sie Ihr Vorgehen und den Ablauf des CYK-Algorithmus.

Lösungsvorschlag

|     |     |     |     |     |   |   |   |
|-----|-----|-----|-----|-----|---|---|---|
| a   | a   | a   | c   | c   | b | b | b |
| -   | -   | -   | S,C | D,D | - | - |   |
| -   | -   | -   | D,D | -   | - |   |   |
| -   | -   | S,S | -   | -   |   |   |   |
| -   | -   | D,D | -   |     |   |   |   |
| -   | S,S | -   |     |     |   |   |   |
| -   | D,D |     |     |     |   |   |   |
| S,S |     |     |     |     |   |   |   |

Das Wort  $aaacbbbb$  liegt in der Sprache.

## 66115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 1

- (a) Geben Sie für folgende Sortiervverfahren jeweils zwei Felder  $A$  und  $B$  an, so dass das jeweilige Sortiervverfahren angewendet auf  $A$  seine Best-Case-Laufzeit und angewendet auf  $B$  seine Worst-Case-Laufzeit erreicht. (Wir messen die Laufzeit durch die Anzahl der Vergleiche zwischen Elementen der Eingabe.) Dabei soll das Feld  $A$  die Zahlen  $1, 2, \dots, 7$  genau einmal enthalten; das Feld  $B$  ebenso. Sie bestimmen also nur die Reihenfolge der Zahlen.

Wenden Sie als Beleg für Ihre Aussagen das jeweilige Sortiervverfahren auf die Felder  $A$  und  $B$  an und geben Sie nach jedem größeren Schritt des Algorithmus den Inhalt der Felder an.

Geben Sie außerdem für jedes Verfahren asymptotische Best- und Worst-Case-Laufzeit für ein Feld der Länge  $n$  an.

Die im Pseudocode verwendete Unteroutine  $\text{Swap}(A, i, j)$  vertauscht im Feld  $A$  die jeweiligen Elemente mit den Indizes  $i$  und  $j$  miteinander.

### (i) Insertionsort

Lösungsvorschlag

#### Best-Case

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|

|   |    |    |   |   |   |   |                |
|---|----|----|---|---|---|---|----------------|
| 1 | 2  | 3  | 4 | 5 | 6 | 7 | Eingabe        |
| 1 | 2* | 3  | 4 | 5 | 6 | 7 | markiere (i 1) |
| 1 | 2  | 3* | 4 | 5 | 6 | 7 | markiere (i 2) |

```

1 2 3 4* 5 6 7 markiere (i 3)
1 2 3 4 5* 6 7 markiere (i 4)
1 2 3 4 5 6* 7 markiere (i 5)
1 2 3 4 5 6 7* markiere (i 6)
1 2 3 4 5 6 7 Ausgabe

```

**Worst-Case**

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|

```

7 6 5 4 3 2 1 Eingabe
7 6* 5 4 3 2 1 markiere (i 1)
>7 7< 5 4 3 2 1 vertausche (i 0<>1)
6 7 5* 4 3 2 1 markiere (i 2)
6 >7 7< 4 3 2 1 vertausche (i 1<>2)
>6 6< 7 4 3 2 1 vertausche (i 0<>1)
5 6 7 4* 3 2 1 markiere (i 3)
5 6 >7 7< 3 2 1 vertausche (i 2<>3)
5 >6 6< 7 3 2 1 vertausche (i 1<>2)
>5 5< 6 7 3 2 1 vertausche (i 0<>1)
4 5 6 7 3* 2 1 markiere (i 4)
4 5 6 >7 7< 2 1 vertausche (i 3<>4)
4 5 >6 6< 7 2 1 vertausche (i 2<>3)
4 >5 5< 6 7 2 1 vertausche (i 1<>2)
>4 4< 5 6 7 2 1 vertausche (i 0<>1)
3 4 5 6 7 2* 1 markiere (i 5)
3 4 5 6 >7 7< 1 vertausche (i 4<>5)
3 4 5 >6 6< 7 1 vertausche (i 3<>4)
3 4 >5 5< 6 7 1 vertausche (i 2<>3)
3 >4 4< 5 6 7 1 vertausche (i 1<>2)
>3 3< 4 5 6 7 1 vertausche (i 0<>1)
2 3 4 5 6 7 1* markiere (i 6)
2 3 4 5 6 >7 7< vertausche (i 5<>6)
2 3 4 5 >6 6< 7 vertausche (i 4<>5)
2 3 4 >5 5< 6 7 vertausche (i 3<>4)
2 3 >4 4< 5 6 7 vertausche (i 2<>3)
2 >3 3< 4 5 6 7 vertausche (i 1<>2)
>2 2< 3 4 5 6 7 vertausche (i 0<>1)
1 2 3 4 5 6 7 Ausgabe

```

- (ii) Standardversion von **Quicksort** (Pseudocode s.u., Feldindizes beginnen bei 1), bei der das letzte Element eines Teilfeldes als Pivot-Element gewählt wird.

Lösungsvorschlag

**Best-Case**

**Funktion Quicksort**( $A, l = 1, r = A.length$ )

```

if $l < r$ then
 | $m = \text{Partition}(A, l, r);$
 | $\text{Quicksort}(A, l, m - 1);$
 | $\text{Quicksort}(A, m + 1, r);$
end

```

**Funktion Partition**( $A, \text{int } l, \text{int } r$ )

```

 $\text{pivot} = A[r];$
 $i = l;$
for $j = l$ to $r - 1$ do
 | if $A[j] < \text{pivot}$ then
 | | $\text{Swap}(A, i, j);$
 | | $i = i + 1;$
 | end
end

```

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 1 | 3 | 2 | 6 | 5 | 7 | 4 |
|---|---|---|---|---|---|---|

```

1 3 2 6 5 7 4 zerlege
1 3 2 6 5 7 4* markiere (i 6)
>1< 3 2 6 5 7 4 vertausche (i 0<>0)
1 >3< 2 6 5 7 4 vertausche (i 1<>1)
1 3 >2< 6 5 7 4 vertausche (i 2<>2)
1 3 2 >6 5 7 4< vertausche (i 3<>6)
1 3 2 zerlege
1 3 2* markiere (i 2)
>1< 3 2 vertausche (i 0<>0)
1 >3 2< vertausche (i 1<>2)
5 7 6 zerlege
5 7 6* markiere (i 6)
>5< 7 6 vertausche (i 4<>4)
5 >7 6< vertausche (i 5<>6)

```

**Worst-Case**

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|

```

1 2 3 4 5 6 7 zerlege
1 2 3 4 5 6 7* markiere (i 6)
>1< 2 3 4 5 6 7 vertausche (i 0<>0)
1 >2< 3 4 5 6 7 vertausche (i 1<>1)

```

```

1 2 >3< 4 5 6 7 vertausche (i 2<>2)
1 2 3 >4< 5 6 7 vertausche (i 3<>3)
1 2 3 4 >5< 6 7 vertausche (i 4<>4)
1 2 3 4 5 >6< 7 vertausche (i 5<>5)
1 2 3 4 5 6 >7< vertausche (i 6<>6)
1 2 3 4 5 6 zerlege
1 2 3 4 5 6* markiere (i 5)
>1< 2 3 4 5 6 vertausche (i 0<>0)
1 >2< 3 4 5 6 vertausche (i 1<>1)
1 2 >3< 4 5 6 vertausche (i 2<>2)
1 2 3 >4< 5 6 vertausche (i 3<>3)
1 2 3 4 >5< 6 vertausche (i 4<>4)
1 2 3 4 5 >6< vertausche (i 5<>5)
1 2 3 4 5 zerlege
1 2 3 4 5* markiere (i 4)
>1< 2 3 4 5 vertausche (i 0<>0)
1 >2< 3 4 5 vertausche (i 1<>1)
1 2 >3< 4 5 vertausche (i 2<>2)
1 2 3 >4< 5 vertausche (i 3<>3)
1 2 3 4 >5< vertausche (i 4<>4)
1 2 3 4 zerlege
1 2 3 4* markiere (i 3)
>1< 2 3 4 vertausche (i 0<>0)
1 >2< 3 4 vertausche (i 1<>1)
1 2 >3< 4 vertausche (i 2<>2)
1 2 3 >4< vertausche (i 3<>3)
1 2 3 zerlege
1 2 3* markiere (i 2)
>1< 2 3 vertausche (i 0<>0)
1 >2< 3 vertausche (i 1<>1)
1 2 >3< vertausche (i 2<>2)
1 2 zerlege
1 2* markiere (i 1)
>1< 2 vertausche (i 0<>0)
1 >2< vertausche (i 1<>1)

```

- (iii) **QuicksortVar:** Variante von Quicksort, bei der immer das mittlere Element eines Teilfeldes als Pivot-Element gewählt wird (Pseudocode s.u., nur eine Zeile neu).

Bei einem Aufruf von PartitionVar auf ein Teilfeld  $A[l \dots r]$  wird also erst mithilfe der Unteroutine Swap  $A \left[ \left\lfloor \frac{l+r-1}{2} \right\rfloor \right]$  mit  $A[r]$  vertauscht.

**Funktion** QuicksortVar( $A, l = 1, r = A.length$ )

```
if $l < r$ then
 $m = \text{PartitionVar}(A, l, r);$
 QuicksortVar($A, l, m - 1$);
 QuicksortVar($A, m + 1, r$);
end
```

**Funktion** PartitionVar( $A, \text{int } l, \text{int } r$ )

```
Swap($A, \lfloor \frac{l+r-1}{2} \rfloor, r$);
pivot = $A[r]$;
 $i = l$;
for $j = l$ to $r - 1$ do
 if $A[j] < \text{pivot}$ then
 Swap(A, i, j);
 $i = i + 1$;
 end
end
end
```

Lösungsvorschlag

**Best-Case**

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|

```

1 2 3 4 5 6 7 zerlege
1 2 3 4* 5 6 7 markiere (i 3)
1 2 3 >4 5 6 7< vertausche (i 3<>6)
>1< 2 3 7 5 6 4 vertausche (i 0<>0)
1 >2< 3 7 5 6 4 vertausche (i 1<>1)
1 2 >3< 7 5 6 4 vertausche (i 2<>2)
1 2 3 >7 5 6 4< vertausche (i 3<>6)
1 2 3
zerlege
1 2* 3
markiere (i 1)
1 >2 3<
vertausche (i 1<>2)
>1< 3 2
vertausche (i 0<>0)
1 >3 2<
vertausche (i 1<>2)
5 6 7 zerlege
5 6* 7 markiere (i 5)
5 >6 7< vertausche (i 5<>6)
>5< 7 6 vertausche (i 4<>4)
5 >7 6< vertausche (i 5<>6)
1 2 3 4 5 6 7 Ausgabe

```

### Worst-Case

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 2 | 4 | 6 | 7 | 1 | 5 | 3 |
|---|---|---|---|---|---|---|

```

2 4 6 7 1 5 3 zerlege
2 4 6 7* 1 5 3 markiere (i 3)
2 4 6 >7 1 5 3< vertausche (i 3<>6)
>2< 4 6 3 1 5 7 vertausche (i 0<>0)
2 >4< 6 3 1 5 7 vertausche (i 1<>1)
2 4 >6< 3 1 5 7 vertausche (i 2<>2)
2 4 6 >3< 1 5 7 vertausche (i 3<>3)
2 4 6 3 >1< 5 7 vertausche (i 4<>4)
2 4 6 3 1 >5< 7 vertausche (i 5<>5)
2 4 6 3 1 5 >7< vertausche (i 6<>6)
2 4 6 3 1 5
zerlege
2 4 6* 3 1 5
markiere (i 2)
2 4 >6 3 1 5<
vertausche (i 2<>5)
>2< 4 5 3 1 6
vertausche (i 0<>0)
2 >4< 5 3 1 6
vertausche (i 1<>1)
2 4 >5< 3 1 6
vertausche (i 2<>2)
2 4 5 >3< 1 6
vertausche (i 3<>3)
2 4 5 3 >1< 6
vertausche (i 4<>4)
2 4 5 3 1 >6<
vertausche (i 5<>5)
2 4 5 3 1
zerlege
2 4 5* 3 1
markiere (i 2)
2 4 >5 3 1<
vertausche (i 2<>4)

```



|             |                     |
|-------------|---------------------|
| >2< 4 1 3 5 | vertausche (i 0<>0) |
| 2 >4< 1 3 5 | vertausche (i 1<>1) |
| 2 4 >1< 3 5 | vertausche (i 2<>2) |
| 2 4 1 >3< 5 | vertausche (i 3<>3) |
| 2 4 1 3 >5< | vertausche (i 4<>4) |
| 2 4 1 3     | zerlege             |
| 2 4* 1 3    | markiere (i 1)      |
| 2 >4 1 3<   | vertausche (i 1<>3) |
| >2< 3 1 4   | vertausche (i 0<>0) |
| 2 >3< 1 4   | vertausche (i 1<>1) |
| 2 3 >1< 4   | vertausche (i 2<>2) |
| 2 3 1 >4<   | vertausche (i 3<>3) |
| 2 3 1       | zerlege             |
| 2 3* 1      | markiere (i 1)      |
| 2 >3 1<     | vertausche (i 1<>2) |
| >2< 1 3     | vertausche (i 0<>0) |
| 2 >1< 3     | vertausche (i 1<>1) |
| 2 1 >3<     | vertausche (i 2<>2) |
| 2 1         | zerlege             |
| 2* 1        | markiere (i 0)      |
| >2 1<       | vertausche (i 0<>1) |
| >1< 2       | vertausche (i 0<>0) |
| 1 >2<       | vertausche (i 1<>1) |

(b) Geben Sie die asymptotische Best- und Worst-Case-Laufzeit von **Mergesort** an.

Lösungsvorschlag

Best-Case:  $\mathcal{O}(n \cdot \log(n))$

Worst-Case:  $\mathcal{O}(n^2)$

## 66115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2

(a) Argumentieren Sie, warum man das Maximum von  $n$  Zahlen nicht mit weniger als  $n - 1$  Vergleichen bestimmen kann.

Lösungsvorschlag

Wenn die  $n$  Zahlen in einem unsortierten Zustand vorliegen, müssen wir alle Zahlen betrachten, um das Maximum zu finden. Wir brauchen dazu  $n - 1$  und nicht  $n$  Vergleiche, da wir die erste Zahl zu Beginn des Algorithmus als Maximum definieren und anschließend die verbleibenden Zahlen  $n - 1$  mit dem aktuellen Maximum vergleichen.

(b) Geben Sie einen Algorithmus im Pseudocode an, der das Maximum eines Feldes der Länge  $n$  mit genau  $n - 1$  Vergleichen bestimmt.

```
public static int bestimmeMaximum(int[] a) {
 int max = a[0];
 for (int i = 1; i < a.length; i++) {
 if (a[i] > max) {
 max = a[i];
 }
 }
 return max;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2021/fruehjahr/MinimumMaximum.java](https://github.com/orgs/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java)

- (c) Wenn man das Minimum und das Maximum von  $n$  Zahlen bestimmen will, dann kann das natürlich mit  $2n - 2$  Vergleichen erfolgen. Zeigen Sie, dass man bei jedem beliebigen Feld mit deutlich weniger Vergleichen auskommt, wenn man die beiden Werte statt in zwei separaten Durchläufen in einem Durchlauf geschickt bestimmt.

```
/**
 * Diese Methode ist nicht optimiert. Es werden $2n - 2$ Vergleiche benötigt.
 *
 * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
→ gesucht
 * werden soll.
 *
 * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das
→ Minimum,
 * der zweite Eintrag das Maximum.
 */
public static int[] minMaxNaiv(int[] a) {
 int max = a[0];
 int min = a[0];
 for (int i = 1; i < a.length; i++) {
 if (a[i] > max) {
 max = a[i];
 }
 if (a[i] < min) {
 min = a[i];
 }
 }
 return new int[] { min, max };
}

/**
 * Diese Methode ist optimiert. Es werden immer zwei Zahlen paarweise
 * betrachtet. Die Anzahl der Vergleiche reduziert sich auf $3n/2 + 2$ bzw.
 * $3(n-1)/2 + 4$ bei einer ungeraden Anzahl an Zahlen im Feld.
 *
 * nach <a href=
 * "https://www.techiedelight.com/find-minimum-maximum-element-array-using-
→ minimum-comparisons/">techiedelight.com
 *

```

```
* @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
→ gesucht
* werden soll.
*
* @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das
→ Minimum,
* der zweite Eintrag das Maximum.
*/
public static int[] minMaxIterativPaarweise(int[] a) {
 int max = Integer.MIN_VALUE, min = Integer.MAX_VALUE;
 int n = a.length;

 boolean istUngerade = (n & 1) == 1;
 if (istUngerade) {
 n--;
 }

 for (int i = 0; i < n; i = i + 2) {
 int maximum, minimum;

 if (a[i] > a[i + 1]) {
 minimum = a[i + 1];
 maximum = a[i];
 } else {
 minimum = a[i];
 maximum = a[i + 1];
 }

 if (maximum > max) {
 max = maximum;
 }

 if (minimum < min) {
 min = minimum;
 }
 }

 if (istUngerade) {
 if (a[n] > max) {
 max = a[n];
 }

 if (a[n] < min) {
 min = a[n];
 }
 }
 return new int[] { min, max };
}

/**
 * Diese Methode ist nach dem Teile-und-Herrsche-Prinzip optimiert. Er
 * funktioniert so ähnlich wie der Mergesort.
 *
 * nach <a href=
```

```

* "https://www.enjoyalgorithms.com/blog/find-the-minimum-and-maximum-
→ value-in-an-array">enjoyalgorithms.com
*
* @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
* gesucht werden soll.
* @param l Die linke Grenze.
* @param r Die rechts Grenze.
*
* @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das
→ Minimum,
* der zweite Eintrag das Maximum.
*/
int[] minMaxRekursiv(int[] a, int l, int r) {
 int max, min;
 if (l == r) {
 max = a[l];
 min = a[l];
 } else if (l + 1 == r) {
 if (a[l] < a[r]) {
 max = a[r];
 min = a[l];
 } else {
 max = a[l];
 min = a[r];
 }
 } else {
 int mid = l + (r - l) / 2;
 int[] lErgebnis = minMaxRekursiv(a, l, mid);
 int[] rErgebnis = minMaxRekursiv(a, mid + 1, r);
 if (lErgebnis[0] > rErgebnis[0]) {
 max = lErgebnis[0];
 } else {
 max = rErgebnis[0];
 }
 if (lErgebnis[1] < rErgebnis[1]) {
 min = lErgebnis[1];
 } else {
 min = rErgebnis[1];
 }
 }
 int[] ergebnis = { max, min };
 return ergebnis;
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2021/fruehjahr/MinimumMaximum.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java)

## 66115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 3

Wir betrachten eine Variante der Breitensuche (BFS), bei der die Knoten markiert werden, wenn sie das erste Mal besucht werden. Außerdem wird die Suche einmal bei jedem unmarkierten Knoten gestartet, bis alle Knoten markiert sind. Wir betrachten gerichtete Graphen. Ein gerichteter Graph  $G$  ist *schwach zusammenhängend*, wenn der unge-

richtete Graph (der sich daraus ergibt, dass man die Kantenrichtungen von  $G$  ignoriert) zusammenhängend ist.

### Exkurs: Schwach zusammenhängend gerichteter Graph

Beim gerichteten Graphen musst du auf die Kantenrichtung achten. Würde man die Richtungen der Kanten ignorieren wäre aber trotzdem jeder Knoten erreichbar. Einen solchen Graphen nennt man schwach zusammenhängend.<sup>a</sup>

Ein gerichteter Graph heißt (schwach) zusammenhängend, falls der zugehörige ungerichtete Graph (also der Graph, der entsteht, wenn man jede gerichtete Kante durch eine ungerichtete Kante ersetzt) zusammenhängend ist.<sup>b</sup>

<sup>a</sup><https://studyflix.de/informatik/grundbegriffe-der-graphentheorie-1285>

<sup>b</sup>[https://de.wikipedia.org/wiki/Zusammenhang\\_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Zusammenhang_(Graphentheorie))

- (a) Beschreiben Sie für ein allgemeines  $n \in \mathbb{N}$  mit  $n \geq 2$  den Aufbau eines schwach zusammenhängenden Graphen  $G_n$ , mit  $n$  Knoten, bei dem die Breitensuche  $\Theta(n)$  mal gestartet werden muss, bis alle Knoten markiert sind.

Lösungsvorschlag

?

Die Breitensuche benötigt einen Startknoten. Die unten aufgeführten Graphen finden immer nur einen Knoten nämlich den Startknoten.

Oder so:

```

 graph TD
 B[B] --> A[A]
 D[D] --> A
 C[C] --> A

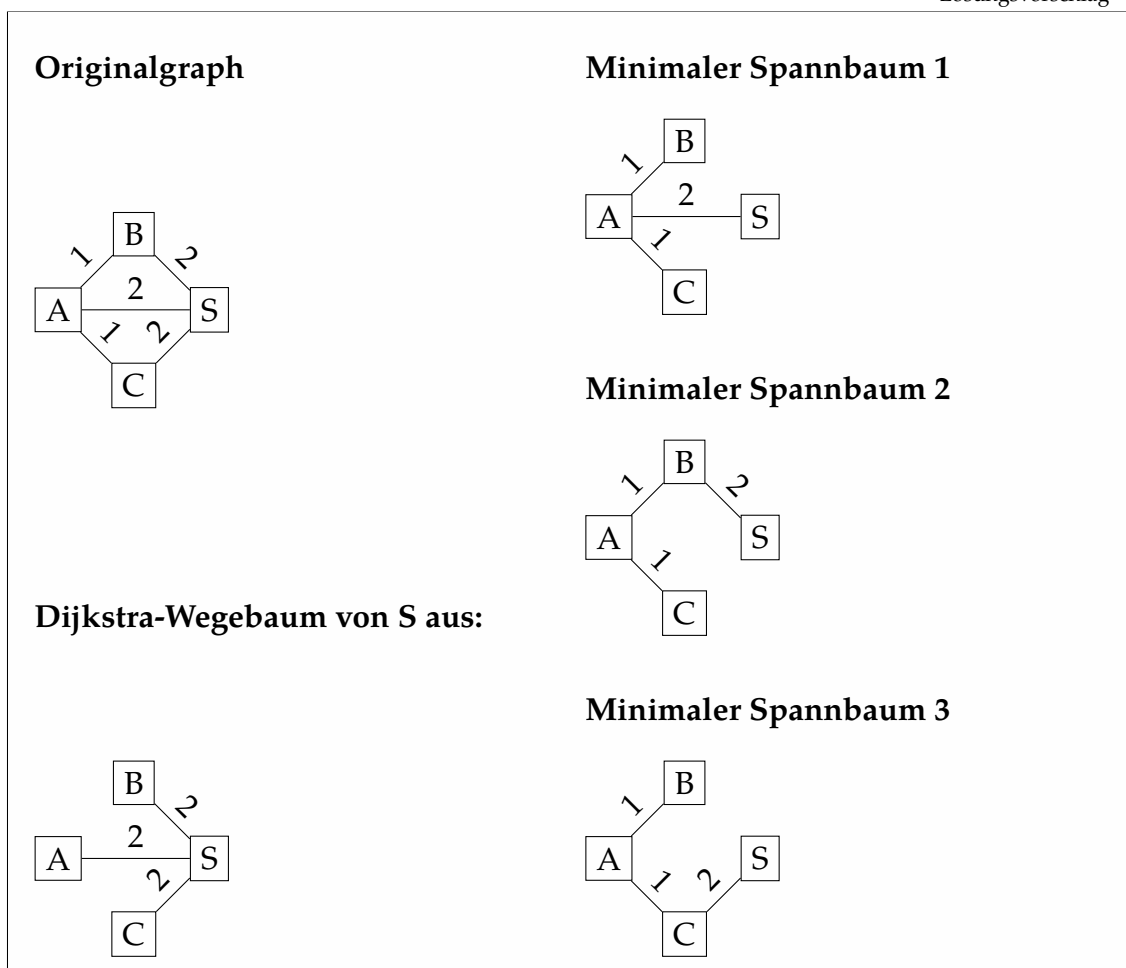
```

- (b) Welche asymptotische Laufzeit in Abhängigkeit von der Anzahl der Knoten ( $n$ ) und von der Anzahl der Kanten ( $m$ ) hat die Breitensuche über alle Neustarts zusammen? Beachten Sie, dass die Markierungen nicht gelöscht werden. Geben Sie die Laufzeit in  $\Theta$ -Notation an. Begründen Sie Ihre Antwort.

## 66115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 4

Die Algorithmen von Dijkstra und Jarník-Prim gehen ähnlich vor. Beide berechnen, ausgehend von einem Startknoten, einen Baum. Allerdings berechnet der Algorithmus von Dijkstra einen Kürzesten-Wege-Baum, während der Algorithmus von Jarník-Prim einen minimalen Spannbaum berechnet.

- (a) Geben Sie einen ungerichteten gewichteten Graphen  $G$  mit höchstens fünf Knoten und einen Startknoten  $s$  von  $G$  an, so dass **Dijkstra**( $G, s$ ) und **Jarník-Prim**( $G, s$ ) ausgehend von  $s$  verschiedene Bäume in  $G$  liefern. Geben Sie beide Bäume an.



- (b) Geben Sie eine unendlich große Menge von Graphen an, auf denen der Algorithmus von Jarník-Prim asymptotisch schneller ist als der Algorithmus von Kruskal, der ebenfalls minimale Spannbäume berechnet.

*Hinweis:* Für einen Graphen mit  $n$  Knoten und  $m$  Kanten benötigt Jarník-Prim  $\mathcal{O}(m + n \log n)$  Zeit, Kruskal  $\mathcal{O}(m \log m)$  Zeit.

- (c) Sei  $Z$  die Menge der zusammenhängenden Graphen und  $G \in Z$ . Sei  $n$  die Anzahl der Knoten von  $G$  und  $m$  die Anzahl der Kanten von  $G$ . Entscheiden Sie mit Begründung, ob  $\log m \in \Theta(\log n)$  gilt.

## 66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1

- (a) Sei

$$L_1 = \{ w \in \{a, b, c\}^* \mid w \text{ enthält genau zweimal den Buchstaben } a \text{ und der vorletzte Buchstabe ist } a \}$$

Geben Sie einen regulären Ausdruck für die Sprache  $L_1$  an.

```
(
 ((b|c)* a (b|c)* a (b|c)* c (b|c))
 |
)
```



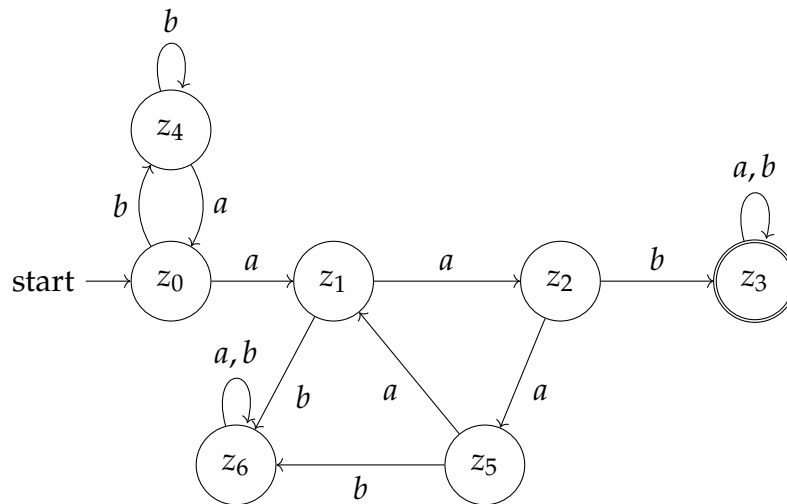
$$((b|c)^* a (b|c)^* c a)$$

 Kontextfreie Sprache  
 Pumping-Lemma  
 (Kontextfreie Sprache)

- (b) Konstruieren Sie einen deterministischen endlichen Automaten für die Sprache  $L_2$ :

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ enthält genau einmal das Teilwort } aab \}$$

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahf2oduri

- (c) Sei  $\mathbb{N} = \{1, 2, 3, \dots\}$  die Menge der strikt positiven natürlichen Zahlen. Sei  $L_3 = \{ \#a^{i_1}\#a^{i_1}\#\dots a^{i_{n-1}}\#a^{i_n}\# \mid n, i_1, \dots, i_n \in \mathbb{N} \text{ und es existiert } j \in \mathbb{N} \text{ mit } i_j = n + 1 \}$  eine Sprache über Alphabet  $\{\#, a\}$ .  
 So ist z. B.  $\#a\#aaa\# \in L_3$  (da das Teilwort  $a^3 = aaa$  vorkommt) und  $\#a\#a\#a\#a\# \in L_3$  (da das Teilwort  $a^5 = aaaaa$  nicht vorkommt). Beweisen Sie, dass  $L_3$  nicht regulär ist.

## 66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2

- (a) Zeigen Sie, dass die Sprache

$$L = \{ ww_1ww_2 \mid w, w_1, w_2 \in \{a, b, c\}^* \text{ und } 2|w| \geq |w_1| + |w_2| \}$$

nicht kontextfrei ist.

### Exkurs: Pumping-Lemma für Kontextfreie Sprachen

Es sei  $L$  eine kontextfreie Sprache. Dann gibt es eine Zahl  $j$ , sodass sich alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  zerlegen lassen in  $\omega = uvwxy$ , sodass die folgenden Eigenschaften erfüllt sind:

- (i)  $|vx| \geq 1$  (Die Wörter  $v$  und  $x$  sind nicht leer.)

- (ii)  $|vwx| \leq j$  (Die Wörter  $v$ ,  $w$  und  $x$  haben zusammen höchstens die Länge  $j$ .)
- (iii) Für alle  $i \in \mathbb{N}_0$  gilt  $uv^iwx^iy \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^iwx^iy$  in der Sprache  $L$ )

Lösungsvorschlag

Es gibt eine Pumpzahl. Sie sei  $j$ .  $a^jb^ja^jc^j$  ist ein Wort aus  $L$ , das sicher länger als  $j$  ist. Außerdem gilt  $2|a^j| \geq |b^j| + |c^j|$ . Unser gewähltes Wort ist deshalb in  $L$ . Da  $|vwx| \leq j$  und  $|xv| \geq 1$  sein muss, liegt  $vwx$  entweder in  $w$ ,  $w_1$  oder  $w_2$ .

**Aufteilung:  $vwx$  in  $w$  (erstes  $w$ ):**

**u** :  $\varepsilon$

**v** :  $a$

**w** :  $a^{j-2}$

**x** :  $a$

**y** :  $b^ja^jc^j$

Es gilt  $uv^iwx^iy \notin L$  für alle  $i \in \mathbb{N}_0$ , da  $a^jb^ja^jc^j \notin L$  für  $i = 0$ , da  $|a^{j-2}| + |a^j| < |b^j| + |c^j|$

**Aufteilung:  $vwx$  in  $w$  (zweites  $w$ ):**

**u** :  $a^jb^j$

**v** :  $a$

**w** :  $a^{j-2}$

**x** :  $a$

**y** :  $c^j$

Es gilt  $uv^iwx^iy \notin L$  für alle  $i \in \mathbb{N}_0$ , da  $a^jb^ja^jc^j \notin L$  für  $i = 0$ , da  $|a^j| + |a^{j-2}| < |b^j| + |c^j|$

**Aufteilung:  $vwx$  in  $w_1$ :**

**u** :  $a^j$

**v** :  $b$

**w** :  $b^{j-2}$

**x** :  $b$

**y** :  $a^jc^j$

Es gilt nicht  $uv^iwx^i y \in L$  für alle  $i \in \mathbb{N}_0$ , da  $a^i b^i a^i c^i \notin L$  für alle  $i > 2$  da  $2|a^i| < |b^{i-2+2i}| + |c^i|$  für alle  $i > 2$

Entscheidbarkeit

**Aufteilung:**  $vwx$  in  $w_2$ :

Analog zur Aufteilung  $vwx$  in  $w_1$

$\Rightarrow L$  ist nicht kontextfrei.

(b) Betrachten Sie die Aussage

Seien  $L_1, \dots, L_n$  beliebige kontextfreie Sprachen.  
Dann ist  $\bigcap_{i=1}^n L_i$  immer eine entscheidbare Sprache.

Entscheiden Sie, ob diese Aussage wahr ist oder nicht und begründen Sie Ihre Antwort.

Lösungsvorschlag

Diese Aussage ist falsch.

Kontextfreie Sprachen sind nicht abgeschlossen unter dem Schnitt, die Schnittmenge zweier kontextfreier Sprachen kann in einer Sprache eines anderen Typs in der Chomsky Sprachen-Hierarchie resultieren. Entsteht durch den Schnitt eine Typ-0-Sprache, dann ist diese nicht entscheidbar.

(c) Sei  $\mathbb{N}_0 = \{0, 1, 2, \dots\}$  die Menge der nicht negativen natürlichen Zahlen. Es ist bekannt, dass  $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$  keine kontextfreie Sprache ist. Ist die Komplementsprache  $L_5 = \{a, b, c\}^* \setminus L$  kontextfrei? Begründen Sie Ihre Antwort.

## 66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 3

Wir betrachten eine Gödelisierung von Turingmaschinen und bezeichnen mit  $M_w$  die Turingmaschine, die gemäß der Kodierung des Binärworts  $w$  kodiert wird. Außerdem bezeichnen wir mit  $M_w(x)$  die Ausgabe der Maschine  $M_w$  bei Eingabe  $x$ . Sie dürfen davon ausgehen, dass  $x$  immer ein Binärstring ist. Der bekannte Satz von Rice sagt:

Sei  $S$  eine Menge berechenbarer Funktionen mit  $\emptyset \neq S \neq \mathcal{R}$ , wobei  $\mathcal{R}$  die Menge aller berechenbaren Funktionen ist. Dann ist die Sprache  $L = \{w \mid f_{M_w} \in S\}$  unentscheidbar.

Hier ist  $f_{M_w}$  die von  $M_w$  berechnete Funktion.

Zeigen Sie für jede der nachfolgenden Sprachen über dem Alphabet  $\{0, 1\}$  entweder, dass sie entscheidbar ist, oder zeigen Sie mit Hilfe des Satzes von Rice, dass sie unentscheidbar ist. Geben Sie beim Beweis der Unentscheidbarkeit die Menge  $S$  der berechenbaren Funktionen an, auf die Sie den Satz von Rice anwenden. Wir bezeichnen die Länge der Eingabe  $x$  mit  $|x|$ .

(a)  $L = \{w \mid M_w \text{ akzeptiert die Binarkodierungen der Primzahlen (und lehnt alles andere ab)}\}$

(b)  $L = \{w \mid \text{es gibt eine Eingabe } x, \text{ so dass } M_w(x) \text{ das Symbol 1 enthält}\}$

- (c)  $L = \{ w \mid M_w(x) \text{ hält für jedes } x \text{ mit } |x| < 1000 \text{ nach höchstens 100 Schritten an} \}$
- (d)  $L = \{ w \mid M_w \text{ hat für jede Eingabe dieselbe Ausgabe} \}$
- (e)  $L = \{ w \mid \text{die Menge der Eingaben, die von } M_w \text{ akzeptiert werden, ist endlich} \}$

## 66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 4

Betrachten Sie die folgenden Probleme:

### CLIQUE

**Gegeben:** Ein ungerichteter Graph  $G = (V, E)$ ,  
eine Zahl  $k \in \mathcal{N}$

**Frage:** Gibt es eine Menge  $S \subseteq V$  mit  $|S| = k$ ,  
sodass für alle Knoten  $u \neq v \in V$  gilt,  
dass  $\{u, v\}$  eine Kante in  $E$  ist?

### ALMOST CLIQUE

**Gegeben:** Ein ungerichteter Graph  $G = (V, E)$ ,  
eine Zahl  $k \in \mathcal{N}$

**Frage:** Gibt es eine Menge  $S \subseteq V$  mit  $|S| = k$ ,  
sodass die Anzahl der Kanten zwischen Knoten in  $S$  genau  $\frac{k(k-1)}{2} - 1$  ist?

Zeigen Sie, dass das Problem ALMOST CLIQUE NP-vollständig ist. Nutzen Sie dafür die NP-Vollständigkeit von CLIQUE.

Hinweis: Die Anzahl der Kanten einer  $k$ -Clique sind  $\frac{k(k-1)}{2}$ .

#### Exkurs: Cliquesproblem

Das **Cliquesproblem** fragt nach der Existenz einer Clique der Mindestgröße  $n$  in einem gegebenen Graphen. Eine Clique ist eine Teilmenge von Knoten in einem ungerichteten Graphen, bei der *jedes Knotenpaar durch eine Kante verbunden ist*.

#### Exkurs: Almost Clique

Eine Gruppe von Knoten wird ALMOST CLIQUE genannt, wenn nur eine Kante ergänzt werden muss, damit sie zu einer Clique wird.

Lösungsvorschlag

You can reduce to this from CLIQUE.

Given a graph  $G = (V, E)$  and  $t$ , construct a new graph  $G^*$  by adding two new vertices  $\{v_{n+1}, v_{n+2}\}$  and connecting them with all of  $G$ 's vertices but removing the edge  $\{v_{n+1}, v_{n+2}\}$ , i.e. they are not neighbors in  $G^*$ . return  $G^*$  and  $t + 2$ .

If  $G$  has a  $t$  sized clique by adding it to the two vertices we get an  $t + 2$  almost

clique in  $G^*$  (by adding  $\{v_{n+1}, v_{n+2}\}$ ).

If  $G^*$  has a  $t + 2$  almost clique we can look at three cases:

1) It contains the two vertices  $\{v_{n+1}, v_{n+2}\}$ , then the missing edge must be  $\{v_{n+1}, v_{n+2}\}$  and this implies that the other  $t$  vertices form a  $t$  clique in  $G$ .

2) It contains one of the vertices  $\{v_{n+1}, v_{n+2}\}$ , say w.l.o.g.  $v_{n+1}$ , then the missing edge must be inside  $G$ , say  $e = \{u, v\} \in G$ . If we remove  $u$  and  $v_{n+1}$  then the other  $t$  vertices, which are in  $G$  must form a clique of size  $t$ .

3) It does not contain any of the vertices  $\{v_{n+1}, v_{n+2}\}$ , then it is clear that this group is in  $G$  and must contain a clique of size  $t$ .

It is also clear that the reduction is in polynomial time, actually in linear time, log-space.<sup>a</sup>

<sup>a</sup><https://cs.stackexchange.com/a/76627>

## 66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Gegeben sei die folgende Java-Implementierung einer doppelt-verketteten Liste.

```
class DoubleLinkedList {
 private Item head;

 public DoubleLinkedList() {
 head = null;
 }

 public Item append(Object val) {
 if (head == null) {
 head = new Item(val, null, null);
 head.prev = head;
 head.next = head;
 } else {
 Item item = new Item(val, head.prev, head);
 head.prev.next = item;
 head.prev = item;
 }
 return head.prev;
 }

 public Item search(Object val) {
 // ...
 }

 public void delete(Object val) {
 // ...
 }
}

class Item {
 private Object val;
 private Item prev;
 private Item next;

 public Item(Object val, Item prev, Item next) {
 this.val = val;
 }
}
```

```

 this.prev = prev;
 this.next = next;
 }
}
}

```

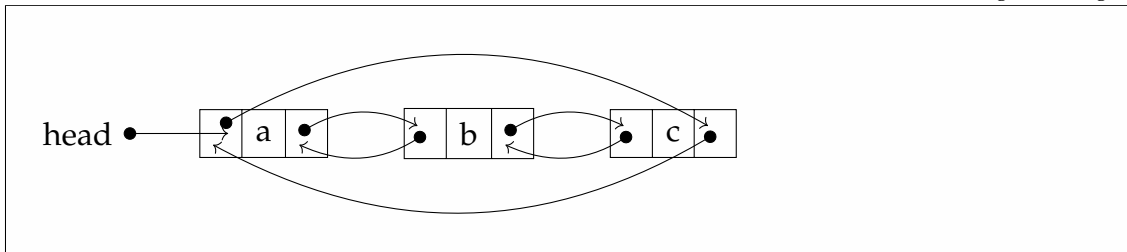
- (a) Skizzieren Sie den Zustand der Datenstruktur nach Aufruf der folgenden Befehlssequenz. Um Variablen mit Zeigern auf Objekte darzustellen, können Sie mit dem Variablennamen beschriftete Pfeile verwenden.

```

DoubleLinkedList list = new DoubleLinkedList();
list.append("a");
list.append("b");
list.append("c");

```

Lösungsvorschlag



- (b) Implementieren Sie in der Klasse `DoubleLinkedList` die Methode `search`, die zu einem gegebenen Wert das Item der Liste mit dem entsprechenden Wert, oder `null` falls der Wert nicht in der Liste enthalten ist, zurückgibt.

Lösungsvorschlag

```

public Item search(Object val) {
 Item item = null;
 if (head != null) {
 item = head;
 do {
 if (item.val.equals(val)) {
 return item;
 }
 item = item.next;
 } while (!item.equals(head));
 }
 return null;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2021/fruehjahr/DoubleLinkedList.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/DoubleLinkedList.java)

- (c) Implementieren Sie in der Klasse `DoubleLinkedList` die Methode `delete`, die das erste Vorkommen eines Wertes aus der Liste entfernt. Ist der Wert nicht in der Liste enthalten, terminiert die Methode „stillschweigend“, ohne Änderung der Liste und ohne Fehlermeldung. Sie dürfen die Methode `search` aus Teilaufgabe b) verwenden, auch wenn Sie sie nicht implementiert haben.

```

public void delete(Object val) {
 Item item = search(val);
 if (item != null) {
 if (head.next.equals(head)) {
 head = null;
 } else {
 if (item.equals(head)) {
 head = item.next;
 }
 item.prev.next = item.next;
 item.next.prev = item.prev;
 }
 }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2021/fruehjahr/DoubleLinkedList.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/DoubleLinkedList.java)

- (d) Beschreiben Sie die notwendigen Änderungen an der Datenstruktur und an den bisherigen Implementierungen, um eine Methode `size`, die die Anzahl der enthaltenen Items zurück gibt, mit Laufzeit  $\mathcal{O}(1)$  zu realisieren.

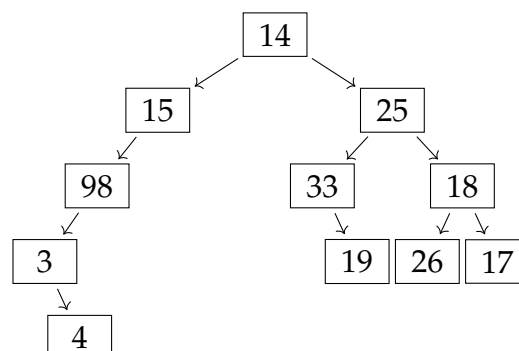
Lösungsvorschlag

In der Klasse wird ein Zähler eingefügt, der bei jedem Aufruf der Methode `append` um eins nach oben gezählt wird und bei jedem erfolgreichen Löschen in der `delete`-Methode um eins nach unten gezählt wird. Mit `return` kann der Zählerstand in  $\mathcal{O}(1)$  ausgegeben werden. Dazu müsste ein Getter zum Ausgeben implementiert werden. Die Datenstruktur bleibt unverändert.

## 66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 3

- (a) Betrachten Sie folgenden Binärbaum T.

Geben Sie die Schlüssel der Knoten in der Reihenfolge an, wie sie von einem Preorder-Durchlauf (= TreeWalk) von T ausgegeben werden.



**Exkurs: Preorder-Traversierung eines Baum**

besuche die Wurzel, dann den linken Unterbaum, dann den rechten Unterbaum; auch: WLR

```
private void besuchePreorder(BaumKnoten knoten, ArrayList<Comparable>
 ↪ schlüssel) {
 if (knoten != null) {
 schlüssel.add((Comparable) knoten.gibSchlüssel());
 besuchePreorder(knoten.gibLinks(), schlüssel);
 besuchePreorder(knoten.gibRechts(), schlüssel);
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/baum/BinaerBaum.java](https://github.com/org/bschlangaul/baum/BinaerBaum.java)

Lösungsvorschlag

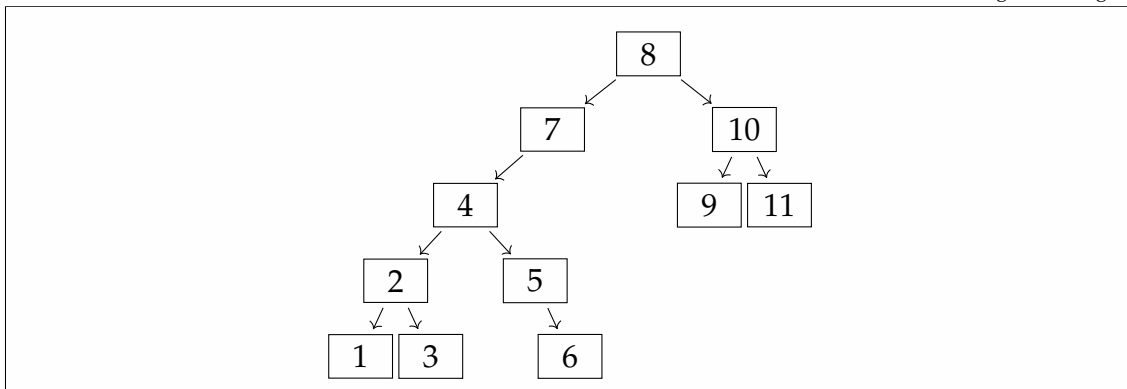
14, 15, 98, 3, 4, 25, 33, 19, 18, 26, 17

- (b) Betrachten Sie folgende Sequenz als Ergebnis eines Preorder-Durchlaufs eines binären Suchbaumes  $T$ . Zeichnen Sie  $T$  und erklären Sie, wie Sie zu Ihrer Schlussfolgerung gelangen.

[8,7,4,2,1,3,5,6,10,9,11]

**Hinweis:** Welcher Schlüssel ist die Wurzel von  $T$ ? Welche Knoten sind in seinem linken/rechten Teilbaum gespeichert? Welche Schlüssel sind die Wurzeln der jeweiligen Teilbäume?

Lösungsvorschlag



- (c) Anstelle von sortierten Zahlen soll ein Baum nun verwendet werden, um relative Positionsangaben zu speichern. Jeder Baumknoten enthält eine Beschriftung und einen Wert (vgl. Abb. 1), der die ganzzahlige relative Verschiebung in horizontaler Richtung gegenüber seinem Elternknoten angibt. Die zu berechnenden Koordinaten für einen Knoten ergeben sich aus seiner Tiefe im Baum als  $y$ -Wert und aus der Summe aller Verschiebungen auf dem Pfad zur Wurzel als  $x$ -Wert. Das Ergebnis der Berechnung ist in Abb. 2 visualisiert. Geben Sie einen Algorithmus



mit linearer Laufzeit in Pseudo-Code oder einer objektorientierten Programmiersprache Ihrer Wahl an. Der Algorithmus erhält den Zeiger auf die Wurzel eines Baumes als Eingabe und soll Tupel mit den berechneten Koordination aller Knoten des Baums in der Form (Beschriftung,  $x$ ,  $y$ ) zurück- oder ausgeben.

Lösungsvorschlag

```
public class Knoten {
 public Knoten links;
 public Knoten rechts;
 public String name;

 /**
 * Bewegung bezüglich des Vorknotens. Relative Lage.
 */
 public int xVerschiebung;

 public Knoten(String name, int xVerschiebung) {
 this.name = name;
 this.xVerschiebung = xVerschiebung;
 }

 public void durchlaufen() {
 durchlaufe(this, 0 + xVerschiebung, 0);
 }

 private void durchlaufe(Knoten knoten, int x, int y) {
 System.out.println("Beschriftung: " + knoten.name + " x: " + x + " y: " +
 ↪ y);

 if (links != null) {
 links.durchlaufe(links, x + links.xVerschiebung, y + 1);
 }
 if (rechts != null) {
 rechts.durchlaufe(rechts, x + rechts.xVerschiebung, y + 1);
 }
 }

 public static void main(String[] args) {
 Knoten a = new Knoten("a", 1);
 Knoten b = new Knoten("b", 1);
 Knoten c = new Knoten("c", -2);
 Knoten d = new Knoten("d", 2);
 Knoten e = new Knoten("e", 0);

 a.links = b;
 a.rechts = c;
 c.links = d;
 c.rechts = e;

 a.durchlaufen();
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2021/fruehjahr/Knoten.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/Knoten.java)

## 66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 5

- (a) Nennen Sie zwei wünschenswerte Eigenschaften von Hashfunktionen.

Lösungsvorschlag

**surjektiv** Die Abbildung soll surjektiv sein, ö jeder Index soll berechnet werden können.

**gleichverteilt** Durch die Hashfunktion soll möglichst eine Gleichverteilung auf die Buckets (Indexliste) erfolgen.

**effizient** Zudem sollte die Verteilung mittels Hashfunktion möglichst effizient gewählt werden.

- (b) Wie viele Elemente können bei Verkettung und wie viele Elemente können bei offener Adressierung in einer Hashtabelle mit
- $m$
- Zeilen gespeichert werden?

Lösungsvorschlag

**Verkettung** Es darf mehr als ein Element pro Bucket enthalten sein, deswegen können beliebig viele Element gespeichert werden.

**offene Adressierung** (normalerweise) ein Element pro Bucket, deshalb ist die Anzahl der speicherbaren Elemente höchstens  $m$ . Können in einem Bucket  $k$  Elemente gespeichert werden, dann beträgt die Anzahl der speicherbaren Elemente  $k \cdot m$ .

- (c) Angenommen, in einer Hashtabelle der Größe
- $m$
- sind alle Einträge (mit mindestens einem Wert) belegt und insgesamt
- $n$
- Werte abgespeichert.

Geben Sie in Abhängigkeit von  $m$  und  $n$  an, wie viele Elemente bei der Suche nach einem nicht enthaltenen Wert besucht werden müssen. Sie dürfen annehmen, dass jeder Wert mit gleicher Wahrscheinlichkeit und unabhängig von anderen Werten auf jeden der  $m$  Plätze abgebildet wird (einfaches gleichmäßiges Hashing).

Lösungsvorschlag

$\frac{n}{m}$  Beispiel: 10 Buckets, 30 Elemente:  $\frac{30}{10} = 3$  Elemente im Bucket, die man durchsuchen muss.

- (d) Betrachten Sie die folgende Hashtabelle mit der Hashfunktion
- $h(x) = x \bmod 11$
- . Hierbei steht
- $\emptyset$
- für eine Zelle, in der kein Wert hinterlegt ist.

| Index | 0  | 1           | 2           | 3 | 4           | 5  | 6  | 7  | 8           | 9           | 10 |
|-------|----|-------------|-------------|---|-------------|----|----|----|-------------|-------------|----|
| Wert  | 11 | $\emptyset$ | $\emptyset$ | 3 | $\emptyset$ | 16 | 28 | 18 | $\emptyset$ | $\emptyset$ | 32 |

Führen Sie nun die folgenden Operationen mit offener Adressierung mit linearem Sondieren aus und geben Sie den Zustand der Datenstruktur nach jedem Schritt an. Werden für eine Operation mehrere Zellen betrachtet, aber nicht modifiziert, so geben Sie deren Indizes in der betrachteten Reihenfolge an.

- (i) Insert 7

|       |    |   |   |   |   |    |    |    |                |   |    |
|-------|----|---|---|---|---|----|----|----|----------------|---|----|
| Index | 0  | 1 | 2 | 3 | 4 | 5  | 6  | 7  | 8              | 9 | 10 |
| Wert  | 11 | ∅ | ∅ | 3 | ∅ | 16 | 28 | 18 | 7 <sub>2</sub> | ∅ | 32 |

(ii) Insert 20

Lösungsvorschlag

|       |    |   |   |   |   |    |    |    |                |                 |    |
|-------|----|---|---|---|---|----|----|----|----------------|-----------------|----|
| Index | 0  | 1 | 2 | 3 | 4 | 5  | 6  | 7  | 8              | 9               | 10 |
| Wert  | 11 | ∅ | ∅ | 3 | ∅ | 16 | 28 | 18 | 7 <sub>2</sub> | 20 <sub>1</sub> | 32 |

(iii) Delete 18

Lösungsvorschlag

|       |    |   |   |   |   |    |    |     |                |                 |    |
|-------|----|---|---|---|---|----|----|-----|----------------|-----------------|----|
| Index | 0  | 1 | 2 | 3 | 4 | 5  | 6  | 7   | 8              | 9               | 10 |
| Wert  | 11 | ∅ | ∅ | 3 | ∅ | 16 | 28 | del | 7 <sub>2</sub> | 20 <sub>1</sub> | 32 |

del ist eine Marke, die anzeigt, dass gelöscht wurde und der Bucket nicht leer ist.

(iv) Search 7

Lösungsvorschlag

|       |    |   |   |   |   |    |    |     |                |                 |    |
|-------|----|---|---|---|---|----|----|-----|----------------|-----------------|----|
| Index | 0  | 1 | 2 | 3 | 4 | 5  | 6  | 7   | 8              | 9               | 10 |
| Wert  | 11 | ∅ | ∅ | 3 | ∅ | 16 | 28 | del | 7 <sub>2</sub> | 20 <sub>1</sub> | 32 |

$h(7) = 7 \bmod 11 = 7$   
 7 (Index) → del lineares sondieren → 8 (Index) → gefunden

(v) Insert 5

Lösungsvorschlag

|       |    |   |   |   |   |    |    |                |                |                 |    |
|-------|----|---|---|---|---|----|----|----------------|----------------|-----------------|----|
| Index | 0  | 1 | 2 | 3 | 4 | 5  | 6  | 7              | 8              | 9               | 10 |
| Wert  | 11 | ∅ | ∅ | 3 | ∅ | 16 | 28 | 5 <sub>3</sub> | 7 <sub>2</sub> | 20 <sub>1</sub> | 32 |

## 66116 (Datenbanksysteme / Softwaretechnologie (vertieft))

### 66116 / 2012 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1

Ein Handelsunternehmen möchte seine Struktur verbessern und ein Datenbanksystem zur Verwaltung seiner Filialen, angebotenen Waren und Kunden erstellen.

Die Basis dieses Systems bilden die Filialen des Unternehmens. Jede Filiale ist eindeutig durch ihre Filialnummer gekennzeichnet und befindet sich in einer Stadt. Außerdem hat jede Filiale einen Filialleiter.

Zu jeder Filiale gehört genau ein Lager mit einer eindeutigen Lagernummer und ebenfalls einem Leiter. Jedes Lager verfügt über eine bestimmte Menge an verschiedenen Waren. Jede Ware kann in mehreren Lagern vorrätig sein und ist über eine Nummer, einen Namen und einen Preis gekennzeichnet.

Ein Kunde kann in einer Filiale des Unternehmens Bestellungen aufgeben. Der Kunde hat eine Kundennummer, einen Namen und eine Adresse. Eine Bestellung enthält dabei jeweils einen Warenartikel, dessen gewünschte Menge und das Datum, an dem die Bestellung abgeholt wird.

- (a) Erstellen Sie ein Entity-Relationship-Diagramm für obige Datenbank.
- (b) Setzen Sie das in Teilaufgabe a) erstellte Entity-Relationship-Diagramm in ein Relationenschema um. Relationships sollen mit einer möglichst geringen Anzahl von Relationen realisiert werden. Dabei sind unnötige Redundanzen zu vermeiden. Ein Relationenschema ist in folgender Form anzugeben: Relation (Attribut1, Attribut2, ...). Schlüsselattribute sind dabei zu unterstreichen. Achten Sie bei der Wahl des Schlüssels auf Eindeutigkeit und Minimalität.

## 66116 / 2012 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Gegeben sei folgende Datenbank für Wareneingänge eines Warenlagers. Die Primärschlüssel-Attribute sind unterstrichen.

| <u>ZulieferungsNr</u> | <u>ArtikelNr</u> | Datum      | Artikelname | Menge |
|-----------------------|------------------|------------|-------------|-------|
| 1                     | 1                | 01.01.2009 | Handschuhe  | 5     |
| 1                     | 2                | 01.01.2009 | Mütze       | 10    |
| 2                     | 3                | 05.01.2009 | Schal       | 2     |
| 2                     | 1                | 05.01.2009 | Handschuhe  | 18    |
| 3                     | 4                | 06.01.2009 | Jacke       | 2     |

- (a) Erläutern Sie, inwiefern obiges Schema die 3. Normalform verletzt.

Lösungsvorschlag

- (b) Geben Sie für obige Datenbank alle vollen funktionalen Abhängigkeiten (einschließlich der transitiven) an.

Lösungsvorschlag

Lösungsvorschlag

### Exkurs: Voll funktionale Abhängigkeit

Eine vollständig funktionale Abhängigkeit liegt dann vor, wenn dass Nicht-Schlüsselattribut nicht nur von einem Teil der Attribute eines zusammengesetzten Schlüsselkandidaten funktional abhängig ist, sondern von allen Teilen eines Relationstyps. Die vollständig funktionale Abhängigkeit wird mit der 2. Normalform (2NF) erreicht.<sup>a</sup>

<sup>a</sup>datenbank-verstehen.de

**Exkurs: Transitive Abhängigkeit**

Eine transitive Abhängigkeit liegt dann vor, wenn Y von X funktional abhängig und Z von Y, so ist Z von X funktional abhängig. Diese Abhängigkeit ist transitiv. Die transitive Abhängigkeit wird mit 3. Normalform (3NF) erreicht.<sup>a</sup>

<sup>a</sup>datenbank-verstehen.de

$$FA = \left\{ \begin{array}{l} \{ \text{ZulieferungsNr} \} \rightarrow \{ \text{Datum} \}, \\ \{ \text{ArtikelNr} \} \rightarrow \{ \text{Artikelname} \}, \\ \{ \text{ZulieferungsNr}, \text{ArtikelNr} \} \rightarrow \{ \text{Menge} \}, \end{array} \right\}$$

- (c) Überführen Sie das obige Relationenschema in die 3. Normalform. Erläutern Sie die dazu durchzuführenden Schritte jeweils kurz.

**66116 / 2012 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3**

Gegeben sei das folgende Relationenschema:

Fahrzeug : {[ MNR[Modell], FZGNR, Baujahr, KMStand, Preis ]}

Modell : {[ MNR, HNR[Hersteller], Typ, Neupreis, ps ]}

Hersteller : {[ HNR, Name ]}

Dabei sind die Schlüsselattribute jeweils unterstrichen und zusätzlich für alle Attribute die Typen angegeben. Formulieren Sie die folgenden Anfragen bzw. Anweisungen in SQL.

- (a) Geben Sie die Anweisungen in SQL-DDL an, die notwendig sind, um die Relationen „Fahrzeug“, „Modell“ und „Hersteller“ zu erzeugen. Achten Sie dabei darauf, die Primärschlüssel der Relationen zu kennzeichnen.

```
CREATE TABLE IF NOT EXISTS Hersteller (
 HNR INTEGER PRIMARY KEY,
 Name CHAR(20)
);

CREATE TABLE IF NOT EXISTS Modell (
 MNR INTEGER PRIMARY KEY,
 HNR INTEGER REFERENCES Hersteller(HNR),
 Typ CHAR(20),
 Neupreis INTEGER,
```

```

 ps INTEGER
);

CREATE TABLE IF NOT EXISTS Fahrzeug (
 MNR INTEGER REFERENCES Modell(MNR),
 FZGNR CHAR(12) PRIMARY KEY,
 Baujahr INTEGER,
 KMStand INTEGER,
 Preis INTEGER
);

```

- (b) Bestimmen Sie die Typen aller Modelle des Herstellers mit Namen BMW.

Lösungsvorschlag

```

SELECT m.Typ
FROM Modell m, Hersteller h
WHERE h.HNR = m.HNR AND h.Name = 'BMW'
GROUP BY m.Typ;

```

- (c) Bestimmen Sie den Mindestpreis, bezogen auf das Attribut „Preis“, der Fahrzeuge eines jeden Herstellers.

Lösungsvorschlag

```

SELECT h1.Name AS Hersteller, (
 SELECT MIN(f.Preis)
 FROM Fahrzeug f, Modell m, Hersteller h2
 WHERE
 f.MNR = m.MNR AND
 m.HNR = h2.HNR AND
 h2.HNR = h1.HNR
) AS Mindestpreis
FROM Hersteller h1;

```

- (d) Bestimmen Sie die Namen der Hersteller, für die von jedem ihrer Modelle mindestens ein Fahrzeug in der Datenbank gespeichert ist.

Lösungsvorschlag

```

SELECT h.Name AS Hersteller
FROM Fahrzeug f, Modell m, Hersteller h
WHERE
 f.MNR = m.MNR AND
 m.HNR = h.HNR
GROUP BY h.Name;

```

- (e) Bestimmen Sie die Namen aller Hersteller, von denen mindestens fünf Fahrzeuge eines beliebigen Modells in der Datenbank gespeichert sind.

## 66116 / 2012 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Ein Getränkeliesservice verwaltet die Bestellungen verschiedener Kunden. Die folgenden Teilaufgaben sind in einer objektorientierten Programmiersprache zu lösen (die verwendete Sprache ist vorab anzugeben.).

- (a) Implementieren Sie eine Klasse `Kasten` zur Beschreibung eines Getränkekastens mit den folgenden Eigenschaften. Entscheiden Sie dabei jeweils ob eine Realisierung als Objekt- oder Klassenfeld sinnvoll ist.

- Es existiert ein einheitliches Kastenpfand in Höhe von 1,50 Euro.
- Für alle Flaschen in einem Kasten gelte ein einheitliches Flaschenpfand, das jedoch von Kasten zu Kasten verschieden sein kann.
- Während das Flaschenpfand für alle Flaschen eines Kastens gleich ist, sind die Einzelpreise der Flaschen je nach Inhalt unterschiedlich. Die Einzelpreise (ohne Flaschenpfand) der im Kasten enthaltenen Flaschen sollen in einem 2-dimensionalen Array abgelegt werden.

Geben Sie für die Klasse `Kasten` einen geeigneten Konstruktor an. Ergänzen Sie in der Klasse `Kasten` eine Objektmethode zur Berechnung des Gesamtpreises des Getränkekastens inklusive Kasten- und Flaschenpfand.

- (b) Schreiben Sie eine Klasse `Bestellung`. Jeder Bestellung soll eine eindeutige Bestellnummer zugeordnet werden, die über den Konstruktoraufbau erstellt wird. Außerdem soll zu jeder Bestellung der Name des Kunden gespeichert werden, sowie eine einfach verkettete Liste der bestellten Getränkekästen. Die Klasse `Bestellung` soll weiterhin eine Methode beinhalten, die den Gesamtpreis der Bestellung ermittelt.
- (c) Schreiben Sie ein kleines Testprogramm, das eine Bestellung erstellt, die zwei Getränkekästen umfasst. Der erste Kasten soll ein 1 x 1 Getränkekasten mit einer Flasche zu 0,75 Euro sein, der zweite Kasten soll - wie in Abbildung 1 dargestellt - ein 3 x 3 Getränkekasten mit 3 Flaschen zu 0,7 Euro auf der Diagonalen und 3 weiteren Flaschen zu je 1 Euro sein. Das Flaschenpfand beider Kästen beträgt 0,15 Euro pro Flasche, das Kastenpfand 1,50 Euro. Anschließend soll der Preis der Bestellung berechnet und auf der Standardausgabe ausgegeben werden.

|     |     |     |
|-----|-----|-----|
| 1,0 | 1,0 | 0,7 |
| 1,0 | 0,7 | 0   |
| 0,7 | 0   | 0   |

Lösungsvorschlag

```
/**
 * „Implementieren Sie eine Klasse Kasten zur Beschreibung eines
 * → Getränkekastens
 * mit den folgenden Eigenschaften. Entscheiden Sie dabei jeweils ob eine
 * Realisierung als Objekt- oder Klassenfeld sinnvoll ist.“
 */
public class Kasten {
 /**
 * Wir verwenden static, also ein sogenanntes Klassenfeld, da das
 * → Kastenpfand
 */
}
```



```

 * für alle Kästen gleich ist: „Es existiert ein einheitliches Kastenpfand
→ in
 * Höhe von 1,50 Euro."
 */
 static double kastenPfad = 1.5;

 /**
 * Wir verwenden ein Objektfeld, d.h. ein nicht statisches Feld: „Für alle
 * Flaschen in einem Kasten gelte ein einheitliches Flaschenpfand, das
→ jedoch
 * von Kasten zu Kasten verschieden sein kann."
 */
 double flaschenPfad;

 /**
 * „Während das Flaschenpfand für alle Flaschen eines Kastens gleich ist,
→ sind
 * die Einzelpreise der Flaschen je nach Inhalt unterschiedlich. Die
 * Einzelpreise (ohne Flaschenpfand) der im Kasten enthaltenen Flaschen
→ sollen
 * in einem 2-dimensionalen Array abgelegt werden."
 */
 double[][] flaschen;

 /**
 * „sowie eine einfach verkettete Liste der bestellten Getränkekästen. "
 */
 Kasten nächsterKasten = null;

 /**
 * „Geben Sie für die Klasse Kasten einen geeigneten Konstruktor an."
 *
 * @param flaschen Die Belegung des Kastens mit Flaschen als
 * zweidimensionales Feld der Flaschenpreise ohne
 * Flaschenpfand.
 * @param flaschenPfad Die Höhe des Flaschenpfads, dass für alle Flaschen
→ in
 * diesem Kasten gleich ist.
 */
 public Kasten(double[][] flaschen, double flaschenPfad) {
 this.flaschen = flaschen;
 this.flaschenPfad = flaschenPfad;
 }

 /**
 * „Ergänzen Sie in der Klasse Kasten eine Objektmethode zur Berechnung des
 * Gesamtpreises des Getränkekastens inklusive Kasten- und Flaschenpfand."
 *
 * @return Der Gesamtpreis des Getränkekastens inklusive Kasten- und
 * Flaschenpfand.
 */
 double berechneGesamtPreis() {
 double gesamtPreis = kastenPfad;
 for (int i = 0; i < flaschen.length; i++) {

```

```
double[] reihe = flaschen[i];
for (int j = 0; j < reihe.length; j++) {
 double flaschenPreis = flaschen[i][j];
 // Nur im Kasten vorhandene Flaschen kosten auch Flaschenpfand.
 if (flaschenPreis > 0)
 gesamtPreis += flaschenPfad + flaschen[i][j];
}
}
return gesamtPreis;
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2012/herbst/getraenke/Kasten.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2012/herbst/getraenke/Kasten.java)

```
import java.text.DecimalFormat;

/**
 * „Schreiben Sie eine Klasse Bestellung“
 */
public class Bestellung {
 /**
 * „Jeder Bestellung soll eine eindeutige Bestellnummer zugeordnet werden,
 * → die
 * über den Konstruktoraufruf erstellt wird.“
 */
 int bestellNummer;

 /**
 * „Außerdem soll zu jeder Bestellung der Name des Kunden gespeichert
 * → werden.“
 */
 String kundenName;

 /**
 * „sowie eine einfach verkettete Liste der bestellten Getränkekästen.“
 */
 Kasten kästen = null;

 /**
 * „Jeder Bestellung soll eine eindeutige Bestellnummer zugeordnet werden,
 * → die
 * über den Konstruktoraufruf erstellt wird. Außerdem soll zu jeder
 * → Bestellung
 * der Name des Kunden gespeichert werden.“
 *
 * @param bestellNummer Die Nummer der Getränkebestellung.
 * @param kundenName Der Name des/der KundenIn.
 */
 public Bestellung(int bestellNummer, String kundenName) {
 this.bestellNummer = bestellNummer;
 this.kundenName = kundenName;
 }

 /**
```

```
* „Die Klasse Bestellung soll weiterhin eine Methode beinhalten, die den
* Gesamtpreis der Bestellung ermittelt.“
*
* @return Der Gesamtpreis der Getränkebestellung.
*/
double berechneGesamtPreis() {
 double gesamtPreis = 0;
 Kasten kasten = kästen;
 while (kasten != null) {
 gesamtPreis += kasten.berechneGesamtPreis();
 kasten = kasten.nächsterKasten;
 }
 return gesamtPreis;
}

/**
 * Nicht verlangt. Könnte auch in die Test-Methode geschrieben werden.
 *
 * @param flaschen Die Belegung des Kasten mit Flaschen als
 * zweidimensionales Feld der Flaschenpreise ohne
 * Flaschenpfad.
 * @param flaschenPfad Die Höhe des Flaschenpfads, dass für alle Flaschen
→ in
 * diesem Kasten gleich ist.
 */
void bestelleKasten(double[][] flaschen, double flaschenPfad) {
 Kasten bestellterKasten = new Kasten(flaschen, flaschenPfad);
 if (kästen == null) {
 kästen = bestellterKasten;
 return;
 }
 Kasten kasten = kästen;

 Kasten letzterKasten = null;
 while (kasten != null) {
 letzterKasten = kasten;
 kasten = kasten.nächsterKasten;
 }
 letzterKasten.nächsterKasten = bestellterKasten;
}

/**
 * Kleines Schmankerl. Nicht verlangt. Damit wir nicht 9.899999999999999
→ als
 * Aufgabe bekommen.
 *
 * @param preis Ein Preis als Gleitkommazahl.
 *
 * @return Der Preis als Text mit zwei Stellen nach dem Komma.
 */
static String runde(double preis) {
 DecimalFormat df = new DecimalFormat("#.##");
 df.setMinimumFractionDigits(2);
 return df.format(preis);
}
```

```
}

/**
 * Die main-Methode soll hier als Testmethode verwendet werden:
 *
 * „Schreiben Sie ein kleines Testprogramm, das eine Bestellung erstellt,
→ die
 * zwei Getränkekästen umfasst. Der erste Kasten soll ein 1 x 1
→ Getränkekasten
 * mit einer Flasche zu 0,75 Euro sein, der zweite Kasten soll - wie in
 * Abbildung 1 dargestellt - ein 3 x 3 Getränkekasten mit 3 Flaschen zu 0,7
→ Euro
 * auf der Diagonalen und 3 weiteren Flaschen zu je 1 Euro sein. Das
 * Flaschenpfand beider Kästen beträgt 0,15 Euro pro Flasche, das
→ Kastenpfand
 * 1,50 Euro. Anschließend soll der Preis der Bestellung berechnet und auf
→ der
 * Standardausgabe ausgegeben werden."
 *
 * @param args Kommandozeilenargumente, die uns nicht zu interessieren
→ brauchen.
 */
public static void main(String[] args) {
 Bestellung bestellung = new Bestellung(1, "Hermine Bschlangaul");

 // Müsste eigentlich nicht mehr gesetzt werden, da wir es schon in der
 // Klassendefinition gesetzt haben.
 Kasten.kastenPfad = 1.50;

 bestellung.bestelleKasten(new double[][] { { 0.75 } }, 0.15);
 bestellung.bestelleKasten(new double[][] { { 1.0, 1.0, 0.7 }, { 1.0, 0.7,
→ 0 }, { 0.7, 0, 0 } }, 0.15);

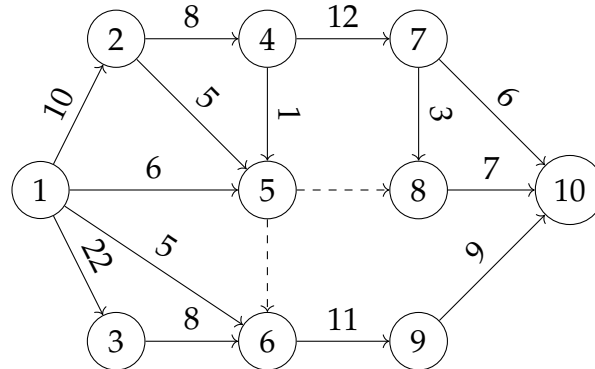
 // Oder kürzer
 // bestellung.bestelleKasten(new double[][] { { 1, 1, .7 }, { 1, .7 }, {
→ .7 } }, .15);

 // Gegenrechnung:
 // 1 x 0.75 = 0.75
 // 3 x 1.00 = 3.00
 // 3 x 0.70 = 2.10
 // 7 x 0.15 = 1.05 (Flaschenpfad)
 // 3 x 1.50 = 3.00 (Kastenpfand)
 // ----
 // 9.90
 System.out.println("Der Gesamtpreis der Getränkebestellung beträgt: " +
→ runde(bestellung.berechneGesamtPreis()) + " €");
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2012/herbst/getraenke/Bestellung.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2012/herbst/getraenke/Bestellung.java)

## 66116 / 2012 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Die unten stehende Abbildung stellt ein CPM-Netzwerk dar. Die Ereignisse sind fortlaufend nummeriert (Nummer im Inneren der Kreise) und tragen keine Namen. Gestrichelte Linien stellen Pseudo-Aktivitäten mit einer Dauer von 0 dar.



- (a) Berechnen Sie die früheste Zeit für jedes Ereignis, wobei angenommen wird, dass das Projekt zum Zeitpunkt 0 startet!

Lösungsvorschlag

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $FZ_i$ : Frühester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann.

| $i$ | Nebenrechnung            | $FZ_i$ |
|-----|--------------------------|--------|
| 1   |                          | 0      |
| 2   |                          | 10     |
| 3   |                          | 22     |
| 4   |                          | 18     |
| 5   | $\max(15_2, 6_1, 19_4)$  | 19     |
| 6   | $\max(5_1, 30_6, 19_5)$  | 30     |
| 7   |                          | 30     |
| 8   | $\max(33_7, 19_5)$       | 33     |
| 9   |                          | 41     |
| 10  | $\max(36_7, 40_8, 50_9)$ | 50     |

- (b) Setzen Sie anschließend beim letzten Ereignis die späteste Zeit gleich der frühesten Zeit und berechnen Sie die spätesten Zeiten!

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $SZ_i$ : Spätester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann. \_\_\_\_\_

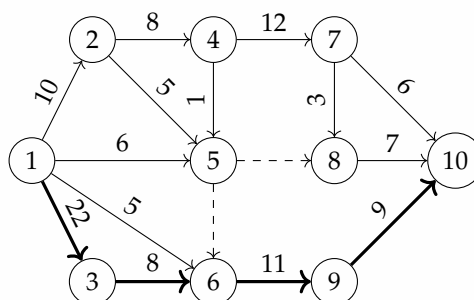
| $i$ | Nebenrechnung                 | $SZ_i$ |
|-----|-------------------------------|--------|
| 10  | siehe $FZ_{10}$               | 50     |
| 9   |                               | 41     |
| 8   |                               | 43     |
| 7   | $\min(44_{10}, 40_8)$         | 40     |
| 6   |                               | 30     |
| 5   | $\min(30_6, 43_8)$            | 30     |
| 4   | $\min(29_5, 28_7)$            | 28     |
| 3   |                               | 22     |
| 2   | $\min(20_4, 25_5)$            | 20     |
| 1   | $\min(10_2, 24_5, 0_3, 25_6)$ | 0      |

(c) Berechnen Sie nun für jedes Ereignis die Pufferzeiten!

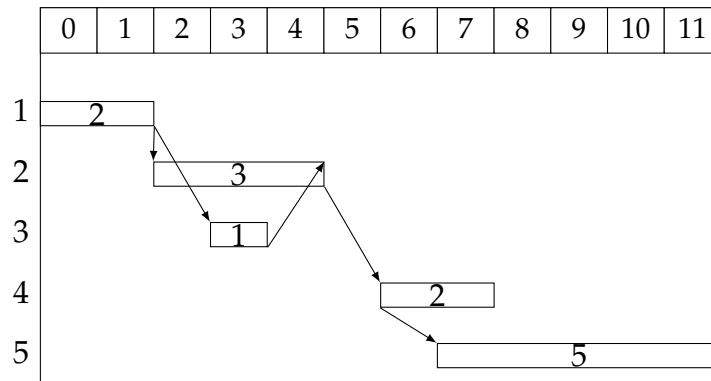
| $i$    | 1 | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
|--------|---|----|----|----|----|----|----|----|----|----|
| $FZ_i$ | 0 | 10 | 22 | 18 | 19 | 30 | 30 | 33 | 41 | 50 |
| $SZ_i$ | 0 | 20 | 22 | 28 | 30 | 30 | 40 | 43 | 41 | 50 |
| GP     | 0 | 10 | 0  | 10 | 11 | 0  | 10 | 10 | 0  | 0  |

(d) Bestimmen Sie den kritischen Pfad!

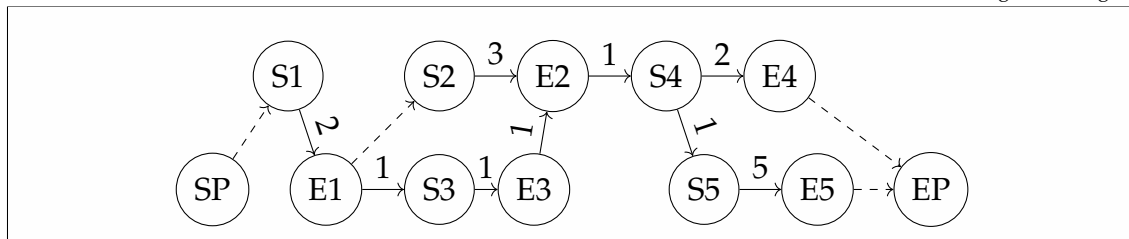
$1 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 10$



(e) Konvertieren Sie das Gantt-Diagramm aus Abbildung 3 in ein CPM-Netzwerk!



Lösungsvorschlag



## 66116 / 2014 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Sie sollen das Design für ein einfaches Wahlsystem entwerfen. Das System soll dabei die Verteilung der Stimmen auf die einzelnen Parteien ermöglichen. Zusätzlich soll es verschiedene Darstellungen dieser Daten erlauben: Eine Tabelle, in der die Daten gelesen und auch eingegeben werden können, und ein Diagramm als alternative Darstellung der Informationen. Das System soll mit dem *Model-View-Controller* Muster modelliert werden.

(a) Beschreiben Sie das *Model-View-Controller* Muster:

(i) Beschreiben Sie das Problem, welches das Muster adressiert.

Lösungsvorschlag

Das MVC-Muster wird verwendet, um spätere Änderungen bzw. Erweiterungen zu vereinfachen. Dies unterstützt somit auch die Wiederverwendbarkeit.

(ii) Beschreiben Sie die Aufgaben der Komponenten, die im Muster verwendet werden.

Lösungsvorschlag

Im Modell werden die Daten verwaltet, die View ist für die Darstellung der Daten sowie die Benutzerinteraktion zuständig und der Controller übernimmt die Steuerung zwischen View und Modell.

Das MVC-Muster ist aus den drei Entwurfsmustern Beobachter, Kom-

positum und Strategie zusammengesetzt.

(b) Modellieren Sie das System unter Anwendung des Musters:

- (i) Entwerfen Sie ein UML Klassendiagramm.
- (ii) Implementieren Sie eine setup-Methode, die das Objektmodell erstellt.

## 66116 / 2014 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 3

Im Folgenden ist ein Algorithmus angegeben, der für eine positive Zahl `until` die Summe aller Zahlen bildet, die kleiner als `until` und Vielfache von 4 oder 6 sind. Für nicht positive Zahlen soll 0 zurückgegeben werden. Der Algorithmus soll also folgender Spezifikation genügen:

$\text{until} > 0 \Rightarrow \text{specialSums}(\text{until}) = \sum \{y \mid 0 < y < \text{until} \wedge (y \% 4 = 0 \vee y \% 6 = 0)\}$

$\text{until} \leq 0 \Rightarrow \text{specialSums}(\text{until}) = 0$

wobei `%` den Modulo-Operator bezeichnet.

```
public static long specialSums(int until) {
 long sum = 0; // 0
 if (until > 0) { // 1
 for (int i = 1; i <= until; i++) { // 2 // 5
 if (i % 4 == 0 || i % 6 == 0) { // 3
 sum += i; // 4
 }
 }
 }
 return sum; // 6
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2014/herbst/SpecialSum.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2014/herbst/SpecialSum.java)

Beachten Sie, dass der Algorithmus nicht der Spezifikation genügt. Der Fehler liegt in der Bedingung der for-Schleife. Der Fehler kann jedoch einfach korrigiert werden indem die Bedingung

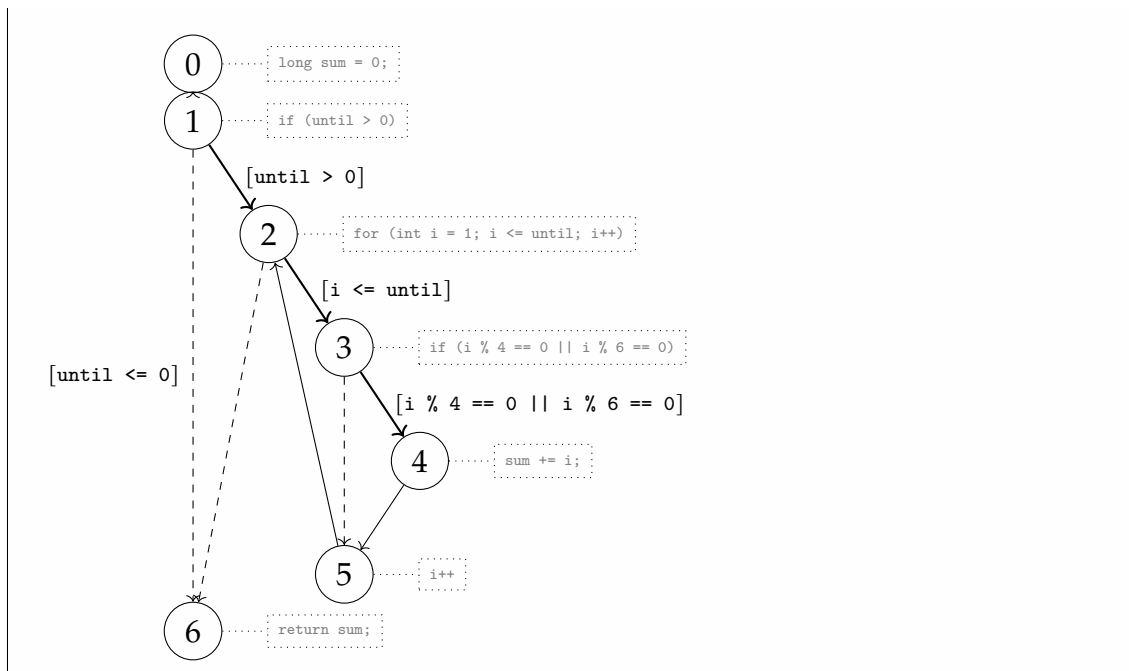
$$i \leq \text{until} \text{ in } i < \text{until}$$

geändert würde.

(a) Zeichnen Sie das zum Programm gehörige Ablaufdiagramm.

Lösungsvorschlag





C0-Test  
Anweisungsüberdeckung  
(Statement Coverage)  
C1-Test Zweigüberdeckung  
(Branch Coverage)

- (b) Schreiben Sie einen Testfall, der das Kriterium „100% Anweisungsüberdeckung“ erfüllt, aber den Fehler trotzdem nicht aufdeckt.

Lösungsvorschlag

Der Fehler fällt nur dann auf, wenn `until` durch 4 oder 6 ohne Rest teilbar ist. `until = 0%4` oder `until = 0%6`. Wähle daher den Testfall  $\{(1, 0)\}$ . Alternativ kann für die Eingabe auch 2, 3, 5, 7, 9, 10, 11, 13, ... gewählt werden.

- (c) Schreiben Sie einen Testfall, der das Kriterium „100% Zweigüberdeckung“ erfüllt, aber den Fehler trotzdem nicht aufdeckt.

Lösungsvorschlag

Betrachte den Testfall  $\{(0, 0), (5, 4)\}$ .

**erste if-Bedingung** Das erste Tupel mit `until = 0` stellt sicher, dass die erste `if`-Bedingung **false** wird.

**Bedingung der for-Schleife** Für die zweite Eingabe `until = 5` werden für `i` die Werte 1, 2, 3, 4, 5, 6 angenommen. Wobei für `i = 6` die Bedingung der for-Schleife **false** ist.

**Innere if-Bedingung** Für `i = 1, 2, 3, 5` wird die innere `if`-Bedingung jeweils **false**, für `i = 4` wird sie **true**.

- (d) Schreiben Sie einen Testfall, der den Fehler aufdeckt. Berechnen Sie Anweisungsüberdeckung und Zweigüberdeckung ihres Testfalls.

Lösungsvorschlag

**Anweisungsüberdeckung** Wähle  $\{(4, 0)\}$ . Durch die fehlerhafte Bedingung in der for-Schleife wird der Wert `i = 4` akzeptiert. Da alle Anweisungen ausgeführt werden, wird eine Anweisungsüberdeckung mit 100%

erreicht.

**Verzweigungsüberdeckung** Da die erste Verzweigung nur zur Hälfte überdeckt wird und die anderen beiden vollständig, gilt für die Verzweigungsüberdeckung:

$$\frac{1+2+2}{2+2+2} = \frac{5}{6}$$

- (e) Es ist nicht immer möglich vollständige Pfadüberdeckung zu erreichen. Geben Sie einen gültigen Pfad des Programmes an, der nicht erreichbar ist. Ein Testfall kann als Menge von Paaren dargestellt werden, wobei jedes Paar  $(I, O)$  die Eingabe  $I$  und die zu dieser erwartete Ausgabe  $O$  darstellt.

Lösungsvorschlag

Ein gültiger Pfad im Kontrollflussgraphen wäre ① - ① - ② - ③ - ④ - ⑤ - ② - ⑥. Der Übergang von ③ auf ④ ist hier aber nicht möglich, da beim ersten Durchlaufen der for-Schleife (②)  $i$  immer 1 ist und 1 weder durch 4 noch durch 6 teilbar ist. Somit kann die Bedingung des inneren ifs (③) beim ersten Durchlauf nie *wahr* sein, womit immer der Übergang ③ - ⑤ zu Beginn genommen werden muss. Alle Pfade, die zu Beginn ③ - ④ enthalten, sind somit nicht überdeckbar.

## 66116 / 2014 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 5

Lösen Sie folgende Aufgabe in einer objektorientierten Programmiersprache Ihrer Wahl. Ein Regal habe 5 Fächer, in die je 30 Disks passen. Das Regal beinhaltet DVDs, CDs und BDs der Genres *Musik*, *Action*, *Komödie*, *Thriller* und *Fantasy*. Jede Disk ist mit 1 bis 10 bewertet, wobei 10 für sehr gut steht.

- Deklarieren Sie einen Aufzählungsdatentyp für den **Typ** der Disk. Deklarieren Sie einen weiteren Aufzählungsdatentyp für das **Genre** der Disk.

Lösungsvorschlag

```
public enum Typ {
 DVD, CD, BD
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2014/herbst/regal/Typ.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2014/herbst/regal/Typ.java)

```
public enum Genre {
 MUSIK, ACTION, KOMOEDIE, THRILLER, FANTASY
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2014/herbst/regal/Genre.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2014/herbst/regal/Genre.java)

- Deklarieren Sie eine Klasse **Disk**, die den Typ, das Genre, die Bewertung, sowie den Titel der Disk speichert.

```

public class Disk {
 private Typ typ;
 private Genre genre;
 private int bewertung;
 private String titel;

 /**
 * Erzeuge eine neue Disk. Die Bewertung wird automatisch mit Hilfe des
 * ↪ Genres
 * berechnet.
 *
 * @param typ Typ der Disk
 * @param genre Genre der Disk
 * @param titel Titel der Disk
 */
 public Disk(Typ typ, Genre genre, String titel) {
 this.typ = typ;
 this.genre = genre;
 this.titel = titel;
 this.bewertung = (int) erstelleStdBewertung(this.genre);
 }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2014/herbst/regal/Disk.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2014/herbst/regal/Disk.java)

- Deklarieren Sie ein Array `Regal`, mit den Abmessungen des oben genannten Regals, das Disks enthält. Initialisieren Sie das Array als leeres Regal.

```

public class Regal {
 private Disk[] [] regal = new Disk[5][30];

 /**
 * Erzeuge eine neues (leeres) Regal.
 */
 public Regal() {
 for (int i = 0; i < regal.length; i++) {
 for (int j = 0; j < regal[i].length; j++) {
 regal[i][j] = null;
 }
 }
 }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2014/herbst/regal/Regal.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2014/herbst/regal/Regal.java)

- Initialisieren Sie die Bewertung einer Disk. Schreiben sie dazu eine rekursive Methode `erstelleStdBewertung`, der ein Genre einer Disk übergeben wird und die die Bewertungen für alle Disks nach folgenden Regeln bezüglich des Genres vergibt:
  - Eine Disk mit Genre Musik wird mit einer 3 bewertet.
  - Komödien werden mit 2 bewertet.

- Thriller werden mit dem zweifachen einer Komödie bewertet
- Ein Fantasy-Film wird mit dem 1,5 fachen eines Thrillers bewertet.
- Ein Actionfilm wird wie ein Thriller bewertet.

Lösungsvorschlag

```
public double erstelleStdBewertung(Genre genre) {
 if (genre.equals(Genre.MUSIK)) {
 return 3;
 } else if (genre.equals(Genre.KOMOEDIE)) {
 return 2;
 } else if (genre.equals(Genre.THRILLER)) {
 return 2 * erstelleStdBewertung(Genre.KOMOEDIE);
 } else if (genre.equals(Genre.FANTASY)) {
 return 1.5 * erstelleStdBewertung(Genre.THRILLER);
 } else if (genre.equals(Genre.ACTION)) {
 return erstelleStdBewertung(Genre.THRILLER);
 } else {
 return 0;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2014/herbst/regal/Disk.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2014/herbst/regal/Disk.java)

- Schreiben Sie eine Methode `mittlereBewertung`, die die mittlere Bewertung der Disks im Regal berechnet.

Lösungsvorschlag

```
public double mittlereBewertung() {
 int anzahl = 0;
 int bewertungspunkteGesamt = 0;
 for (int i = 0; i < regal.length; i++) {
 for (int j = 0; j < regal[i].length; j++) {
 if (regal[i][j] != null) {
 anzahl++;
 bewertungspunkteGesamt += regal[i][j].gibBewertung();
 }
 }
 }
 if (anzahl == 0) {
 // Falls das Regal komplett leer ist.
 return 0;
 }
 return bewertungspunkteGesamt / anzahl;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2014/herbst/regal/Regal.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2014/herbst/regal/Regal.java)

## 66116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1

In einer Musik-Datenbank sollen folgende Informationen zu Interpreten und deren CDs modelliert werden:

- Zu einem Interpreten soll eine eindeutige ID, der Name, das Jahr seines Bühnensstarts, seine Geschäftsadresse sowie sein Musikgenre angegeben sein. Das Musikgenre kann mehrere Werte umfassen (mehrwertiges Attribut).
  - Eine CD hat eine eindeutige ID, einen Namen (Titel), einen Interpreten, ein Erscheinungsdatum und bis zu 20 Positionen (Musikstücke). An jeder Position steht ein Musikstück. Für dieses ist der Titel und die Länge in Sekunden angegeben.
  - Eine CD kann Auszeichnungen - z. B. vom Typ goldene Schallplatte oder Emmy bekommen. Ebenso kann auch ein einzelnes Musikstück Auszeichnungen bekommen.
- (a) Modellieren Sie das oben dargestellte Szenario möglichst vollständig in einem ER-Modell. Verwenden Sie, wann immer möglich, (binäre oder auch höherstellige) Relationships. Modellieren Sie Musikstücke in einem schwachen Entity-Typen.
- (b) Übertragen Sie Ihr ER-Modell - bis auf die Typen zu den Auszeichnungen - ins relationale Datenmodell. Erstellen Sie dazu Tabellen mit Hilfe von CREATE TABLE-Statements in Sov. Berücksichtigen Sie die Fremdschlüsselbeziehungen.
- (c) Es soll die Integritätsbedingung eingehalten werden, so dass die Anzahl der Positionen auf einer CD höchstens 20 ist. Schreiben Sie ein SELECT-Statement, das diese Integritätsbedingung überprüft, indem es die verletzenden CDs ausgibt.
- (d) Geben Sie geeignete INSERT-Statements an, die in alle beteiligten Tabellen jeweils mindestens ein Tupel einfügen, so dass alle Integritätsbedingungen erfüllt sind, nachdem alle Einfügungen ausgeführt wurden. Lediglich zu den Auszeichnungen müssen keine Tupel eingefügt werden.

## 66116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Formulieren Sie in SQL die folgenden Anfragen, Views bzw. Datenmanipulations-Statements an Teile der Musik-Datenbank aus Teilaufgabe DB.1:

Interpret (Interpreten\_ID, Name, Bühnenstart, Geschäftsadresse), CD (CD\_ID, Name, Interpreten\_ID, Erscheinungsdatum), Musikstück (CD\_ID, Position, Titel, Länge), Auszeichnung\_CD (CD\_ID, Typ), Auszeichnung\_Stück (CD\_ID, Position, Typ).

- (a) Welche CDs hat „Adele“ herausgebracht? Geben Sie die Namen der CDs aus.

Lösungsvorschlag

```
SELECT c.Name DISTINCT
FROM CD c, Interpret i
WHERE
 i.Interpreten_ID = c.Interpreten_ID AND
 i.name = 'Adele';
```

- (b) Geben Sie für alle Interpreten - gegeben durch die ID und den Namen - die Anzahl ihrer veröffentlichten CDs an.

Lösungsvorschlag

MAX  
OR  
INSERT  
UPDATE

```
SELECT i.Interpreten_ID, i.Name, COUNT(*)
FROM Interpret i, CD c
WHERE
 i.Interpreten_ID = c.Interpreten_ID
GROUP BY i.Interpreten_ID, i.Name;
```

- (c) Geben Sie die Länge des längsten Musikstücks auf der CD mit dem Namen „Thriller“ des Interpreten „Michael Jackson“ an.

Lösungsvorschlag

```
SELECT MAX(m.Länge)
FROM Interpret i, CD c, Musikstück m
WHERE
 m.CD_ID = c.CD_ID AND
 i.Name = 'Michael Jackson' AND
 c.Name = 'Thriller';
```

- (d) Geben Sie die Namen aller Interpreten aus, die eine Auszeichnung für eine CD oder eines ihrer Musikstücke bekommen haben.

Lösungsvorschlag

```
SELECT i.Name
FROM Auszeichnung_CD acd, Auszeichnung_Stück ast, CD c, Interpret i
WHERE
 (acd.CD_ID = c.CD_ID OR ast.CD_ID = c.CD_ID) AND
 c.Interpreten_ID = i.Interpreten_ID;
```

- (e) Fügen Sie ein, dass „Adele“ einen „Emmy“ für ihre CD mit dem Namen „Adele 21“ bekommen hat.

Lösungsvorschlag

```
INSERT INTO Auszeichnung_CD VALUES
(
 (
 SELECT c.CD_ID
 FROM CD c, Interpret i
 WHERE
 c.Name = 'Adele 21' AND
 i.Interpreten_ID = c.Interpreten_ID AND
 i.Name = 'Adele'
),
 'Emmy'
);

-- Test
SELECT * FROM Auszeichnung_CD;
```

- (f) Ändern Sie die Geschäftsadresse von „Genesis“ auf „Hollywood Boulevard 13, Los Angeles“.

```
-- Test
SELECT * FROM Interpret;

UPDATE Interpret
SET Geschäftsadresse = 'Hollywood Boulevard 13, Los Angeles'
WHERE Name = 'Gensis';

-- Test
SELECT * FROM Interpret;
```

### 66116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Gegeben sei das Relationenschema  $R=(U, F)$  mit der Attributmenge

$$\{ U \} A, B, C, D, E$$

und der folgenden Menge  $F$  von funktionalen Abhängigkeiten:

$$FA = \left\{ \begin{array}{l} \{ A \} \rightarrow \{ B \}, \\ \{ A, B, C \} \rightarrow \{ D \}, \\ \{ D \} \rightarrow \{ B, C \}, \end{array} \right\}$$

- Geben Sie alle Schlüssel für das Relationenschema  $R$  (jeweils mit Begründung) sowie die Nichtschlüsselattribute an.
- Ist  $R$  in 3NF bzw. in BCNF? Geben Sie jeweils eine Begründung an.
- Geben Sie eine Basis  $G$  von  $F$  an. Zerlegen Sie  $R$  mittels des Synthesalgorithmus in ein 3NF-Datenbankschema. Es genügt, die resultierenden Attributmengen anzugeben.

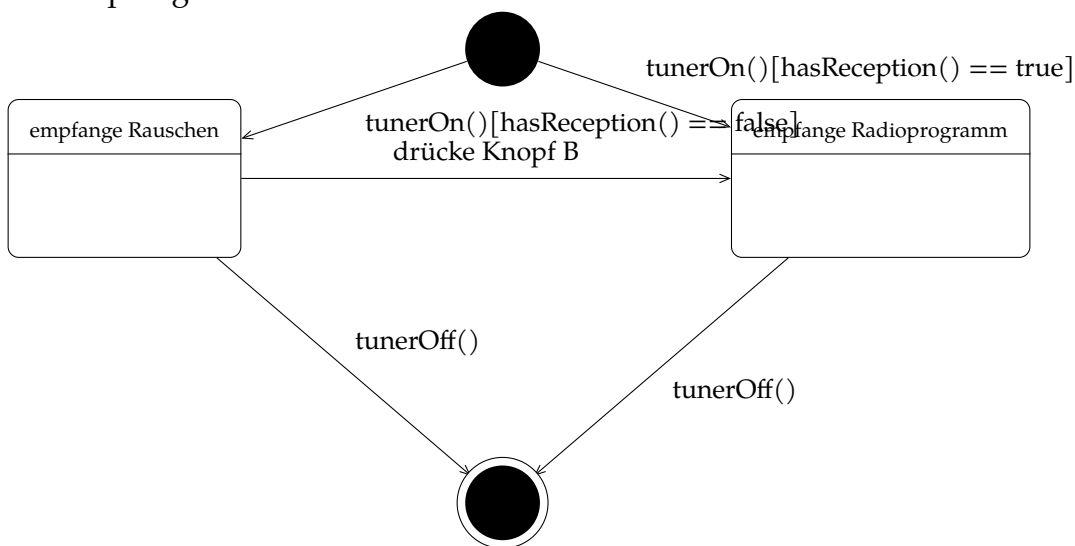
### 66116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2

Erstellen Sie ein UML-Zustandsdiagramm für einen Radiotuner. Mit dem Radiotuner können Sie ein Radioprogramm auf der Frequenz  $f$  empfangen. Beim Einschalten wird  $f$  auf 87,5 MHz gesetzt und der Tuner empfängt. Sie können nun die Frequenz in 0,5 MHz Schritten erhöhen oder senken. Bitte beachten Sie, dass das Frequenzband für den Radioempfang von 87,5 MHz bis 108 MHz reicht. Sobald  $f$  87,5 MHz unter- bzw. 108 MHz überschreitet, soll  $f$  auf 108 MHz bzw. 87,5 MHz gesetzt werden. Weiterhin kann ein Suchmodus gestartet werden, der automatisch die Frequenz erhöht, bis ein Sender empfangen wird. Wird während des Suchmodus die Frequenz verändert oder erneut Suchen ausgeführt, dann wird die Suche beendet (und, je nach Knopf, ggf. noch die Frequenz um 0,5 MHz verändert).

Die Klasse `RadioTuner` besitzt folgende Methoden:

- tunerOn()
- tunerOff()
- increaseFrequency()
- decreaseFrequency()
- seek()
- hasReception()

Hinweis: Die Hilfsmethode `hasReception()` liefert `true` zurück, genau dann wenn ein Sender empfangen wird.



### 66116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 3

In dieser Aufgabe implementieren Sie ein konzeptionelles Datenmodell für eine Firma, die Personendaten von Angestellten und Kunden verwalten möchte. Gegeben seien dazu folgende Aussagen:

- Eine *Person* hat einen *Namen* und ein *Geschlecht* (männlich oder weiblich).
- Ein *Angestellter* ist eine *Person*, zu der zusätzlich das monatliche *Gehalt* gespeichert wird.
- Ein *Kunde* ist eine *Person*, zu der zusätzlich eine *Kundennummer* hinterlegt wird.

- (a) Geben Sie in einer objektorientierten Programmiersprache Ihrer Wahl (geben Sie diese an) eine Implementierung des aus den obigen Aussagen resultierenden konzeptionellen Datenmodells in Form von **Klassen** und **Interfaces** an. Gehen Sie dabei wie folgt vor:



- Schreiben Sie ein Interface `Person` sowie zwei davon ererbende Interfaces `Angestellter` und `Kunde`. Die Interfaces sollen jeweils lesende Zugriffsmethoden (Getter) die entsprechenden Attribute (Name, Geschlecht, Gehalt, Kundennummer) deklarieren.

Lösungsvorschlag

```
public interface Person {
 String getName();

 char getGeschlecht();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/Person.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Person.java)

```
public interface Angestellter extends Person {
 int getGehalt();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/Angestellter.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Angestellter.java)

```
public interface Kunde extends Person {
 int getKundennummer();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/Kunde.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Kunde.java)

- Schreiben Sie eine abstrakte Klasse `PersonImpl`, die das Interface `Person` implementiert. Für jedes Attribut soll ein Objektfeld angelegt werden. Außerdem soll ein Konstruktor definiert werden, der alle Objektfelder initialisiert.

Lösungsvorschlag

```
public abstract class PersonImpl implements Person {
 protected String name;
 protected char geschlecht;

 public PersonImpl(String name, char geschlecht) {
 this.name = name;
 this.geschlecht = geschlecht;
 }

 public String getName() {
 return name;
 }

 public char getGeschlecht() {
 return geschlecht;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/PersonImpl.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/PersonImpl.java)

- Schreiben Sie zwei Klassen `AngestellterImpl` und `KundeImpl`, die von `PersonImpl` erben und die jeweils dazugehörigen Interfaces implementieren. Es sollen wiederum Konstruktoren definiert werden, die alle Objektfelder

initialisieren und dabei auf den Konstruktor der Basisklasse `PersonImpl` Bezug nehmen.

Lösungsvorschlag

```
public class AngestellterImpl extends PersonImpl implements Angestellter
↪ {
 protected int gehalt;

 public AngestellterImpl(String name, char geschlecht, int gehalt) {
 super(name, geschlecht);
 this.gehalt = gehalt;
 }

 public int getGehalt() {
 return gehalt;
 }
}
```

Code-Beispiel auf Github ansehen:  
[src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/AngestellterImpl.java](#)

```
public class KundeImpl extends PersonImpl implements Kunde {
 protected int kundennummer;

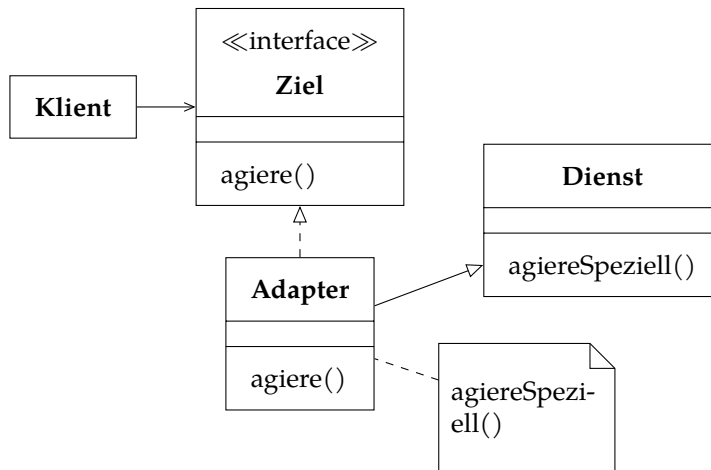
 public KundeImpl(String name, char geschlecht, int kundennummer) {
 super(name, geschlecht);
 this.kundennummer = kundennummer;
 }

 public int getKundennummer() {
 return kundennummer;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/KundeImpl.java](#)

- (b) Verwenden Sie das Entwurfsmuster **Adapter**, um zu ermöglichen, dass vorhandene Angestellte die Rolle eines Kunden einnehmen können. Der Adapter soll eine zusätzliche Klasse sein, die das Kunden-Interface implementiert. Wenn möglich, sollen Methodenaufrufe an den adaptierten Angestellten delegiert werden. Möglicherweise müssen Sie neue Objektfelder einführen.

### Exkurs: Entwurfsmuster „Adapter“



**Ziel (Target)** Das Ziel definiert die Schnittstelle, die der Klient nutzen kann.

**Klient (Client)** Der Klient nutzt Dienste über inkompatible Schnittstellen und greift dabei auf adaptierte Schnittstellen zurück.

**Dienst (Adaptee)** Der Dienst bietet wiederzuverwendende Dienstleistungen mit fest definierter Schnittstelle an.

**Adapter** Der Adapter adaptiert die Schnittstelle des Dienstes auf die Schnittstelle zum Klienten.

```

/**
 * GoF: Adaptee
 */
public class Dienst {

 public void agiereSpeziell() {
 System.out.println("Agiere speziell!");
 }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Dienst.java](https://github.com/bschlangaul/entwurfsmuster/tree/master/adapter/allgemein/Dienst.java)

```

public interface Ziel {
 public void agiere();
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Ziel.java](https://github.com/bschlangaul/entwurfsmuster/tree/master/adapter/allgemein/Ziel.java)

```

public class Adapter extends Dienst implements Ziel {

 @Override
 public void agiere() {
 System.out.print("agiere: ");
 agiereSpeziell();
 }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Adapter.java](https://github.com/bschlangaul/entwurfsmuster/tree/master/adapter/allgemein/Adapter.java)

```

public class Klient {

 public static void main(String[] args) {
 new Adapter().agiere();
 // agiere: Agiere speziell!
 }
}

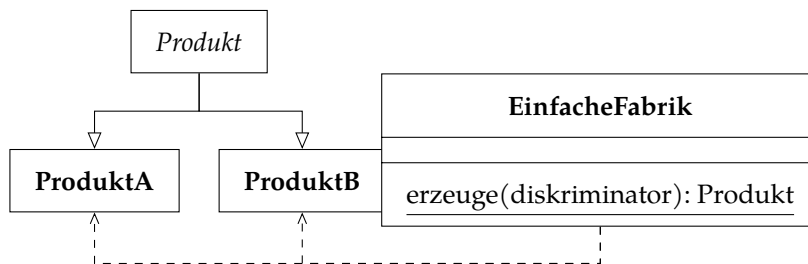
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Klient.java](https://github.com/bschlangaul/entwurfsmuster/adapter/allgemein/Klient.java)

- (c) Verwenden Sie das Entwurfsmuster **Simple Factory**, um die Erzeugung von Angestellten, Kunden, sowie Adapter-Instanzen aus Aufgabe (b) zu vereinheitlichen. Die entsprechende Erzeugungs-Methode soll neben einem Typ-Diskriminator (z. B. einem Aufzählungstypen oder mehreren booleschen Werten) alle Parameter übergeben bekommen, die für den Konstruktor irgendeiner Implementierungsklasse des Interface **Person** notwendig sind.

Hinweis: Um eine Adapter-Instanz zu erzeugen, müssen Sie möglicherweise zwei Konstruktoren aufrufen.

#### Exkurs: Entwurfsmuster „Einfache Fabrik“



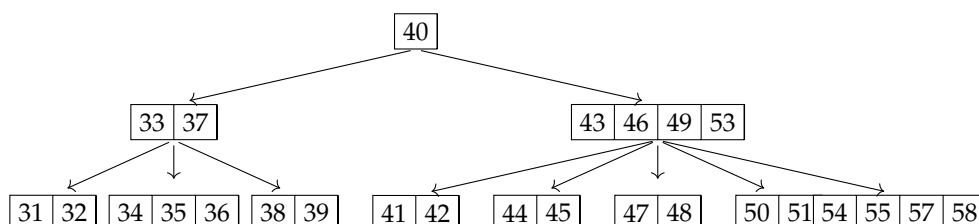
**EinfacheFabrik** Eine Klasse mit einer Erzeugungsmethode, die über eine größere Bedingung verschiedene Objekt instanziiert.

**Produkt** Eine abstrakte Klasse, die von den konkreten Produkten geerbt wird.

**KonkretesProdukt** Ein konkretes Produkt, das von der einfachen Fabrik erzeugt wird.

## 66116 / 2015 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 3

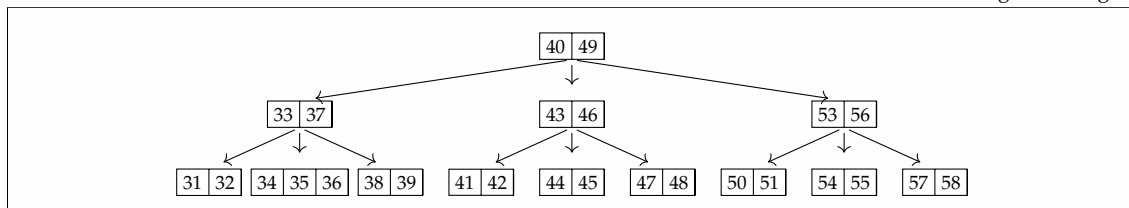
Als Indexstruktur einer Datenbank sei folgender B-Baum ( $k = 2$ ) gegeben:



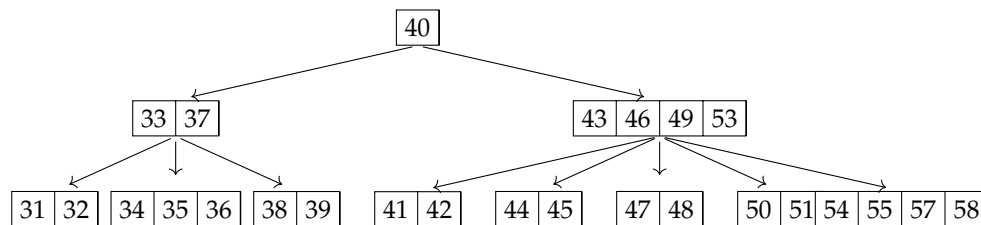
Führen Sie nacheinander die folgenden Operationen aus. Geben Sie die auftretenden Zwischenergebnisse an. Teilbäume, die sich in einem Schritt nicht verändern, müssen nicht erneut gezeichnet werden. Sollten Wahlmöglichkeiten auftreten, so sind größere Schlüsselwerte bzw. weiter rechts liegende Knoten zu bevorzugen.

(a) Einfügen des Wertes 56

Lösungsvorschlag



(b) Löschen des Wertes 37



## 66116 / 2015 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Leider ist das Klassendiagramm der folgenden Klassen verloren gegangen. Führen Sie ein Reverse Engineering durch und erstellen Sie aus dem Quellcode ein vollständiges UML-Klassendiagramm inklusive aller Klassen, Schnittstellen, Attribute, Methoden, Konstruktoren, Sichtbarkeiten, Assoziationen, Rollennamen, Multiplizitäten, Navigationspfeilen und evtl. Stereotypen. Der Quellcode innerhalb von Methoden und Konstruktoren soll nicht übertragen werden, wohl aber die Methodensignaturen. Assoziationsnamen und deren Leserichtung lassen sich aus dem Quellcode nur schwer erahnen und sollen deshalb ebenfalls weggelassen werden.

```
public abstract class Display implements PixelPainter {
 protected HardwareMatrix hardwareMatrix;
 protected int lastPaintedX;
 protected int lastPaintedY;

 public Display(HardwareMatrix hardwareMatrix) {
 this.hardwareMatrix = hardwareMatrix;
 }

 public int getWidth() {
 return hardwareMatrix.getWidth() / getWidthFactor();
 }

 public int getHeight() {
```

```
 return hardwareMatrix.getHeight() / getHeightFactor();
}

public void clear() {
 // some longer code
}

protected abstract int getWidthFactor();

protected abstract int getHeightFactor();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/reverse/Display.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/Display.java)

```
import java.awt.Color;

public interface PixelPainter {
 void set(int x, int y, Color color);

 int getHeight();

 int getWidth();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/reverse/PixelPainter.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/PixelPainter.java)

```
public interface HardwareMatrix {
 void set(int x, int y, int v);

 int getWidth();

 int getHeight();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/reverse/HardwareMatrix.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/HardwareMatrix.java)

```
import java.awt.Color;

@SuppressWarnings({ "unused" })
public class DisplayUnion extends RGBDisplay {
 public static final int MAX_DISPLAY_COUNT = 50;

 private int currentDisplayCount;

 private Display[] displays;

 public DisplayUnion(Display[] displays) {
 super(null);
 }

 public int getDisplayCount() {
 return 0;
 }

 protected int getWidthFactor() {
 return 1;
 }
}
```

```
}

protected int getHeightFactor() {
 return 1;
}

public void set(int x, int y, Color color) {
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/reverse/DisplayUnion.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/DisplayUnion.java)

```
import java.awt.Color;

public class RGBDisplay extends Display {
 public RGBDisplay(HardwareMatrix matrix) {
 super(matrix);
 }

 public void set(int x, int y, Color color) {
 }

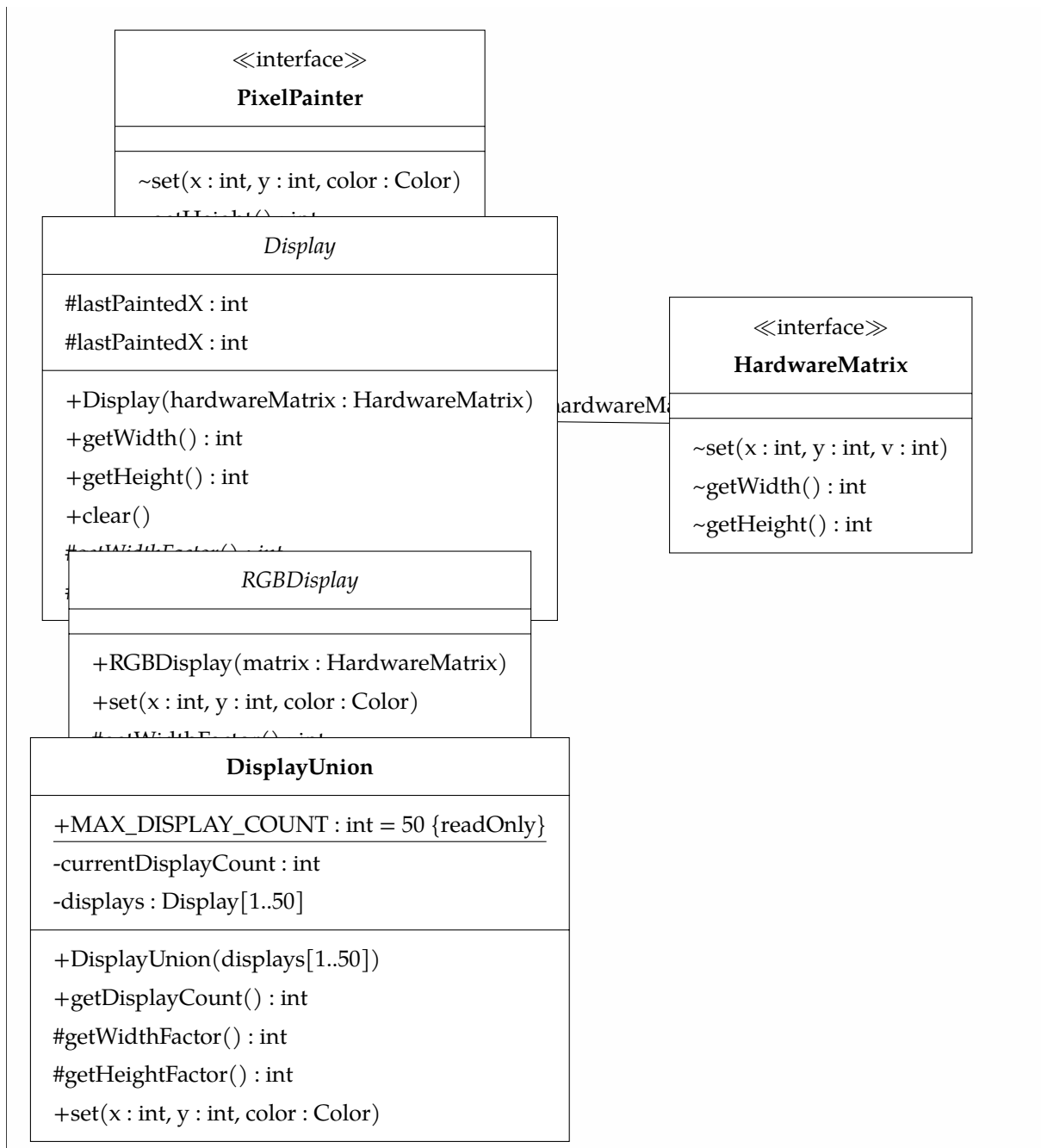
 protected int getWidthFactor() {
 return 3;
 }

 protected int getHeightFactor() {
 return 1;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/reverse/RGBDisplay.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/RGBDisplay.java)

Lösungsvorschlag

---



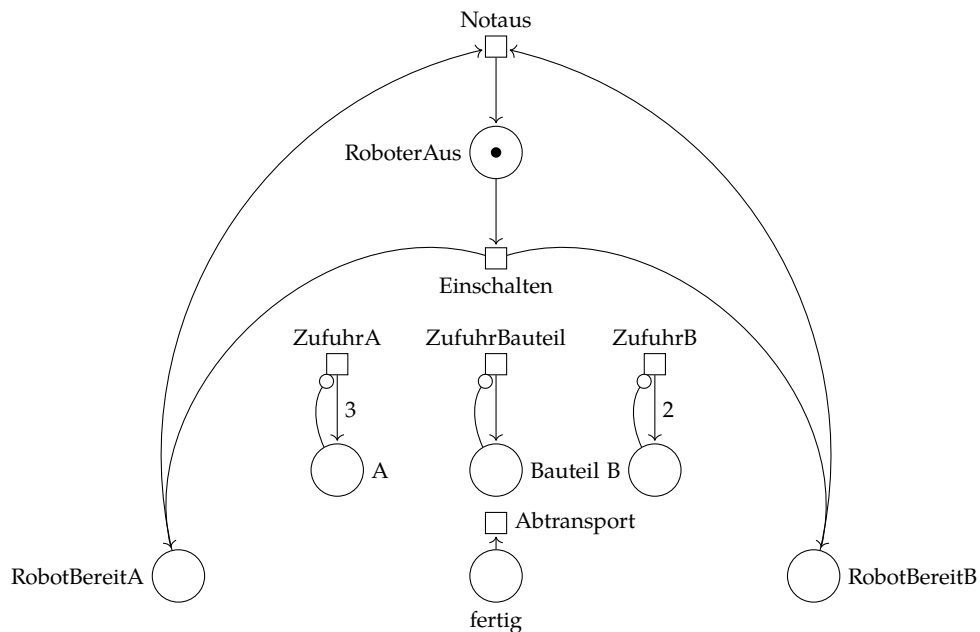
## 66116 / 2015 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 3

Das folgende Grundgerüst stammt aus dem Petri-Netz-Modell einer Automatisierungsanlage mit zwei Robotern, das Sie auf Ihr Blatt übernehmen und geeignet um weitere Plätze (Stellen), Transitionen, Kapazitäten, Gewichte und Markierungen so ergänzen sollen, dass die darunter angegebenen Anforderungen eingehalten werden:

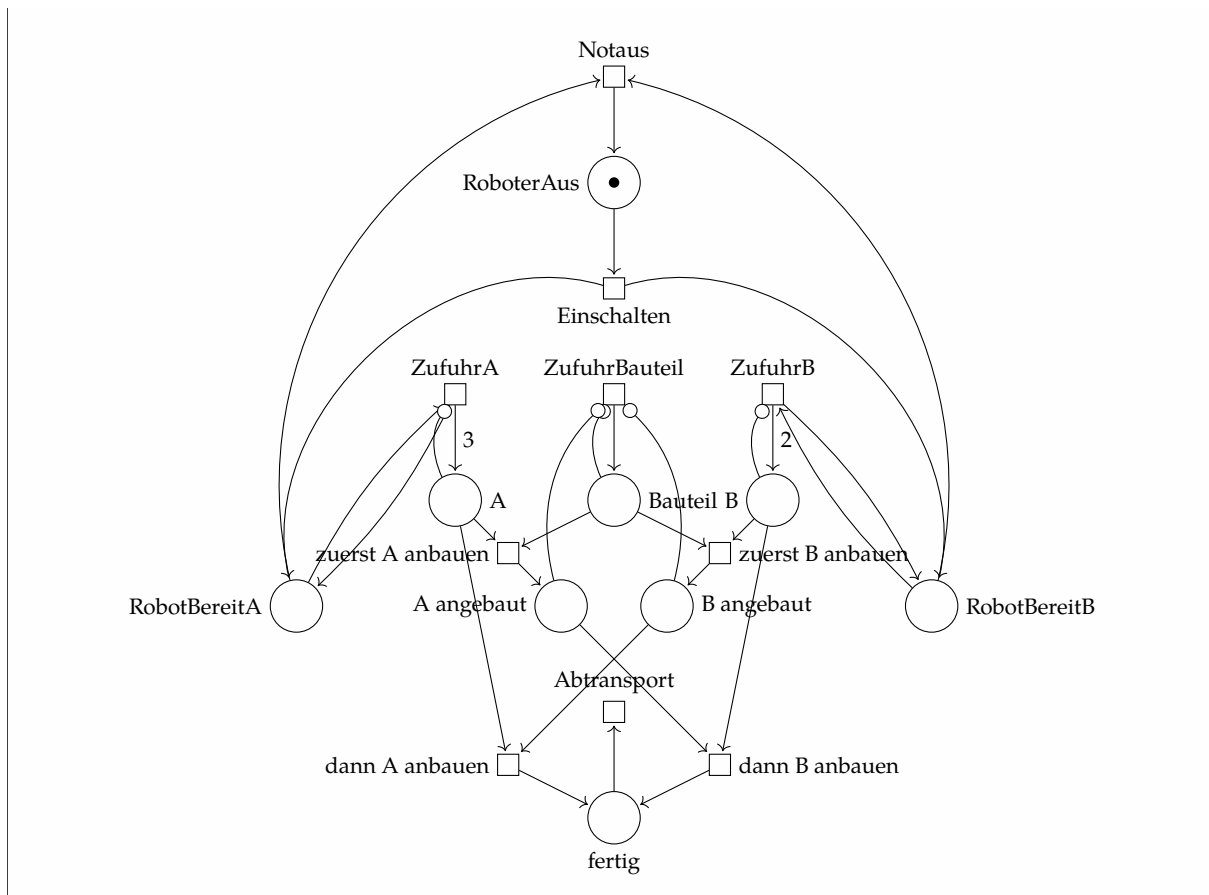
In der Anlage arbeiten zwei Roboter A und B, die über einen Schalter „Einschalten“ aktiviert und *jederzeit* über einen „Notaus“-Schalter deaktiviert werden können müssen. Aufgabe der Roboter ist es, jeweils abwechselnd an Bauteilen zu arbeiten, die einzeln über ein Förderband „ZufuhrBauteil“ ins System eingefahren und nach der Fer-



tigstellung mittels „Abtransport“ aus dem System abgeführt werden. Roboter A bringt genau 3 Anbauten vom Typ „A“ an jedes Bauteil an und Roboter B macht das entsprechend mit genau 2 Anbauten vom Typ „B“. Die Anbauten werden jeweils passend über „ZufuhrA“ auf „A“ bzw. mittels „ZufuhrB“ auf „B“ bereitgestellt. Die beiden Roboter dürfen *niemals* gleichzeitig an einem Bauteil arbeiten — die Reihenfolge, in der sie darauf zugreifen, ist jedoch beliebig und darf von Bauteil zu Bauteil frei variieren. Sobald einer der Roboter mit der Arbeit an einem neuen Bauteil begonnen hat, darf kein weiteres Bauteil angenommen werden, ehe der jeweils andere Roboter nicht ebenfalls mit seiner Arbeit am gleichen Bauteil fertig geworden ist (und das fertige Bauteil „auf den Platz ‚fertig‘ legt“). Aus Platzgründen darf höchstens ein Bauteil auf dem Ausgangsplatz „fertig“ abgestellt werden, ehe es abtransportiert wird.



Lösungsvorschlag



## 66116 / 2015 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 1

Wir wollen eine relationale Datenbankstruktur für ein Online-Auktionshaus modellieren.

Das Auktionshaus hat Mitglieder. Diese Mitglieder haben Kundennummern, Namen und Adressen. Sie können Verkäufer und/oder Käufer sein. Verkäufer können eine Laden-URL (eine Subdomain) erhalten, die zu einer Seite mit ihren aktuellen Auktionen führt. Verkäufer können neue Auktionen starten. Die Auktionen eines Verkäufers werden durchnummeriert. Diese Auktionsnummer ist nur je Verkäufer eindeutig. Jede Auktion hat ein Mindestgebot und eine Ablaufzeit.

Auf Auktionen können Gebote abgegeben werden. Die Gebote auf eine Auktion werden nach ihrem Eintreffen nummeriert. Diese Nummer ist nur innerhalb einer Auktion eindeutig. Zu einem Gebot werden noch die Zeit des Gebots sowie der gebotene Geldbetrag angegeben. Die Gebote werden von Käufern abgegeben. Jedes Gebot muss einem Käufer zugeordnet sein.

Jede Auktion besteht aus einer Menge von Artikeln, aber aus mindestens einem. Artikel haben eine Beschreibung und können über ihre Artikel-ID identifiziert werden. Zur Katalogisierung der Artikel gibt es Kategorien. Kategorien haben eine eindeutige ID und einen Namen. Jede Kategorie kann Subkategorien besitzen. Auch Subkategorien sind Kategorien (und können damit weitere Subkategorien besitzen). Jeder Artikel kann beliebig vielen Kategorien zugeordnet werden.

Entwerfen Sie für das beschriebene Szenario ein ER-Diagramm. Bestimmen Sie hier-

zu:

- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- ein passendes ER-Diagramm,
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen,
- die Funktionalitäten der Relationship-Typen, welche Sie ebenfalls in das ER-Diagramm eintragen.

**66116 / 2015 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2**

Gegeben sei folgendes verallgemeinerte Relationenschema in 1. Normalform:

$$R(A, B, C, D, E, F, G, H)$$

Für R soll die folgende Menge FD von funktionalen Abhängigkeiten gelten:

$$\text{FA} = \left\{ \begin{array}{l} \{F\} \rightarrow \{E\}, \\ \{A\} \rightarrow \{B, D\}, \\ \{A, E\} \rightarrow \{D\}, \\ \{A\} \rightarrow \{E, F\}, \\ \{A, G\} \rightarrow \{H\}, \end{array} \right\}$$

Bearbeiten Sie mit diesen Informationen folgende Teilaufgaben. Vergessen Sie dabei nicht Ihr Vorgehen stichpunktartig zu dokumentieren und zu begründen.

- Bestimmen Sie alle Schlüsselkandidaten von R. Begründen Sie stichpunktartig, warum es außer den von Ihnen gefundenen Schlüsselkandidaten keine weiteren geben kann.
- Ist R in 2NF, 3NF?
- Berechnen Sie eine kanonische Überdeckung von FD. Es genügt, wenn Sie für jeden der vier Einzelschritte die Menge der funktionalen Abhängigkeiten als Zwischenergebnis angeben.
- Bestimmen Sie eine Zerlegung von R in 3NF. Wenden Sie hierfür den Synthesalgorithmus an.

**66116 / 2015 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3**

Gegeben seien folgende Relationen:

Mensch : {[ ID, MutterID, VaterID ]}

Mann : {[ ID ]}

Frau : {[ ID ]}

Das zugehörige ER-Modell für dieses relationale Datenbankschema sieht folgendermaßen aus:

Bearbeiten Sie folgende Teilaufgaben:

- (a) Finden Sie die Töchter der Frau mit ID 42.

Lösungsvorschlag

```
SELECT Mensch.ID
FROM Mensch, Frau
WHERE
 Mensch.MutterID = Frau.id AND
 Frau.ID = 42;
```

- (b) Gibt es Männer, die ihre eigenen Großväter sind? Formulieren Sie eine geeignete SQL-Anfrage.

Lösungsvorschlag

```
SELECT Mensch.ID
FROM Mann, Mensch
WHERE
 Mensch.ID = Mann.id AND (
 Mensch.VaterID IN (SELECT v.ID FROM Mensch v WHERE v.VaterID = Mensch.ID)
 OR
 Mensch.MutterID IN (SELECT v.ID FROM Mensch v WHERE v.VaterID =
→ Mensch.ID)
);
```

- (c) Definieren Sie eine View VaterKind (VaterID; KindID), die allen Vätern (VaterID) ihre Kinder (KinderID) zuordnet. Diese View darf keine NULL-Werte enthalten.

Lösungsvorschlag

```
-- Wir erzeuge bereits beim Erstellen der Datenbank diese View, damit
-- sie für spätere Aufgaben zur Verfügung steht.
DROP VIEW IF EXISTS VaterKind;
CREATE VIEW VaterKind AS
SELECT Mensch.VaterID, Mensch.ID as KindID
FROM Mensch
WHERE
 Mensch.VaterID IS NOT NULL;
SELECT * FROM VaterKind;
```

- (d) Verwenden Sie die View aus c), um alle Väter zurückzugeben, absteigend geordnet nach der Anzahl ihrer Kinder.

```
SELECT VaterID, COUNT(VaterID) as Anzahl
FROM VaterKind
GROUP BY VaterID
ORDER BY Anzahl DESC;
```

- (e) Hugo möchte mit folgender Anfrage auf Basis der View aus c) alle kinderlosen Männer erhalten:

```
SELECT VaterID
FROM VaterKind
GROUP BY VaterID
HAVING COUNT(KindID) = 0
```

- (i) Was ist das Ergebnis von Hugos Anfrage und warum?

Lösungsvorschlag

Die Anfrage liefert kein Ergebnis. Da die View laut Angabe keine Null-Werte enthalten darf, sind in der View nur Männer verzeichnet, die wirklich Väter sind.

- (ii) Formulieren Sie eine Anfrage, die tatsächlich alle kinderlosen Männer zurückliefert.

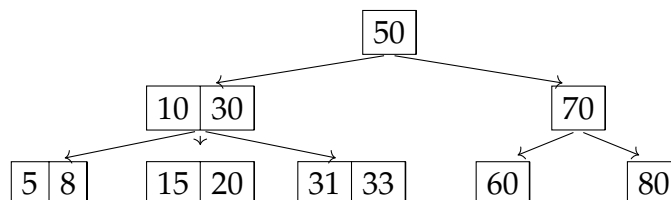
Lösungsvorschlag

```
SELECT * FROM Mann
EXCEPT
SELECT VaterID
FROM VaterKind
GROUP BY VaterID;
```

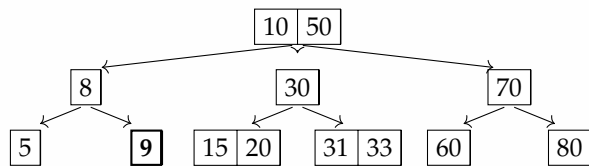
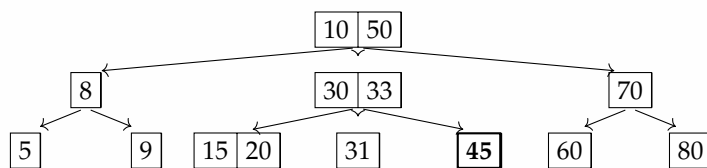
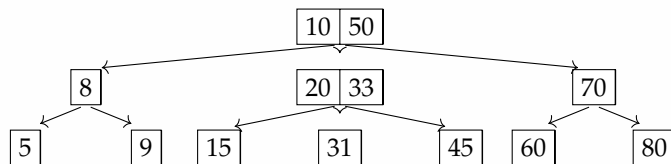
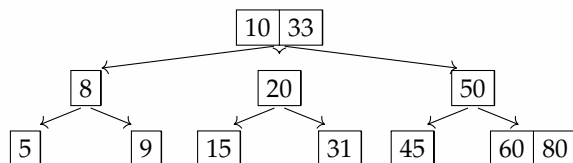
Hinweis: Denken Sie daran, dass SQL auch Mengenoperationen kennt.

## 66116 / 2015 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3

Gegeben ist der folgende B-Baum der Ordnung 3 (max. drei Kindknoten, max. zwei Schlüssel pro Knoten):



Fügen Sie die Werte 9 und 45 ein. Löschen Sie anschließend die Werte 30 und 70. Zeichnen Sie den Baum nach jeder Einfüge- bzw. Lösch-Operation.

**9 einfügen****45 einfügen****30 löschen****70 löschen****66116 / 2015 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 3**

Gegeben Sei folgendes Programm:

```
long doubleFac (long n) {
 /* P */ long df = 1;
 for (long x = n; x > 1; x -= 2) {
 df *= x;
 } /* Q */
 return df;
}
```

sowie die Vorbedingung  $P \equiv n \geq 0$  und Nachbedingung  $Q \equiv (df = n!!)$  wobei gilt

$$n!! := \begin{cases} 2^k \cdot k! & n \text{ gerade, } k := \frac{n}{2} \\ \frac{(2k)!}{2^k \cdot k!} & n \text{ ungerade, } k := \frac{n+1}{2} \end{cases}$$

**Exkurs: Fakultät**

Die Fakultät ist eine Funktion, die einer natürlichen Zahl das Produkt aller natürlichen Zahlen (ohne Null) kleiner und gleich dieser Zahl zuordnet. Für alle natürlichen Zahlen  $n$  ist

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n = \prod_{k=1}^n k$$

**Exkurs: Doppelfakultät**

Die seltener verwendete „Doppelfakultät“ oder „doppelte Fakultät“ ist für gerade  $n$  das Produkt aller geraden Zahlen kleiner gleich  $n$ . Für ungerade  $n$  ist es das Produkt aller ungeraden Zahlen kleiner gleich  $n$ . Sie ist definiert als:

$$n!! = \begin{cases} n \cdot (n-2) \cdot (n-4) \cdot \dots \cdot 2 & \text{für } n \text{ gerade und } n > 0, \\ n \cdot (n-2) \cdot (n-4) \cdot \dots \cdot 1 & \text{für } n \text{ ungerade und } n > 0, \\ 1 & \text{für } n \in \{-1, 0\} \end{cases}$$

Häufig werden anstelle der Doppelfakultät Ausdrücke mit der gewöhnlichen Fakultät verwendet.

Es gilt  $(2k)!! = 2^k k!$  und  $(2k-1)!! = \frac{(2k)!}{2^k k!}$

Zur Vereinfachung nehmen Sie im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können.

- (a) Welche der folgenden Bedingungen ist eine zum Beweisen der Korrektheit der Methode mittels wp-Kalkül (Floyd-Hoare-Kalkül) sinnvolle Schleifeninvariante?

- (i)  $df = n!! - x!! \wedge x \geq 1$
- (ii)  $df = (n - x)!! \wedge x \geq 1$
- (iii)  $df \cdot x!! = n!! \wedge x \geq 0$
- (iv)  $(df + x)!! = n!! \wedge x \geq 0$

Lösungsvorschlag

Zunächst wird der Code in einen äquivalenten Code mit while-Schleife umgewandelt:

```
long doubleFac (long n) {
 /* P */ long df = 1;
 long x = n ;
 while (x > 1) {
 df = df * x;
 x = x - 2;
 } /* Q */
 return df;
}
```

- (i)  $df = n!! - x!! \wedge x \geq 1$
- (ii)  $df = (n - x)!! \wedge x \geq 1$

Die ersten beiden Bedingungen sind unmöglich, da z. B. für  $n = 2$  nach der Schleife  $x = 0$  gilt und daher  $x \geq 1$  verletzt wäre.

$$(iii) \quad df \cdot x!! = n!! \wedge x \geq 0$$

Nach dem Ausschlussprinzip ist es daher die dritte Bedingung:  $I \equiv (df + x)!! = n!! \wedge x \geq 0$ .

$$(iv) \quad (df + x)!! = n!! \wedge x \geq 0$$

Die letzte kann es auch nicht sein, da vor der Schleife  $df = 1$  und  $x = n$  gilt,  $\delta(df + x)!! = (1 + n)!!$ . Jedoch ist offenbar  $(1 + n)!! \neq n!!$ .

$\Rightarrow$  Die Schleifeninvariante lautet:  $df \cdot x!! = n!! \wedge x \geq 0$

- (b) Zeigen Sie formal mittels wp-Kalkül, dass die von Ihnen gewählte Bedingung unmittelbar vor Beginn der Schleife gilt, wenn zu Beginn der Methode die Anfangsbedingung  $P$  gilt.

Lösungsvorschlag

Zu zeigen  $P \Rightarrow wp(\text{"Code vor der Schleife"}, I)$

$$\begin{aligned} wp(\text{"Code vor der Schleife"}, I) &\equiv wp(\text{"df = 1; x = n;"}, (df \cdot x)!! = n!! \wedge x \geq 0) \\ &\equiv wp(\text{"df = 1;"}, (df \cdot n)!! = n!! \wedge n \geq 0) \\ &\equiv wp("", (1 \cdot n)!! = n!! \wedge n \geq 0) \\ &\equiv n!! = n!! \wedge n \geq 0 \\ &\equiv n \geq 0 \\ &\equiv P \end{aligned}$$

Insbesondere folgt damit die Behauptung.

- (c) Zeigen Sie formal mittels wp-Kalkül, dass die von Ihnen gewählte Bedingung tatsächlich eine Invariante der Schleife ist.

Lösungsvorschlag

zu zeigen:  $I \wedge \text{Schleifenbedingung} \Rightarrow wp(\text{"Code in der Schleife"}, I)$

Bevor wir dies beweisen, zeigen wir erst  $x \cdot (x - 2)!! = x!!$ .

- Fall  $x$  ist gerade ( $n!! = 2^k \cdot k!$  für  $k := \frac{n}{2}$ ):

$$x \cdot (x - 2)!! = x \cdot 2^{\frac{x-2}{2}} \cdot (\frac{x-2}{2})! = x \cdot \frac{1}{2} \cdot 2^{\frac{x}{2}} \cdot (\frac{x}{2} - 1)! = 2^{\frac{x}{2}} \cdot (\frac{x}{2})! = x!!$$

Nebenrechnung (Division mit gleicher Basis:  $x^{a-b} = \frac{x^a}{x^b}$ ):

$$2^{\frac{x-2}{2}} = 2^{(\frac{x}{2}-2)} = \frac{2^{\frac{x}{2}}}{2^2} = \frac{2^{\frac{x}{2}}}{2^1} = \frac{2^{\frac{x}{2}}}{2} = \frac{1}{2} \cdot 2^{\frac{x}{2}}$$



Nebenrechnung ( $n! = (n-1)! \cdot n$ ):

$$x \cdot \frac{1}{2} \cdot \left(\frac{x}{2} - 1\right)! = \frac{x}{2} \cdot \left(\frac{x}{2} - 1\right)! = \frac{x}{2}!$$

- Fall  $x$  ist ungerade:

Dies benutzen wir nun, um den eigentlichen Beweis zu führen:

$$\begin{aligned} \text{wp}(\text{"Code vor der Schleife"}, I) &\equiv \text{wp}(\text{"df = df * x; x = x - 2;"}, (df \cdot x)!! = n!! \wedge x \geq 0) \\ &\equiv \text{wp}(\text{"df = df * x;"}, (df \cdot (x - 2))!! = n!! \wedge x - 2 \geq 0) \\ &\equiv \text{wp}(\text{"", } (df \cdot x \cdot (x - 2))!! = n!! \wedge x - 2 \geq 0) \\ &\equiv (df \cdot x)!! = n!! \wedge x \geq 2 \\ &\equiv (df \cdot x)!! = n!! \wedge x > 1 \\ &\equiv I \wedge x > 1 \\ &\equiv I \wedge \text{Schleifenbedingung} \end{aligned}$$

- (d) Zeigen Sie formal mittels wp-Kalkül, dass am Ende der Methode die Nachbedingung  $Q$  erfüllt wird.

Lösungsvorschlag

$$\text{z.z. } I \wedge \neg \text{Schleifenbedingung} \Rightarrow \text{wp}(\text{"Code nach der Schleife"}, Q)$$

Wir vereinfachen den Ausdruck  $I \wedge \neg \text{Schleifenbedingung}$ :

$$I \wedge \neg \text{Schleifenbedingung} \equiv I \wedge (x \leq 1) \equiv I \wedge ((x = 0) \vee (x = 1)) \equiv (I \wedge (x = 0)) \vee (I \wedge (x = 1)) \equiv (df \cdot 1 = n!!) \vee (df \cdot 1 = n!!) \equiv df = n!!$$

Damit gilt:

$$\text{wp}(\text{"Code nach der Schleife"}, Q) \equiv \text{wp}(\text{"", } df = n!!) \equiv df = n!! \equiv I \wedge \neg \text{Schleifenbedingung}$$

- (e) Beweisen Sie, dass die Methode immer terminiert. Geben Sie dazu eine Terminierungsfunktion an und begründen Sie kurz ihre Wahl.

Lösungsvorschlag

Sei  $T(x) := x$ .  $T$  ist offenbar ganzzahlig. Da  $x$  in jedem Schleifendurchlauf um 2 verringert wird, ist  $T$  streng monoton fallend. Aus der Schleifeninvariante folgt  $x \geq 0$  und daher ist  $x$  auch nach unten beschränkt. Damit folgt  $I \Rightarrow T \geq 0$  und  $T$  ist eine gültige Terminierungsfunktion.

## 66116 / 2016 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1

Für die bayerische Forstverwaltung wird eine Datenbank zur Erschließung einer Jagd-Statistik benötigt. Gehen Sie dabei von folgendem Szenario aus:

- Die Administration von Jagdgebieten obliegt den Landkreisen. Jeder **Landkreis** besitzt, neben seinem *Namen* (LName) und der *Einwohnerzahl*, ein eindeutiges *KFZ-Kennzeichen* (KFZKennzeichen).

☐ E: Landkreis

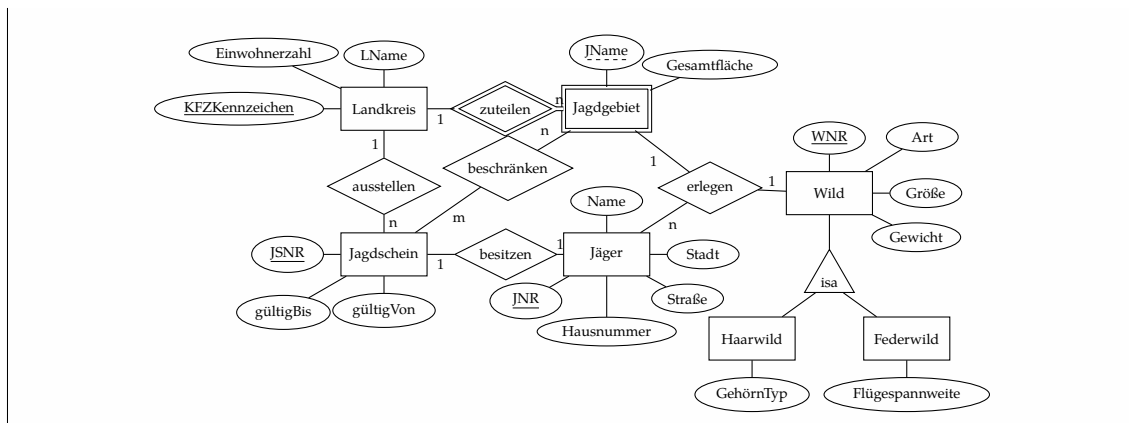
☐ A: Namen

☐ A: Einwohnerzahl

☐ A: KFZ-Kennzeichen

- Die Jagd findet in Jagdgebieten statt. Ein **Jagdgebiet** soll dem Landkreis zugeteilt werden, indem es liegt. Gehen Sie davon aus, dass Jagdgebiete nicht in mehreren Landkreisen liegen können. Zusätzlich ist für jedes Jagdgebiet der *Name* (JName) und die *Gesamtfläche* zu speichern. Dabei ist zu beachten, dass die Namen nur innerhalb eines einzelnen Landkreises eindeutig sind.
    - ☐ E: Jagdgebiet
    - ☒ R: zugeteilt
    - ☐ A: Name
    - ☐ A: Gesamtfläche
  - Die Erlaubnis zum Jagen wird durch einen **Jagdschein** erteilt. Dieser kann nur von einem Landkreis ausgestellt werden und beschränkt sich auf ein oder mehrere Jagdgebiete. Er wird durch eine *Jagdschein-Nummer* (JSNR) identifiziert und ist in einem bestimmtem Zeitintervall gültig. Dieses soll über zwei Zeitpunkte festgelegt werden (*gültig von* (gültigVon), *gültig bis* (gültigBis)).
    - ☐ E: Jagdschein
    - ☒ R: ausgestellt
    - ☒ R: beschränkt
    - ☐ A: Jagdschein-Nummer
    - ☐ A: gültig von
    - ☐ A: gültig bis
  - Ein **Jäger** besitzt genau einen Jagdschein. Zu einem Jäger sollen *Name*, *Stadt*, *Straße* und *Hausnummer*, gespeichert werden. Da die Jagdtradition innerhalb einer Familie häufig von einer zur nächsten Generation weitergegeben wird, kann es vorkommen, dass Name und Adresse von zwei unterschiedlichen Jägern gleich ist (z. B. Vater und Sohn). Aus diesem Grund ist eine eindeutige *Identifikationsnummer* (JNR) notwendig.
    - ☐ E: Jäger
    - ☒ R: besitzt
    - ☐ A: Name
    - ☐ A: Stadt
    - ☐ A: Straße
    - ☐ A: Hausnummer
    - ☐ A: Identifikationsnummer
  - Um Statistiken erheben zu können, muss berücksichtigt werden, welches **Wild** von welchen Jägern zu welchem Zeitpunkt in welchem Jagdgebiet erlegt worden ist. Gehen Sie davon aus, dass es mehrere Jäger geben kann, die gemeinsam ein Wild erlegen (z. B. in einer Jagdgesellschaft). Zu einem Wild gehört die *Art* (z. B. Reh), die *Größe*, das *Gewicht*, sowie eine eindeutige *Identifikationsnummer* (WNR). Zusätzlich unterscheidet man zwischen **Haarwild** und **Federwild**, wobei beim Haarwild der *Typ des Gehörns* (GehörnTyp) (z. B. Hirschgeweih) und beim Federwild die *Flügelspannweite* betrachtet werden soll.
    - ☐ E: Wild
    - ☒ R: erlegt
    - ☐ A: Art
    - ☐ A: Größe
    - ☐ A: Gewicht
    - ☐ A: Identifikationsnummer
    - ☐ E: Haarwild
    - ☐ E: Federwild
    - ☐ A: Typ des Gehörns
    - ☐ A: Flügelspannweite
- (a) Entwerfen Sie für das beschriebene Szenario ein ER-Modell in Chen-Notation. Bestimmen Sie hierzu:
- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
  - die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
  - die Funktionalitäten der Relationship-Typen.

Lösungsvorschlag



- (b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

Lösungsvorschlag

```

Landkreis(KFZKennzeichen, LName, Einwohnerzahl)

Jagdgebiet(JName, KFZKennzeichen[Landkreis], Gesamtfläche)

Jagdschein(JSNR, KFZKennzeichen[Landkreis], gültigVon, gültigBis)

Jäger(JNR, JSNR, Name, Stadt, Straße, Hausnummer)

Wild(WNR, Art, Größe, Gewicht)

Haarwild(WNR, GehörnTyp)

Federwild(WNR, Flügelspannweite)

erlegen(JNR[Jäger], WNR[Wild], JName[Jagdgebiet], KFZKennzeichen[Landkreis])

beschränken(JSNR[Jagdschein], JName[Jagdgebiet], KFZKennzeichen[Landkreis])

```

## 66116 / 2016 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Gehen Sie dabei von dem dazugehörigen relationalen Schema aus:

Polizist : {[ PersNr, DSID, Vorname, Nachname, Dienstgrad, Gehalt ]}

Dienststelle : {[ DSID, Name, Strasse, HausNr, Stadt ]}

Fall : {[ AkZ, Titel, Beschreibung, Status ]}

Arbeitet\_An : {[ PersNr, AkZ, Von, Bis ]}

Vorgesetzte : {[ PersNr, PersNr, Vorgesetzter ]}

Gegeben sei folgendes ER-Modell, welches Polizisten, deren Dienststelle und Fälle, an denen sie arbeiten, speichert:

- (a) Formulieren Sie eine Anfrage in relationaler Algebra, welche den *Vornamen* und *Nachnamen* von Polizisten zurückgibt, deren Dienstgrad „Polizeikommissar“ ist und die mehr als 1500 Euro verdienen.

Lösungsvorschlag

$$\pi_{\text{Vorname, Nachname}}(\sigma_{\text{Dienstgrad}='Polizeikommissar' \wedge \text{Gehalt} > 1500}(\text{Polizist}))$$

- (b) Formulieren Sie eine Anfrage in relationaler Algebra, welche die *Titel* der *Fälle* ausgibt, die von *Polizisten* mit dem *Nachnamen* „Mayer“ bearbeitet wurden.

Lösungsvorschlag

$$\pi_{\text{Titel}}(\sigma_{\text{Nachname}='Mayer'}(\text{Polizist}) \bowtie_{\text{PersNr}} \text{Arbeitet\_An} \bowtie_{\text{AkZ}} \text{Fall})$$

- (c) Formulieren Sie eine SQL-Anfrage, welche die Anzahl der Polizisten ausgibt, die in der Stadt „München“ arbeiten und mit Nachnamen „Schmidt“ heißen.

Lösungsvorschlag

```
SELECT COUNT(*) AS Anzahl_Polizisten
FROM Polizist p, Dienststelle d
WHERE
 p.DSID = d.DSID AND
 d.Stadt = 'München' AND
 p.Nachname = 'Schmidt';
```

```
anzahl_polizisten

 1
(1 row)
```

- (d) Formulieren Sie eine SQL-Anfrage, welche die *Namen* der *Dienststellen* ausgibt, die am 14.02.2012 an dem Fall mit dem XZ1508 beteiligt waren. Ordnen Sie die Ergebnismenge alphabetisch (aufsteigend) und achten Sie darauf, dass keine Duplikate enthalten sind.

Lösungsvorschlag

```
SELECT DISTINCT d.Name
FROM Dienststelle d, Polizist p, Arbeitet_An a
WHERE
 a.AkZ = 'XZ1508' AND
 p.PersNr = a.PersNr AND
 p.DSID = d.DSID AND
 a.Von <= '2012-02-14' AND
 a.Bis >= '2012-02-14'
ORDER BY d.Name ASC;
```

```
name
```

```

Dienststelle Nürnberg (Mitte)
(1 row)
```

VIEW  
WITH  
UNION

- (e) Definieren Sie die View „*Erstrebenswerte Dienstgrade*“, welche Dienstgrade enthalten soll, die in *München* mit durchschnittlich mehr als 2500 Euro besoldet werden.

Lösungsvorschlag

```
CREATE VIEW ErstrebenswerteDienstgrade AS (
 SELECT DISTINCT p.Dienstgrad
 FROM Polizist p, Dienststelle d
 WHERE
 p.DSID = d.DSID AND
 d.Stadt = 'München'
 GROUP BY Dienstgrad
 HAVING (AVG(Gehalt) > 2500)
);
```

```
SELECT * FROM ErstrebenswerteDienstgrade;
```

```

dienstgrad

Polizeikommissar
Polizeimeister
(2 rows)
```

- (f) Formulieren Sie eine SQL-Anfrage, welche *Vorname*, *Nachname* und *Dienstgrad* von *Polizisten* mit *Vorname*, *Nachname* und *Dienstgrad* ihrer *Vorgesetzten* als ein Ergebnistupel ausgibt (siehe Beispiel-Tabelle). Dabei sind nur *Polizisten* zu selektieren, die an Fällen gearbeitet haben, deren Titel den Ausdruck „Fussball“ beinhalten. An *Vorgesetzte* sind keine Bedingungen gebunden. Achten Sie darauf, dass Sie nicht nur direkte Vorgesetzte, sondern alle Vorgesetzte innerhalb der Vorgesetzten-Hierarchie betrachten. Ordnen Sie ihre Ergebnismenge alphabetisch (absteigend) nach Nachnamen des Polizisten.

Hinweis: Sie dürfen Views verwenden, um Teilergebnisse auszudrücken.

Lösungsvorschlag

Vorarbeiten:

```
SELECT p.Vorname, p.Nachname
FROM Polizist p, Arbeitet_An a, Fall f
WHERE
 p.PersNr = a.PersNr AND
 a.AkZ = f.Akz AND
 f.Titel LIKE '%Fussball%';
```

```

vorname | nachname
-----+-----
Hans | Müller
```

Josef | Fischer  
(2 rows)

## Lösungsansatz 1

```
WITH RECURSIVE Fussball_Vorgesetzte (PersNr, VN, NN, DG, PN_VG, VN_VG, NN_VG,
→ DG_VG) AS
(
 SELECT
 p1.PersNr,
 p1.Vorname AS VN,
 p1.Nachname AS NN,
 p1.Dienstgrad AS DG,
 p2.PersNr AS PN_VG,
 p2.Vorname AS VN_VG,
 p2.Nachname AS NN_VG,
 p2.Dienstgrad AS DG_VG
 FROM Polizist p1, Fall f, Arbeitet_An a, Vorgesetzte v
 LEFT JOIN Polizist p2 ON v.PersNr_Vorgesetzter = p2.PersNr
 WHERE
 p1.PersNr = a.PersNr AND
 a.AkZ = f.Akz AND
 f.Titel LIKE '%Fussball%' AND
 p1.PersNr = v.PersNr

 UNION ALL

 SELECT
 m.PersNr,
 m.VN AS VN,
 m.NN AS NN,
 m.DG AS DG,
 p.PersNr AS PN_VG,
 p.Vorname AS VN_VG,
 p.Nachname AS NN_VG,
 p.Dienstgrad AS DG_VG
 FROM Fussball_Vorgesetzte m, Vorgesetzte v
 LEFT JOIN Polizist p ON v.PersNr_Vorgesetzter = p.PersNr
 WHERE m.PN_VG = v.PersNr
)

SELECT VN, NN, DG, VN_VG, NN_VG, DG_VG
FROM Fussball_Vorgesetzte
ORDER BY NN DESC;
```

| vn    | nn      | dg                  | vn_vg   | nn_vg    | dg_vg            |
|-------|---------|---------------------|---------|----------|------------------|
| Hans  | Müller  | Polizeimeister      | Andreas | Schmidt  | Polizeikommissar |
| Hans  | Müller  | Polizeimeister      | Stefan  | Hoffmann | Polizeidirektor  |
| Josef | Fischer | Polizeihauptmeister | Stefan  | Hoffmann | Polizeidirektor  |

Josef | Fischer | Polizeihauptmeister | Sebastian | Wagner | Polizeioberkommissar  
(4 rows)

## Lösungsansatz 2

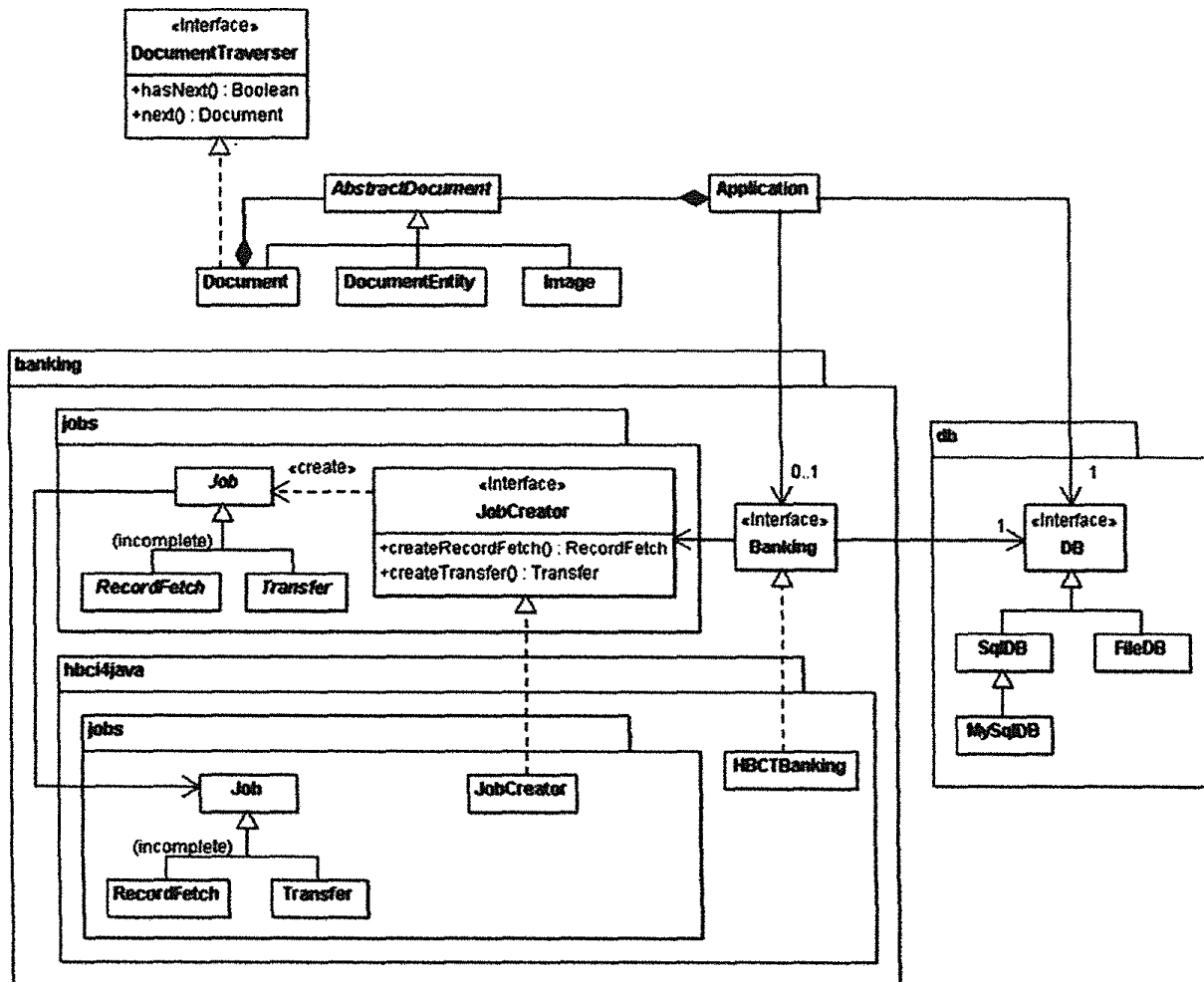
```
CREATE VIEW naechste_Vorgesetzte AS
SELECT
 p.PersNr,
 p.Vorname,
 p.Nachname,
 p.Dienstgrad,
 v.PersNr_Vorgesetzter AS Vorgesetzter
FROM Polizist p LEFT JOIN Vorgesetzte v
ON p.PersNr = v.PersNr;

WITH RECURSIVE Fussball_Vorgesetzte (VN, NN, DG, VN_VG, NN_VG, DG_VG) AS (
 SELECT
 x.Vorname AS VN,
 x.Nachname AS NN,
 x.Dienstgrad AS DG,
 y.Vorname AS VN_VG,
 y.Nachname AS NN_VG,
 y.Dienstgrad AS DG_VG
 FROM naechste_Vorgesetzte x, Fall f, Arbeitet_An a,
 naechste_Vorgesetzte y
 WHERE
 f.Titel LIKE '%Fussball%' AND
 f.AkZ = a.AkZ AND
 x.PersNr = a.PersNr AND
 x.Vorgesetzter = y.PersNr
 UNION ALL
 SELECT
 a.Vorname AS VN,
 a.Nachname AS NN,
 a.Dienstgrad AS DB,
 Vorname AS VN_VG,
 Nachname AS NN_VG,
 Dienstgrad AS DG_VG
 FROM naechste_Vorgesetzte a INNER JOIN Fussball_Vorgesetzte
 ON a.Vorgesetzter = PersNr
)

SELECT *
FROM Fussball_Vorgesetzte;
```

| vn    | nn      | dg                  | vn_vg   | nn_vg    | dg_vg            |
|-------|---------|---------------------|---------|----------|------------------|
| Hans  | Müller  | Polizeimeister      | Andreas | Schmidt  | Polizeikommissar |
| Hans  | Müller  | Polizeimeister      | Stefan  | Hoffmann | Polizeidirektor  |
| Josef | Fischer | Polizeihauptmeister | Stefan  | Hoffmann | Polizeidirektor  |

66116 / 2016 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2



- (a) Kennzeichnen Sie im folgenden Klassendiagramm die Entwurfsmuster „*Abstrakte Fabrik*“, „*Iterator*“, „*Adapter*“ und „*Kompositum*“. Geben Sie die jeweils beteiligten Klassen und deren Zuständigkeit im entsprechenden Muster an.

Lösungsvorschlag

## Iterator

**DocumentTraverser (interface)** Schnittstelle zur Traversierung und zum Zugriff auf Dokumente

**Document** implementiert die Schnittstelle

# Kompositum

**AbstractDocument** abstrakte Basisklasse, die gemeinsames Verhalten der beteiligten Klassen definiert

**Document** enthält wiederum weitere Dokumente bzw. DocumentEnti-



|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| <p>ties und Images</p> <p><b>DocumentEntity, Image</b> primitive Unterklassen, besitzen keine Kind-objekte</p> <p><b>Adapter (Objektadapter)</b></p> <p><b>Banking (interface)</b> vom Client (hier Application) verwendete Schnittstelle</p> <p><b>HBCTBanking</b> passt Schnittstelle der unpassenden Klasse an Zielschnittstelle (Banking) an</p> <p><b>DB (interface)</b> anzupassende Schnittstelle</p> <p><b>abstrakte Fabrik</b></p> <p><b>JobCreator (interface)</b> abstrakte Fabrik</p> <p><b>Job (abstrakt) mit Unterklassen RecordFetch und Transfer</b> abstraktes Produkt</p> <p><b>Job (konkret) mit Unterklassen</b> konkretes Produkt</p> | <p>Klassenadapter<br/>Objektadapter<br/>Implementierung in Java<br/>Gantt-Diagramm</p> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|

- (b) (i) Beschreiben Sie die Funktionsweise der folgenden Entwurfsmuster und geben Sie ein passendes UML-Diagramm an.
- Dekorierer
  - Klassenadapter
  - Objektadapter
- (ii) Erklären Sie mit maximal zwei Sätzen den Unterschied zwischen Klassenadapter und Objektadapter.
- (c) Implementieren Sie einen Stapel in der Programmiersprache Java. Nutzen Sie dazu ein Array mit fester Größe. Auf eine Überlaufprüfung darf verzichtet werden. Implementieren Sie in der Klasse das Iterator Entwurfsmuster, um auf die Inhalte zuzugreifen, sowie eine Funktion zum Hinzufügen von Elementen. Als Typ für den Stapel kann zur Vereinfachung ein Integertyp verwendet werden.

## 66116 / 2016 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 4

- (a) Erklären Sie in maximal zwei Sätzen den Unterschied zwischen Knoten- und Kanten Netzwerken im Kontext des Projektmanagements.
- (b) Gegeben ist die folgende Tabelle zur Grobplanung eines hypothetischen Softwareprojekts:

| Aktivität           | Minimale Dauer | Einschränkungen                                                                                                                     |
|---------------------|----------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Anforderungsanalyse | 2 Monate       | Endet frühestens einen Monat nach dem Start der Entwurfsphase.                                                                      |
| Entwurf             | 4 Monate       | Startet frühestens zwei Monate nach dem Start der Anforderungsanalyse.                                                              |
| Implementierung     | 5 Monate       | Endet frühestens drei Monate nach dem Ende der Entwurfsphase. Darf erst starten, nachdem die Anforderungsanalyse abgeschlossen ist. |

Geben Sie ein CPM-Netzwerkan, das die Aktivitäten und Abhängigkeit des obigen Projektplans beschreibt. Gehen Sie von der Zeiteinheit „Monate“ aus. Das Projekt hat einen Start- und einen Endknoten.

Jede Aktivität wird auf einen Start- und einen Endknoten abgebildet. Die Dauer der Aktivitäten sowie Abhängigkeiten sollen durch Kanten dargestellt werden. Der Start jeder Aktivität hängt vom Projektstart ab, das Projektende hängt vom Ende aller Aktivitäten ab. Modellieren Sie diese Abhängigkeiten durch Pseudoaktivitäten mit Dauer null.

- (c) Berechnen Sie für jedes Ereignis (für jeden Knoten) die früheste Zeit sowie die späteste Zeit. Beachten Sie, dass die Berechnungsreihenfolge einer topologischen Sortierung des Netzwerks entsprechen sollte.
- (d) Geben Sie einen kritischen Pfad durch das CPM-Netzwerk an. Möglicherweise sind hierfür weitere Vorberechnungen vonnöten. Welche Aktivität sollte sich demnach auf keinen Fall verzögern?
- (e) Geben Sie ein Gantt-Diagramm an, das den Projektplan visualisiert. Gehen Sie davon aus, dass jede Aktivität zur frühesten Zeit ihres Startknotens beginnt und zur spätesten Zeit ihres Endknotens endet (s. jeweils Teilaufgabe (c)). Geben Sie die minimale Dauer jeder Aktivität, sowie die Pufferzeit (in Klammern) an. Beispiel: 4 (+2). Notieren Sie alle Einschränkungen mit Hilfe geeigneter Abhängigkeitsbeziehungen. Geben Sie eine absolute Zeitskala in Monaten an.
- (f) Nennen Sie zwei weitere Aktivitäten, die in der obigen Tabelle fehlen, jedoch typischerweise in Softwareentwicklungs-Prozessmodellen wie etwa dem Wasserfallmodell vorkommen.

## 66116 / 2016 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4

Gegeben sind folgende Relationen aus einer Personalverwaltung:

```
Mitarbeiter : {[MitarbeiterID, Vorname, Nachname, Vorgesetzter[Mitarbeiter], AbteilungsID[Abteilung], Telefonnummer, Gehalt]}
```

Abteilung : {[ AbteilungsID, Bezeichnung ]}

- (a) Schreiben Sie eine SQL-Anfrage, die Vor- und Nachnamen der Mitarbeiter aller Abteilungen mit der Bezeichnung „Buchhaltung“ ausgibt, absteigend sortiert nach Mitarbeiter-ID.

Lösungsvorschlag

```
SELECT Vorname, Nachname
FROM Mitarbeiter m, Abteilung a
WHERE
 m.AbteilungsID = a.AbteilungsID AND
 a.Bezeichnung = 'Buchhaltung'
ORDER BY m.MitarbeiterID DESC;
```

```
vorname | nachname
-----+-----
Till | Fuchs
Hans | Meier
(2 rows)
```

- (b) Schreiben Sie eine SQL-Anfrage, die die Nachnamen aller Mitarbeiter mit dem Nachnamen ihres jeweiligen direkten Vorgesetzten ausgibt. Mitarbeiter ohne Vorgesetzten sollen in der Ausgabe ebenfalls enthalten sein. In diesem Fall soll der Nachname des Vorgesetzten NULL sein.

Lösungsvorschlag

```
SELECT m.Nachname AS Mitarbeiter, v.Nachname AS Vorgesetzter
FROM Mitarbeiter m LEFT OUTER JOIN Mitarbeiter v
ON m.Vorgesetzter = v.MitarbeiterID;
```

```
mitarbeiter | vorgesetzter
-----+-----
Meier | Müller
Wolitz | Müller
Müller |
Fuchs | Wolitz
Hase | Müller
Navratil |
Schmidt | Navratil
(7 rows)
```

- (c) Schreiben Sie eine SQL-Anfrage, die die 10 Abteilungen ausgibt, deren Mitarbeiter das höchste Durchschnittsgehalt haben. Ausgegeben werden sollen der Rang (1 = höchstes Durchschnittsgehalt bis 10 = niedrigstes Durchschnittsgehalt), die Bezeichnung sowie das Durchschnittsgehalt der Abteilung. Gehen Sie davon dass es keine zwei Abteilungen mit gleichem Durchschnittsgehalt gibt. Sie können der Übersichtlichkeit halber Views oder With-Anweisungen verwenden. Verwenden Sie jedoch keine datenbanksystemspezifischen Erweiterungen wie limit oder rownum.

```

CREATE VIEW Durchschnittsgehälter AS
SELECT Abteilung.AbteilungsID, Bezeichnung,
 AVG (Gehalt) AS Durchschnittsgehalt
FROM Mitarbeiter, Abteilung
WHERE Mitarbeiter.AbteilungsID = Abteilung.AbteilungsID
GROUP BY Abteilung.AbteilungsID, Bezeichnung;

SELECT a.Bezeichnung, a.Durchschnittsgehalt, COUNT (*) AS Rang
FROM Durchschnittsgehälter a, Durchschnittsgehälter b
WHERE a.Durchschnittsgehalt <= b.Durchschnittsgehalt
GROUP BY a.AbteilungsID, a.Bezeichnung, a.Durchschnittsgehalt
HAVING COUNT(*) <= 10
ORDER BY Rang ASC;

```

| bezeichnung | durchschnittsgehalt | rang |
|-------------|---------------------|------|
| Managment   | 6514.5              | 1    |
| Buchhaltung | 2340                | 2    |
| Vertrieb    | 1283.5              | 3    |
| Produktion  | 654                 | 4    |

(4 rows)

- (d) Schreiben Sie eine SQL-Anfrage, die das Gehalt aller Mitarbeiter aus der Abteilung mit der AbteilungsID 42 um 5% erhöht.

| vorname | nachname | gehalt |
|---------|----------|--------|
| Lea     | Müller   | 5875   |
| Gerd    | Navratil | 7154   |

(2 rows)

```

SELECT Vorname, Nachname, Gehalt
FROM MITARBEITER
WHERE AbteilungsId = 42
ORDER BY Gehalt;

```

```

UPDATE Mitarbeiter
SET Gehalt = 1.05 * Gehalt
WHERE AbteilungsID = 42;

```

```

SELECT Vorname, Nachname, Gehalt
FROM MITARBEITER
WHERE AbteilungsId = 42
ORDER BY Gehalt;

```

| vorname | nachname | gehalt  |
|---------|----------|---------|
| Lea     | Müller   | 6168.75 |

|          |  |          |  |                    |
|----------|--|----------|--|--------------------|
| Gerd     |  | Navratil |  | 7511.7000000000001 |
| (2 rows) |  |          |  |                    |

DELETE

- (e) Alle *Abteilungen* mit Bezeichnung „Qualitätskontrolle“ sollen zusammen mit den Datensätzen ihrer *Mitarbeiter* gelöscht werden. ON DELETE CASCADE ist für keine der Tabellen gesetzt. Schreiben Sie die zum Löschen notwendigen SQL-Anfragen.

Lösungsvorschlag

| vorname |  | nachname |
|---------|--|----------|
| Hans    |  | Meier    |
| Fred    |  | Wolitz   |
| Lea     |  | Müller   |
| Till    |  | Fuchs    |
| Fred    |  | Hase     |
| Gerd    |  | Navratil |
| Jürgen  |  | Schmidt  |

(7 rows)

| abteilungsid |  | bezeichnung        |
|--------------|--|--------------------|
| 1            |  | Buchhaltung        |
| 2            |  | Vertrieb           |
| 42           |  | Managment          |
| 4            |  | Qualitätskontrolle |
| 5            |  | Produktion         |

(5 rows)

```

SELECT Vorname, Nachname FROM Mitarbeiter;
SELECT * FROM Abteilung;

DELETE FROM Mitarbeiter
WHERE AbteilungsID IN (
 SELECT a.AbteilungsID
 FROM Abteilung a
 WHERE a.Bezeichnung = 'Qualitätskontrolle'
);

DELETE FROM Abteilung
WHERE Bezeichnung = 'Qualitätskontrolle';

SELECT Vorname, Nachname FROM Mitarbeiter;
SELECT * FROM Abteilung;

```

| vorname |  | nachname |
|---------|--|----------|
| Fred    |  | Wolitz   |
| Lea     |  | Müller   |
| Till    |  | Fuchs    |

|              |  |             |  |
|--------------|--|-------------|--|
| Gerd         |  | Navratil    |  |
| Jürgen       |  | Schmidt     |  |
| (5 rows)     |  |             |  |
|              |  |             |  |
| abteilungsid |  | bezeichnung |  |
| -----+-----  |  |             |  |
| 1            |  | Buchhaltung |  |
| 2            |  | Vertrieb    |  |
| 42           |  | Managment   |  |
| 5            |  | Produktion  |  |
| (4 rows)     |  |             |  |

Tupel-Identifikator  
B-Baum

- (f) Alle Mitarbeiter sollen mit SQL-Anfragen nach den Telefonnummern anderer Mitarbeiter suchen können. Sie dürfen jedoch das Gehalt der Mitarbeiter nicht sehen können. Erläutern Sie in zwei bis drei Sätzen eine Möglichkeit, wie dies in einem Datenbanksystem realisiert werden kann, ohne die gegebenen Relationen, die Tabellen als abgelegt sind, zu verändern. Sie brauchen hierzu keinen SQL-Code schreiben.

Lösungsvorschlag

Wir könnten eine VIEW erstellen, die zwar Namen und ID der anderen Mitarbeiter, sowie ihre Telefonnummern enthält (evtl. auch Abteilungsbezeichnung und ID), aber eben nicht das Gehalt: Mitarbeiter arbeiten auf eingeschränkter Sicht.

Alternativ mit GRANT:

explizit mit SELECT die Spalten auswählen, die man lesen können soll (auf nicht angegebene Spalten ist kein Zugriff möglich)

```
GRANT SELECT (Vorname, Nachname, Telefonnummer)
ON Mitarbeiter TO postgres;
```

## 66116 / 2016 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 5

- (a) Erläutern Sie die wesentliche Eigenschaft eines Tupel-Identifikators (TID) in ein bis zwei Sätzen.

Lösungsvorschlag

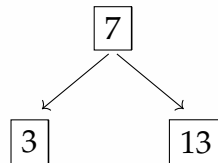
- Daten werden in Form von *Sätzen* auf der Festplatte abgelegt, um auf Sätze zugreifen zu können, verfügt jeder Satz über eine *eindeutige, unveränderliche Satzadresse*
- TID = Tupel Identifier: dient zur Adressierung von Sätzen in einem Segment und besteht aus zwei Komponenten:
  - Seitennummer (Seiten bzw. Blöcke sind größere Speichereinheiten auf der Platte)
  - Relative Indexposition innerhalb der Seite

- Satzverschiebung innerhalb einer Seite bleibt ohne Auswirkungen auf den TID. Wird ein Satz auf eine andere Seite migriert, wird eine „Stellvertreter-TID“ zum Verweis auf den neuen Speicherort verwendet. Die eigentliche TID-Adresse bleibt stabil.

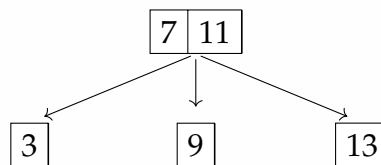
- (b) Fügen Sie in einen anfangs leeren B-Baum mit  $k = 1$  (maximal 2 Schlüsselwerte pro Knoten) die im Folgenden gegebenen Schlüsselwerte der Reihe nach ein. Zeichnen Sie den Endzustand des Baums nach jedem Einfügevorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie die sieben Endzustände deutlich.
- 3, 7, 13, 11, 9, 10, 8

Lösungsvorschlag

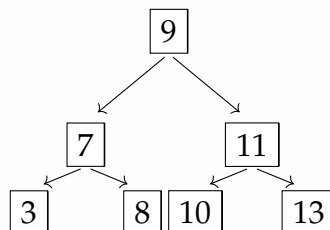
- 3 (einfaches Einfügen)
- 7 (einfaches Einfügen)
- 13 (Split)



- 11 (einfaches Einfügen)
- 9 (Split)



- 10 (einfaches Einfügen)
- 8 (Doppel-Split)



- (c) Gegeben ist der folgende B-Baum:

Die folgenden Teilaufgaben sind voneinander unabhängig.

- (i) Löschen Sie aus dem gegebenen B-Baum den Schlüssel 3 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

- (ii) Löschen Sie aus dem (originalen) gegebenen B-Baum den Schlüssel 17 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.
- (iii) Löschen Sie aus dem (originalen) gegebenen B-Baum den Schlüssel 43 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

## 66116 / 2016 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 1

- (a) Welche Kriterien werden bei der Zielbestimmung im Pflichtenheft unterschieden?

Lösungsvorschlag

Im Rahmen der Zielbestimmung werden die Ziele des Produktes in einer Art Prioritätenliste exakt eingegrenzt, die in drei weitere Kategorien gegliedert ist, die als **Muss-**, **Wunsch-** und **Abgrenzungskriterien** bezeichnet werden.

**Musskriterien** Zu den Musskriterien zählen die Produktfunktionen, die für ein Funktionieren des Produktes unabdingbar sind. Ohne ihre Umsetzung geht einfach nichts, deshalb müssen sie erfüllt werden.

**Wunschkriterien** Die nächste Kategorie, die Wunschkriterien, sind zwar entbehrlich, wurden aber ausdrücklich vom Auftraggeber gefordert. Ihre Umsetzung im Produkt ist also genauso eine Pflicht, wie die der ersten Kategorie, das Produkt würde aber auch ohne ihre Implementierung seinen Betrieb korrekt ausführen.

**Abgrenzungskriterien** Die letzte Kategorie beschreiben die Abgrenzungskriterien. Sie sollen die Grenzen des Produktes klarmachen und sind von daher beinahe wichtiger als die beiden ersten Fälle denn sie hindern die Entwickler daran, über alle Ziele hinauszuschießen.

<https://st.inf.tu-dresden.de/SalesPoint/v3.2/tutorial/stepbystep2/pflh.html>

- (b) Nennen Sie drei Einsatzziele von Softwaremaßen.

Lösungsvorschlag

Aus Baumann K. (1997) Softwaremaße. In: Unterstützung der objektorientierten Systemanalyse durch Softwaremaße. Beiträge zur Wirtschaftsinformatik, vol 23. Physica, Heidelberg. [https://doi.org/10.1007/978-3-662-13273-9\\_2](https://doi.org/10.1007/978-3-662-13273-9_2)

Der Einsatz von Softwaremaßen kann vielfältige Vorteile mit sich bringen. Folgende Ziele können mit der Anwendung von Softwaremaßen verfolgt werden:

- Durch eine Bewertung kann überprüft werden, inwieweit einmal gestellte *Qualitätsanforderungen* erfüllt wurden.



- Eine Bewertung bereits erstellter oder noch zu erstellender Softwareprodukte kann Hilfestellung bei Entscheidungen, die für die *Projektplanung* oder das Entwicklungsmanagement zu treffen sind, leisten. So können Softwaremaße zur Einschätzung des notwendigen Aufwands, der zu erbringenden Kosten und des Personalbedarfs herangezogen werden.
- Ein möglichst frühzeitiges *Aufzeigen von Schwachstellen im Systemdesign*, die Beurteilung der Auswirkungen neuer Techniken oder Tools auf die Produktivität der Entwickler oder auf die Qualität des entwickelten Produkts sowie die Überprüfung der Einhaltung festgelegter Designstandards sind weitere mögliche Einsatzfelder für Softwaremaße.
- Darüber hinaus ist der Einsatz von Softwaremaßen *Teil umfassender Konzepte zum Softwarequalitätsmanagement*. Softwaremaße bzw. die zugehörige Meßergebnisse tragen zunächst dazu bei, die betrachteten Produkte oder Prozesse besser zu verstehen oder hinsichtlich interessierender Eigenschaften zu bewerten.

[https://link.springer.com/chapter/10.1007%2F978-3-662-13273-9\\_2](https://link.springer.com/chapter/10.1007%2F978-3-662-13273-9_2)

(c) Welche Arten von Softwaremaßen werden unterschieden?

Lösungsvorschlag

Die Unterscheidung nach dem Messgegenstand ergibt folgende Arten von Maßen:

**Externe Produktmaße.** Diese beschreiben Merkmale des Produkts, die von außen sichtbar sind, wenngleich eventuell nur indirekt. Dazu gehören insbesondere Maße für Qualitätsattribute wie die Wartbarkeit, die Zuverlässigkeit, die Benutzbarkeit, die Effizienz etc.

**Interne Produktmaße.** Basis für die Charakterisierung externer Produktattribute sind interne Attribute, die sich zumindest zum Teil besser messen lassen. Hierzu gehören z. B. die Größe, die Modularität und die Komplexität.

**Prozessmaße.** Diese charakterisieren Eigenschaften des Produktionsprozesses, z. B. die typische Produktivität, Kostenaufteilung auf bestimmte Arbeiten, Dauer von Arbeitsschritten etc.

**Ressourcenmaße.** Diese charakterisieren Eigenschaften der im Prozess verwendeten Ressourcen, z. B. die Auslastung von Rechnern, die typische Produktivität und Fehlerrate eines bestimmten Ingenieurs etc.

<http://page.mi.fu-berlin.de/prechelt/swt2/node5.html>

Was sind die 3 Arten der (einfachen) Maße zur Messung von Software?

<https://quizlet.com/de/339799466/softwaretechnik-theorie-flash-cards/>

- Größenorientiert/nach Größe

- Funktionsorientiert/nach Funktion
- Objektorientiert/nach Objekt

(d) Nennen Sie drei Merkmale evolutionärer Softwareentwicklung.

Lösungsvorschlag

Wie für jede Form der Software-Entwicklung stellt sich die Frage nach einem geeigneten methodischen Ansatz. Generell lässt sich der Prozess der Software-Entwicklung in die folgenden drei Abschnitte gliedern:

- (i) von der Problembeschreibung bis zur Software-Spezifikation (auch Problemanalyse, Systemanalyse, Requirements Engineering bzw. frühe Phasen der Software-Entwicklung genannt),
- (ii) die Software-Entwicklung im engeren (technischen) Sinn,
- (iii) die Integration der Software in den Anwendungszusammenhang.

Diese Einteilung gilt ebenso für die evolutionäre Software-Entwicklung, allerdings wird hier der Schwerpunkt anders gelegt. Steht beim klassischen Ansatz in der Regel die Software-Entwicklung im engeren Sinn im Mittelpunkt, so werden beim evolutionären Ansatz alle drei Abschnitte möglichst gleichgewichtig und im Zusammenhang betrachtet. Daraus resultiert eine natürliche Tendenz zu einer zyklischen Vorgehensweise.

Daneben gibt es weitere Querbezüge: Orientiert sich die Software-Entwicklung an einem festen Ziel, verfolgt dieses jedoch durch schrittweises Erweitern zunächst unvollkommener Teillösungen, so wird dieses Vorgehen inkrementell genannt. Unfertige Versuchsanordnungen oder Teilsysteme, die am Beginn einer solchen Entwicklung stehen, werden oft als Prototypen bezeichnet. Wählt man Prototyping als Vorgehensweise, so kann entweder das spätere Produkt inkrementell aus einem oder mehreren Prototypen weiterentwickelt werden oder man setzt nach einer (Wegwerf-) Prototypphase neu auf. Beides sind Formen evolutionärer Entwicklung.

<http://waste.informatik.hu-berlin.de/~dahme/edidahm.pdf>

(e) Nennen Sie drei Vorteile des Einsatzes von Versionsverwaltungssoftware.

Lösungsvorschlag

- (i) Möglichkeit, auf ältere Entwicklungsversionen zurückzukehren, wenn sich die aktuelle als nicht lauffähig bzw. unsicher erwiesen hat (z. B. git revert)
- (ii) Möglichkeit, dass mehrere Entwickler gleichzeitig an ein- und demselben Softwareprojekt arbeiten und Möglichkeit, dass ihre Entwicklungen durch das Versionsverwaltungssystem zusammengeführt werden können (z. B. git merge)

- (iii) Möglichkeit, den Zeitpunkt oder den Urheber eines Softwarefehlers ausfindig zu machen (z. B. git blame)

- (f) Worin besteht der Unterschied zwischen funktionalen und nicht-funktionalen Anforderungen?

Lösungsvorschlag

Funktionale Anforderung: Anforderung an die Funktionalität des Systems, also „Was kann es?“

Nicht-funktionale Anforderung: Design, Programmiersprache, Performanz

- (g) Was verbirgt sich hinter dem Begriff „Continuous Integration“?

Lösungsvorschlag

Continuous Integration: Das fertige Modul wird sofort in das bestehende Produkt integriert. Die Integration erfolgt also schrittweise und nicht erst, wenn alle Module fertig sind. Somit können auch neue Funktionalitäten sofort hinzugefügt werden (*rightarrow* neue Programmversion).

## 66116 / 2016 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 2

- (a) Gegeben sei folgende natürlichsprachliche Spezifikation eines PKI-Systems:

*Damit der Schüler seinem Lehrer die Hausaufgaben verschlüsselt per E-Mail übermitteln kann, bedarf es einer entsprechenden Infrastruktur. Nachdem beide Teilnehmer die notwendige Software installiert haben, erstellt der Lehrer zunächst ein sogenanntes Schlüsselpaar, bestehend aus einem „Öffentlichen“ (ÖS) und einem zugehörigen „privaten“ Schlüssel (PS). Anschließend veröffentlicht der Lehrer seinen ÖS durch Hochladen auf einen sogenannten Keyserver (Schlüsselverzeichnisdienst). Damit steht er jedem Schüler zur Verfügung, so dass dieser den ÖS jederzeit vom Keyserver herunterladen kann. Alternativ kann der Lehrer seinen ÖS auch direkt (z. B. per E-Mail oder USB-Stick) an den Schüler übermitteln.*

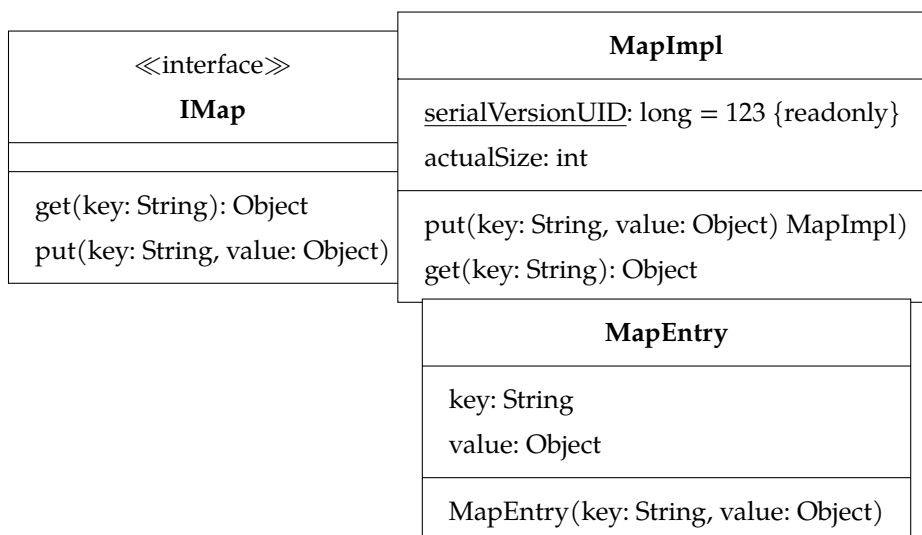
*Der Schüler kann nun seine Nachricht (z. B. seine Lösung) mit dem ÖS des Lehrers verschlüsseln und mit einer E-Mail versenden. Empfängt der Lehrer eine solche E-Mail, so kann er (und nur er) mit seinem PS die Nachricht wieder entschlüsseln. Umgekehrt kann der Lehrer mit seinem PS beliebige Informationen (z. B. die Note) „digital signieren“. Diese unterschriebenen Daten übermittelt der Lehrer dann zusammen mit der digitalen Signatur per E-Mail an den Schüler. Der Schüler kann mit dem ÖS des Lehrers prüfen, ob die übermittelte Nachricht unverändert und tatsächlich vom unterschreibenden Lehrer stammt.*

Modellieren Sie die in der Spezifikation beschriebenen Anwendungsfälle zusammen mit den jeweils beteiligten Akteuren in einem Use-Case-Diagramm. Betrachten Sie den Keyserver zunächst ebenfalls als Akteur, welcher gegenüber PKI als „externer Vermittler“ auftritt.

(b) Erstellen Sie ein geeignetes Klassendiagramm für das obige PKI. Berücksichtigen Sie zusätzlich zur verbalen Spezifikation noch folgende Präzisierungen:

- (i) Die Schlüssel werden mit einer E-Mail-Adresse „benannt“, damit der Schüler den richtigen Schlüssel abrufen kann.
- (ii) Es gibt genau einen Keyserver; dieser verwaltet aber beliebig viele Schlüssel. Er bietet die entsprechenden Dienste zum Veröffentlichen bzw. Abfragen von Schlüsseln an.
- (iii) Jeder Lehrer hat höchstens ein Schlüsselpaar, aber jeder Schlüssel gehört genau einem Lehrer. Der Schüler hingegen kommuniziert mit mehreren Lehrern und kennt daher mehrere E-Mail-Adressen.
- (iv) Eine Nachricht kann (muss aber nicht) eine Signatur oder einen ÖS als „Anhang“ zusätzlich zum eigentlichen Inhalt (zur Vereinfachung: String) mit sich führen.
- (v) Für das Signieren bzw. Entschlüsseln ist der PS (das zugehörige Objekt selbst) zuständig. Dafür bekommt er den Inhalt der Nachricht und gibt entsprechend eine Signatur bzw. den entschlüsselten Inhalt zurück.
- (vi) Für das Prüfen der Signatur und das Verschlüsseln ist der ÖS zuständig. Dazu bekommen die Methoden je nach Bedarf den Inhalt der Nachricht und die Signatur und liefern einen Wahrheitswert bzw. den verschlüsselten Inhalt zurück.

(c) Übertragen Sie folgendes UML-Klassendiagramm in Programm-Code einer gängigen und geeigneten objektorientierten Sprache Ihrer Wahl. Die Methodenrumpfe dürfen Sie dabei leer lassen (auch wenn das Programm dann nicht übersetzbar bzw. ausführbar wäre).



Lösungsvorschlag

Interface „IMap“

```
interface IMap {
 Object get(String key);

 void put(String key, Object value);
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2016/herbst/pki/IMap.java](#)

## Klasse „MapImpl“

```
import java.util.ArrayList;
import java.util.List;

class MapImpl implements IMap {
 final static long serialVersionUID = 123;

 private List<MapEntry> entries;

 MapImpl() {
 entries = new ArrayList<MapEntry>();
 }

 public Object get(String key) {
 return new Object();
 }

 public void put(String key, Object value) {
 // Nicht verlangt in der Aufgabenstellung.
 entries.add(new MapEntry(key, value));
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2016/herbst/pki/MapImpl.java](#)

## Klasse „MapEntry“

```
class MapEntry {
 String key;
 Object value;

 MapEntry(String key, Object value) {
 // Nicht verlangt in der Aufgabenstellung.
 this.key = key;
 this.value = value;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2016/herbst/pki/MapEntry.java](#)

**66116 / 2016 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 3**

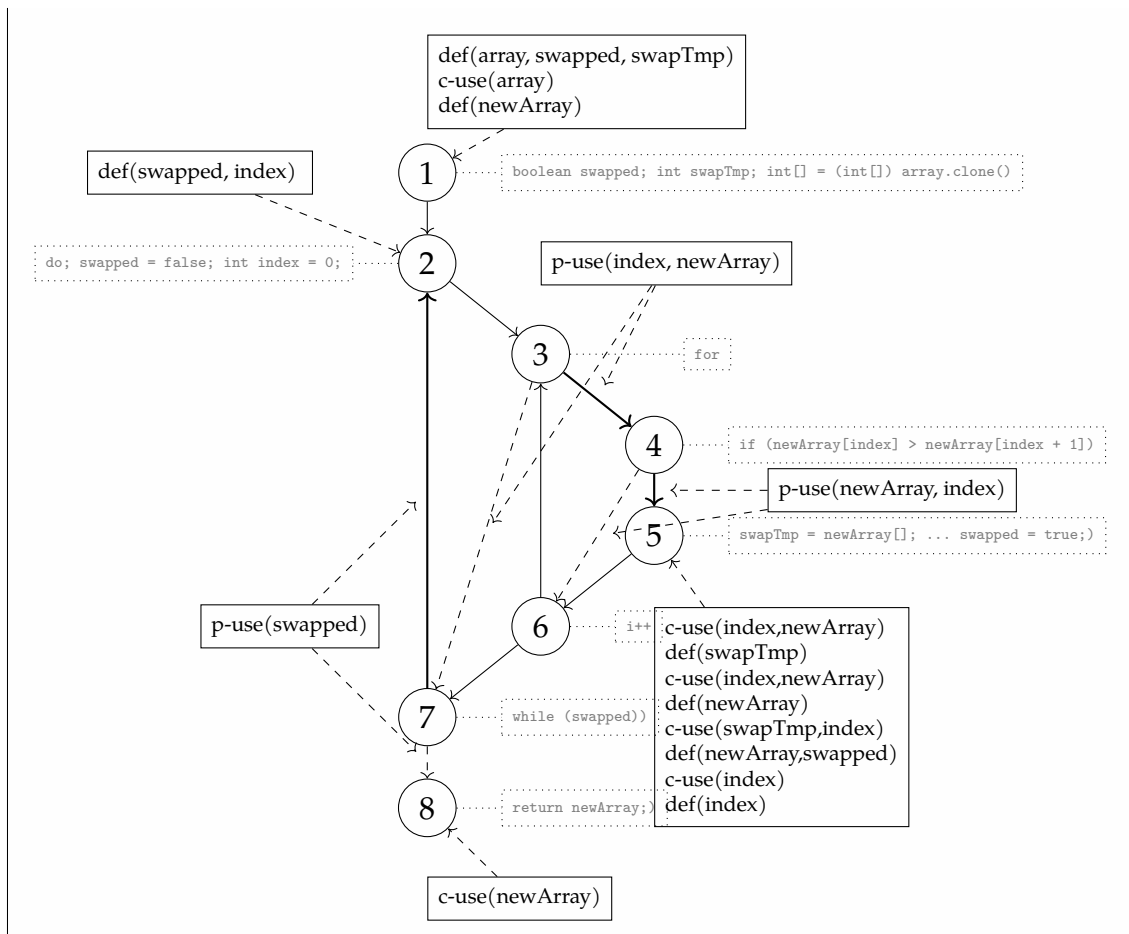
Gegeben Sei folgende Java-Methode `sort` zum Sortieren eines Feldes ganzer Zahlen:

```
public static int[] sort(int[] array) {
 boolean swapped;
 int swapTmp;
 int[] newArray = (int[]) array.clone();
 do {
 swapped = false;
 for (int index = 0; index < newArray.length - 1; index++) {
 if (newArray[index] > newArray[index + 1]) {
 swapTmp = newArray[index];
 newArray[index] = newArray[index + 1];
 newArray[index + 1] = swapTmp;
 swapped = true;
 }
 }
 } while (swapped);
 return newArray;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2016/herbst/BubbleSort.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2016/herbst/BubbleSort.java)

- (a) Konstruieren Sie den Kontrollflussgraphen des obigen Code-Fragments und annotieren Sie an den Knoten und Kanten die zugehörigen Datenflussinformationen (Definitionen bzw. berechnende oder prädikative Verwendung von Variablen).

Lösungsvorschlag



Zyklomatische Komplexität  
nach McCabe  
C1-Test Zweigüberdeckung  
(Branch Coverage)  
all uses

- (b) Nennen Sie die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

Lösungsvorschlag

Der Graph hat 8 Knoten und 10 Kanten. Daher ist die zyklomatische Komplexität nach McCabe gegeben durch  $10 - 8 + 2 = 4$ .

- (c) Geben Sie einen möglichst kleinen Testdatensatz an, der eine 100%-ige Verzweigungsüberdeckung dieses Moduls erzielt.

Lösungsvorschlag

Die Eingabe muss mindestens ein Feld der Länge 3 sein. Ansonsten wäre das Feld schon sortiert bzw. bräuchte nur eine Vertauschung und die innere if-Bedingung wäre nicht zu 100% überdeckt. Daher wählt man beispielsweise `array = [1,3,2]`.

- (d) Beschreiben Sie kurz, welche Eigenschaften eine Testfallmenge allgemein haben muss, damit das datenflussorientierte Überdeckungskriterium „all-uses“ erfüllt.

Das Kriterium all-uses ist das Hauptkriterium des datenflussorientierten Testens, denn es testet den kompletten prädikativen und berechnenden Datenfluss. Konkret: von jedem Knoten mit einem globalen  $\text{def}(x)$  einer Variable  $x$



existiert ein definitions-freier Pfad bzgl.  $x$  ( $\text{def-clear}(x)$ ) zu jedem erreichbaren Knoten mit einem  $\text{c-use}(x)$  oder  $\text{p-use}(x)$ .

## 66116 / 2016 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2

Gegeben sei der folgende Ausschnitt aus dem Schema einer Schulverwaltung:

```
Person : {[
 ID : INTEGER,
 Name : VARCHAR(255),
 Wohnort : VARCHAR(255),
 Typ : CHAR(1)
]}
```

```
Unterricht : {[
 Klassenbezeichnung : VARCHAR(20),
 Schuljahr : INTEGER,
 Lehrer : INTEGER,
 Fach : VARCHAR(100)
]}
```

```
Klasse : {[
 Klassenbezeichnung : VARCHAR(20),
 Schuljahr : INTEGER,
 Klassenlehrer : INTEGER
]}
```

```
Klassenverband : {[
 Schüler : INTEGER,
 Klassenbezeichnung : VARCHAR(20),
 Schuljahr : INTEGER
]}
```

```
CREATE TABLE Person (
 ID INTEGER PRIMARY KEY,
 Name VARCHAR(255),
 Wohnort VARCHAR(255),
 Typ CHAR(1) CHECK(Typ in ('S', 'L'))
);
```

```
CREATE TABLE Klasse (
 Klassenbezeichnung VARCHAR(20),
 Schuljahr INTEGER,
 Klassenlehrer INTEGER REFERENCES Person(ID),
 PRIMARY KEY (Klassenbezeichnung, Schuljahr)
);
```

```
CREATE TABLE Klassenverband (
 Schüler INTEGER REFERENCES Person(ID),
```

```

Klassenbezeichnung VARCHAR(20),
Schuljahr INTEGER,
PRIMARY KEY (Schüler, Schuljahr)
);

CREATE TABLE Unterricht (
 Klassenbezeichnung VARCHAR(20),
 Schuljahr INTEGER,
 Lehrer INTEGER REFERENCES Person(ID),
 Fach VARCHAR(100),
 CONSTRAINT Unterricht_PK
 PRIMARY KEY (Klassenbezeichnung, Schuljahr, Lehrer, Fach)
);

INSERT INTO Person VALUES
(1, 'Lehrer Ludwig', 'München', 'L'),
(2, 'Schüler Max', 'München', 'S'),
(3, 'Schülerin Maria', 'München', 'S'),
(4, 'Schülerin Eva', 'Starnberg', 'S'),
(5, 'Lehrerin Walter', 'München', 'L'),
(6, 'Schüler Karl', 'München', 'S');

INSERT INTO Klasse VALUES
('1a', 2015, 1),
('1a', 2014, 1),
('1b', 2015, 5);

INSERT INTO Klassenverband VALUES
(2, '1a', 2015),
(3, '1a', 2015),
(4, '1b', 2015),
(6, '1a', 2015),
(6, '1a', 2014);

```

Hierbei enthält die Tabelle *Person* Informationen über Lehrer (Typ 'L') und Schüler (Typ 'S'); andere Werte für Typ sind nicht zulässig. *Klasse* beschreibt die Klassen, die in jedem Schuljahr gebildet wurden, zusammen mit ihrem Klassenlehrer. In *Unterricht* wird abgelegt, welcher Lehrer welches Fach in welcher Klasse unterrichtet; es ist möglich, dass derselbe Lehrer mehr als ein Fach in einer Klasse unterrichtet. *Klassenverband* beschreibt die Zuordnung der Schüler zu den Klassen.

- (a) Schreiben Sie eine SQL-Anweisung, die die Tabelle *Unterricht* mit allen ihren Constraints (einschließlich Fremdschlüsselconstraints) anlegt.

Lösungsvorschlag

```

CREATE TABLE IF NOT EXISTS Person(
 ID INTEGER PRIMARY KEY,
 Name VARCHAR(255),
 Wohnort VARCHAR(255),
 Typ CHAR(1) CHECK(Typ in ('S', 'L'))
);

CREATE TABLE IF NOT EXISTS Klasse(
 Klassenbezeichnung VARCHAR(20),

```

```

Schuljahr INTEGER,
Klassenlehrer INTEGER REFERENCES Person(ID),
PRIMARY KEY (Klassenbezeichnung, Schuljahr)
);

CREATE TABLE IF NOT EXISTS Unterricht (
Klassenbezeichnung VARCHAR(20) REFERENCES Klasse(Klassenbezeichnung),
Schuljahr INTEGER REFERENCES Klasse(Schuljahr),
Lehrer INTEGER REFERENCES Person(ID),
Fach VARCHAR(100),
CONSTRAINT Unterricht_PK
PRIMARY KEY (Klassenbezeichnung, Schuljahr, Lehrer, Fach)
);

```

CONSTRAINT  
ALTER TABLE  
GROUP BY

- (b) Definieren Sie ein geeignetes Constraint, das sicherstellt, dass nur zulässige Werte im Attribut Typ der (bereits angelegten) Tabelle *Person* eingefügt werden können.

Lösungsvorschlag

Ich habe REFERENCES bei Unterricht Schuljahr vergessen, die referenzierten Tabellen Person und Klasse wurden in der Musterlösung auch nicht angelegt.

```

ALTER TABLE Person
ADD CONSTRAINT TypLS
CHECK(Typ IN ('S', 'L'));

```

- (c) Schreiben Sie eine SQL-Anweisung, die die Bezeichnung der Klassen bestimmt, die im Schuljahr 2015 die meisten Schüler haben.

Lösungsvorschlag

Falsch: ORDER BY Anzahl;. DESC vergessen.

```

SELECT k.Klassenbezeichnung, COUNT(*) AS Anzahl
FROM Klasse k, Klassenverband v
WHERE
 k.Schuljahr = 2015 AND
 k.Klassenbezeichnung = v.Klassenbezeichnung
GROUP BY k.Klassenbezeichnung
ORDER BY COUNT(*) DESC;

```

- (d) Schreiben Sie eine SQL-Anweisung, die die Namen aller Lehrer bestimmt, die nur Schüler aus ihrem Wohnort unterrichtet haben.

Lösungsvorschlag

```

SELECT DISTINCT l.Name
FROM Person l
WHERE NOT EXISTS(
 SELECT DISTINCT *
 FROM Unterricht u, Klassenverband v, Person s
 WHERE
 u.Lehrer = l.ID AND
 u.Klassenbezeichnung = v.Klassenbezeichnung AND
 v.Schüler = s.ID AND
 l.Wohnort != s.Wohnort
);

```

);

HAVING

- (e) Schreiben Sie eine SQL-Anweisung, die die Namen aller Schüler bestimmt, die immer den gleichen Klassenlehrer hatten.

Lösungsvorschlag

```
SELECT s.Name
FROM Person s, Klasse k, Klassenverband v
WHERE
 v.Klassenbezeichnung = k.Klassenbezeichnung AND
 k.Schuljahr = v.Schuljahr AND
 v.Schüler = s.ID
GROUP BY s.ID, s.Name
HAVING COUNT(*) = 1
```

- (f) Schreiben Sie eine SQL-Anweisung, die alle Paare von Schülern bestimmt, die mindestens einmal in der gleichen Klasse waren. Es genügt dabei, wenn Sie die ID der Schüler bestimmen.

Lösungsvorschlag

```
SELECT DISTINCT s1.ID, s2.ID
FROM Klassenverband s1, Klassenverband s2
WHERE
 s1.Schuljahr = s2.Schuljahr AND
 s1.Schueler <> s2.Schueler AND
 s1.Klassenbezeichnung = s2.Klassenbezeichnung;
```

- (g) Formulieren Sie eine Anfrage in der relationalen Algebra, die die ID aller Schüler bestimmt, die mindestens einmal von „Ludwig Lehrer“ unterrichtet wurden.

Lösungsvorschlag

$$\pi_{\text{Schüler}}(\sigma_{\text{Name}='Ludwig Lehrer'}(\text{Person}) \bowtie_{\text{Person.ID}=\text{Unterricht.Lehrer}} \text{Unterricht} \bowtie_{\text{Unterricht.Klassenbezeichnung}=\text{Klassenverband.Klassenbezeichnung}} \text{Klassenverband})$$

- (h) Formulieren Sie eine Anfrage in der relationalen Algebra, die die Namen und ID der Schüler bestimmt, die von allen Lehrern unterrichtet wurden.

Lösungsvorschlag

$$\pi_{\text{Name,ID}} \left( \left( \pi_{\text{Lehrer,Schueler}}(\text{Unterricht} \bowtie \text{Klassenverband}) \div \pi_{\text{ID}}(\sigma_{\text{Typ}=\text{'L'}}(\text{Person})) \right) \bowtie \text{Person} \right)$$

Beachten Sie bei der Formulierung der SQL-Anfragen, dass die Ergebnisrelationen keine Duplikate enthalten dürfen. Sie dürfen geeignete Views definieren.

## 66116 / 2016 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Ordnen Sie die folgenden Aussagen entsprechend ihres Wahrheitsgehaltes in einer Tabelle der folgenden Form an:

| Kategorie | WAHR   | FALSCH |
|-----------|--------|--------|
| X         | X1, X3 | X2     |
| Y         | Y2     | Y1     |
| ...       | ...    | ...    |

### A Allgemein

- A1** Im Software Engineering geht es vor allem darum qualitativ hochwertige Software zu entwickeln.
- A2** Software Engineering ist gleichbedeutend mit Programmieren.

### B Vorgehensmodelle

- B1** Die Erhebung und Analyse von Anforderungen sind nicht Teil des Software Engineerings.
- B2** Agile Methoden eignen sich besonders gut für die Entwicklung komplexer und sicherer Systeme in verteilten Entwicklerteams.
- B3** Das Spiralmodell ist ein Vorläufer sogenannter Agiler Methoden.

### C Anforderungserhebung

- C1** Bei der Anforderungserhebung dürfen in keinem Fall mehrere Erhebungstechniken (z. B. Workshops, Modellierung) angewendet werden, weil sonst Widersprüche in Anforderungen zu Vorschein kommen könnten.

- C2** Ein Szenario beinhaltet eine Menge von Anwendungsfällen.
- C3** Nicht-funktionale Anforderungen sollten, wenn möglich, immer quantitativ spezifiziert werden.

Nicht-funktionale  
Anforderungen  
Entwurfsmuster  
Schichtenarchitektur  
Blackboard-Muster  
Einbringen von  
Abhängigkeiten  
(Dependency Injection)  
Sequenzdiagramm  
Zustandsdiagramm Wissen  
Komponentendiagramm  
Modell-Präsentation-  
Steuerung  
(Model-View-Controller)  
Einzelstück (Singleton)  
Kommando (Command)  
Validation  
Verifikation

## D Architekturmuster

- D1** Schichtenarchitekturen sind besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.
- D2** Das Black Board Muster ist besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.
- D3** „Dependency Injection“ bezeichnet das Konzept, welches Abhängigkeiten zur Laufzeit reglementiert.

## E UML

- E1** Sequenzdiagramme beschreiben Teile des Verhaltens eines Systems.
- E2** Zustandsübergangsdiagramme beschreiben das Verhalten eines Systems.
- E3** Komponentendiagramme beschreiben die Struktur eines Systems.

## F Entwurfsmuster

- F1** Das MVC Pattern verursacht eine starke Abhängigkeit zwischen Datenmodell und Benutzeroberfläche.
- F2** Das Singleton Pattern stellt sicher, dass es zur Laufzeit von einer bestimmten Klasse höchstens ein Objekt gibt.
- F3** Im Kommando Entwurfsmuster (engl. „Command Pattern“) werden Befehle in einem sog. Kommando-Objekt gekapselt, um sie bei Bedarf rückgängig zu machen.

## G Testen

- G1** Validation dient der Überprüfung von Laufzeitfehlern.
- G2** Testen ermöglicht sicherzustellen, dass ein Programm absolut fehlerfrei ist.
- G3** Verifikation dient der Überprüfung, ob ein System einer Spezifikation entspricht.

Lösungsvorschlag

| Kategorie | WAHR       | FALSCH                            |
|-----------|------------|-----------------------------------|
| A         | A1         | A2                                |
| B         | B3         | B1, B2                            |
| C         | C3         | C1, C2                            |
| D         | D3         | D1, D2                            |
| E         | E1, E2, E3 |                                   |
| F         | F2, F3     | F1                                |
| G         | G3         | G1 <sup>a</sup> , G2 <sup>b</sup> |

<sup>a</sup>Validierung: Prüfung der Eignung beziehungsweise der Wert einer Software bezogen auf ihren Einsatzzweck: „Wird das richtige Produkt entwickelt?“

<sup>b</sup>Ein Softwaretest prüft und bewertet Software auf Erfüllung der für ihren Einsatz definierten Anforderungen und misst ihre Qualität.

## 66116 / 2017 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Konstruieren Sie einen B-Baum, dessen Knoten maximal 4 Einträge enthalten können, indem Sie der Reihe nach diese Suchschlüssel einfügen:

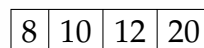
8, 10, 12, 20, 5, 30, 25, 11

Anschließend löschen Sie den Eintrag mit dem Suchschlüssel 8.

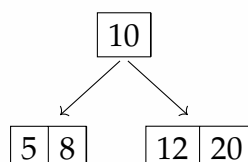
Zeigen Sie jeweils graphisch den entstehenden Baum nach relevanten Zwischenschritten; insbesondere nach Einfügen der 5 sowie nach dem Einfügen der 11 und nach dem Löschen der 8.

Lösungsvorschlag

- Schlüsselwert 8 (einfaches Einfügen)
- Schlüsselwert 10 (einfaches Einfügen)
- Schlüsselwert 12 (einfaches Einfügen)
- Schlüsselwert 20 (einfaches Einfügen)

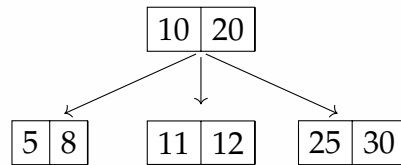


- Schlüsselwert 5 (Split)

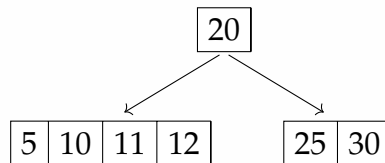


- Schlüsselwert 30 (einfaches Einfügen)
- Schlüsselwert 25 (einfaches Einfügen)

- Schlüsselwert 11 (Split)



- Löschen des Schlüsselwerts 8 (Mischen/Verschmelzen)



## 66116 / 2017 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 5

In der folgenden Datenbank sind die Ausleihvorgänge einer Bibliothek gespeichert:

| Ausleihe | LNr | Name | Adresse | BNr | Titel | Kategorie | ExemplarNr | 1 | Müller | Winklerstr. 1 | Datenbanksysteme | Informatik 1 | 1 | Miller | Winklerstr. 1 | Datenbanksysteme | Informatik 2 | 2 | Huber | Friedrichstr. | 2 | Anatomie I | Medizin 5 | 2 | Huber | Friedrichstr. 3 | Harry Potter | Literatur 20 | 3 | Meier | Bismarkstr. 4 | OODBS | Informatik 1 | 4 | Meier | Marktpl. 5 | Pippi Langstrumpf | Literatur 1

Für die Datenbank gilt:

Jeder Leser hat eine eindeutige Lesernummer (LNr), einen Namen und eine Adresse. Ein Buch hat eine Buchnummer (BNr), einen Titel und eine Kategorie. Es kann mehrere Exemplare eines Buches geben, welche durch eine, innerhalb einer Buchnummer eindeutigen, Exemplarnummer unterschieden werden.

- Beschreiben Sie kurz, welche Redundanzen in der Datenbank vorhanden sind und welche Anomalien auftreten können.
- Nachfolgend sind alle nicht-trivialen funktionalen Abhängigkeiten, welche in der obigen Datenbank gelten, angegeben:

$$\text{FA} = \left\{ \begin{array}{l} \{ \text{LNr} \} \rightarrow \{ \text{Name} \}, \\ \{ \text{LNr} \} \rightarrow \{ \text{Adresse} \}, \\ \{ \text{BNr} \} \rightarrow \{ \text{Titel} \}, \\ \{ \text{BNr} \} \rightarrow \{ \text{Kategorie} \}, \\ \{ \text{LNr}, \text{BNr}, \text{ExemplarNr} \} \rightarrow \{ \text{Name}, \text{Adresse}, \text{Titel}, \text{Kategorie} \}, \end{array} \right\}$$

Einziger Schlüsselkandidat ist  $\{ \text{LNr}, \text{BNr}, \text{ExemplarNr} \}$ . Überführen Sie das Schema mit Hilfe des Synthesealgorithmus für 3NF in die dritte Normalform.



**(i) Kanonische Überdeckung**

— Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden. —

**i. Linksreduktion**

— Führe für jede funktionale Anhängigkeit  $\alpha \rightarrow \beta \in F$  die Linksreduktion durch, überprüfe also für alle  $A \in \alpha$ , ob  $A$  überflüssig ist, d. h. ob  $\beta \subseteq \text{AttrHülle}(F, \alpha - A)$ . —

$$\begin{aligned} \text{AttrHülle}(FA, \{L Nr, B Nr, ExemplarNr \setminus L Nr\}) &= \{ \text{Titel}, \text{Kategorie} \} \\ \text{AttrHülle}(FA, \{L Nr, B Nr, ExemplarNr \setminus B Nr\}) &= \{ \text{Name}, \text{Adresse} \} \\ \text{AttrHülle}(FA, \{L Nr, B Nr, ExemplarNr \setminus \text{ExemplarNr}\}) &= \{ \text{Name}, \text{Adresse}, \text{Titel}, \text{Kategorie} \} \end{aligned}$$

$$FA = \left\{ \begin{array}{l} \{ L Nr \} \rightarrow \{ \text{Name} \}, \\ \{ L Nr \} \rightarrow \{ \text{Adresse} \}, \\ \{ B Nr \} \rightarrow \{ \text{Titel} \}, \\ \{ B Nr \} \rightarrow \{ \text{Kategorie} \}, \\ \{ L Nr, B Nr \} \rightarrow \{ \text{Name}, \text{Adresse}, \text{Titel}, \text{Kategorie} \}, \end{array} \right\}$$

**ii. Rechtsreduktion**

— Führe für jede (verbliebene) funktionale Abhängigkeit  $\alpha \rightarrow \beta$  die Rechtsreduktion durch, überprüfe also für alle  $B \in \beta$ , ob  $B \in \text{AttrHülle}(F - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B)), \alpha)$  gilt. In diesem Fall ist  $B$  auf der rechten Seite überflüssig und kann eliminiert werden,  $\alpha \rightarrow \beta$  wird durch  $\alpha \rightarrow (\beta - B)$  ersetzt. —

$$\begin{aligned}
& \text{AttrHülle}(FA - (\{LNr\} \rightarrow \{Name\}) \cup (\{LNr\} \rightarrow \{\emptyset\}), \{LNr\}) = \\
& \qquad \qquad \qquad \{Adresse\} \\
& \text{AttrHülle}(FA - (\{LNr\} \rightarrow \{Adresse\}) \cup (\{LNr\} \rightarrow \{\emptyset\}), \{LNr\}) = \\
& \qquad \qquad \qquad \{Name\} \\
& \text{AttrHülle}(FA - (\{BNr\} \rightarrow \{Titel\}) \cup (\{BNr\} \rightarrow \{\emptyset\}), \{BNr\}) = \\
& \qquad \qquad \qquad \{Kategorie\} \\
& \text{AttrHülle}(FA - (\{BNr\} \rightarrow \{Kategorie\}) \cup (\{BNr\} \rightarrow \{\emptyset\}), \{BNr\}) = \\
& \qquad \qquad \qquad \{Titel\} \\
& \text{AttrHülle}(FA - (\{LNr, BNr\} \rightarrow \{Name, Adresse, Titel, Kategorie\}) \cup (\{LNr, BNr\} \rightarrow \{Adresse, Titel, Kategorie\}), \{LNr, BNr\}) = \\
& \qquad \qquad \qquad \{Name, Adresse, Titel, Kategorie\} \\
& \text{AttrHülle}(FA - (\{LNr, BNr\} \rightarrow \{Name, Adresse, Titel, Kategorie\}) \cup (\{LNr, BNr\} \rightarrow \{Name, Titel, Kategorie\}), \{LNr, BNr\}) = \\
& \qquad \qquad \qquad \{Name, Adresse, Titel, Kategorie\} \\
& \text{AttrHülle}(FA - (\{LNr, BNr\} \rightarrow \{Name, Adresse, Titel, Kategorie\}) \cup (\{LNr, BNr\} \rightarrow \{Name, Adresse, Titel\}), \{LNr, BNr\}) = \\
& \qquad \qquad \qquad \{Name, Adresse, Titel, Kategorie\} \\
& \text{AttrHülle}(FA - (\{LNr, BNr\} \rightarrow \{Name, Adresse, Titel, Kategorie\}) \cup (\{LNr, BNr\} \rightarrow \{Name, Adresse\}), \{LNr, BNr\}) = \\
& \qquad \qquad \qquad \{Name, Adresse, Titel, Kategorie\}
\end{aligned}$$

$$FA = \left\{ \begin{array}{l} \{LNr\} \rightarrow \{Name\}, \\ \{LNr\} \rightarrow \{Adresse\}, \\ \{BNr\} \rightarrow \{Titel\}, \\ \{BNr\} \rightarrow \{Kategorie\}, \\ \{LNr, BNr\} \rightarrow \{\emptyset\}, \end{array} \right\}$$

### iii. Löschen leerer Klauseln

— Entferne die funktionalen Abhängigkeiten der Form  $\alpha \rightarrow \emptyset$ , die im 2. Schritt möglicherweise entstanden sind.

$$FA = \left\{ \begin{array}{l} \{LNr\} \rightarrow \{Name\}, \\ \{LNr\} \rightarrow \{Adresse\}, \\ \{BNr\} \rightarrow \{Titel\}, \\ \{BNr\} \rightarrow \{Kategorie\}, \end{array} \right\}$$

### iv. Vereinigung

— Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form  $\alpha \rightarrow \beta_1, \dots, \alpha \rightarrow \beta_n$ , so dass  $\alpha \rightarrow \beta_1 \cup \dots \cup \beta_n$  verbleibt.

$$FA = \left\{ \begin{array}{l} \{ LNr \} \rightarrow \{ Name, Adresse \}, \\ \{ BNr \} \rightarrow \{ Titel, Kategorie \}, \end{array} \right\}$$

(ii) **Relationsschemata formen**

— Erzeuge für jede funktionale Abhängigkeit  $\alpha \rightarrow \beta \in F_c$  ein Relationenschema  $\mathcal{R}_\alpha := \alpha \cup \beta$ .

(iii) **Schlüssel hinzufügen**

— Falls eines der in Schritt 2. erzeugten Schemata  $\mathcal{R}_\alpha$  einen Schlüsselkandidaten von  $\mathcal{R}$  bezüglich  $F_c$  enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten  $\mathcal{K} \subseteq \mathcal{R}$  aus und definiere folgendes zusätzliche Schema:  $\mathcal{R}_\mathcal{K} := \mathcal{K}$  und  $\mathcal{F}_\mathcal{K} := \emptyset$

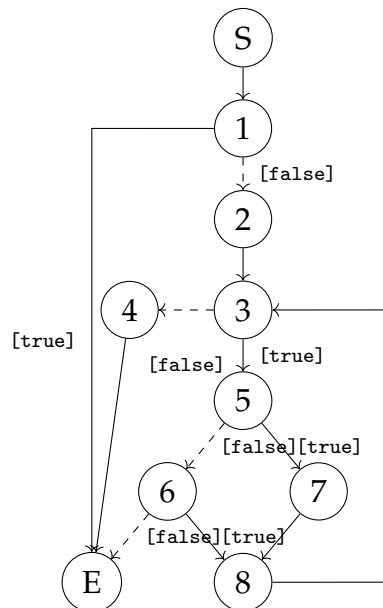
(iv) **Entfernung überflüssiger Teilschemata**

— Eliminiere diejenigen Schemata  $\mathcal{R}_\alpha$ , die in einem anderen Relationenschema  $\mathcal{R}_{\alpha'}$  enthalten sind, d. h.  $\mathcal{R}_\alpha \subseteq \mathcal{R}_{\alpha'}$ .

**66116 / 2017 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1**

Gegeben Sei folgende Methode und ihr Kontrollflussgraph:

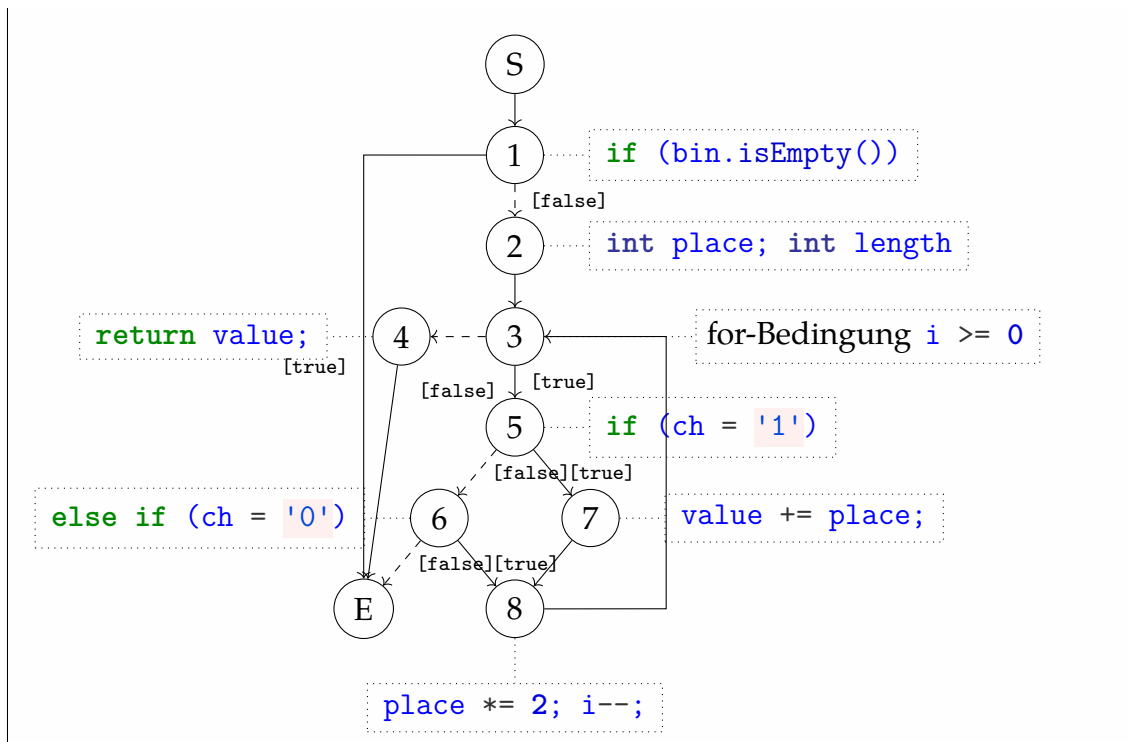
```
int binToInt(String bin) {
 if (bin.isEmpty())
 return -1;
 int place = 1, value = 0;
 int length = bin.length() -
 ↪ 1;
 for (int i = length; i >= 0;
 ↪ --i) {
 char ch = bin.charAt(i);
 if (ch == '1') {
 value += place;
 } else if (ch == '0') {
 // do nothing
 } else {
 return -1;
 }
 place *= 2;
 }
 return value;
}
```



Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/sosy/ab\\_7/Aufgabe5.java](https://github.com/bschlangaul/aufgaben/sosy/ab_7/Aufgabe5.java)

- (a) Geben Sie je einen Repräsentanten aller Pfadklassen im Kontrollflussgraphen an, die zum Erzielen einer vollständigen

Lösungsvorschlag



(i) Verzweigungsüberdeckung

Lösungsvorschlag

**p1 (Pfad 1)** S 1 E  
**p2** S 1 2 3 4 E  
**p3** S 1 2 3 5 7 8 3 5 6 8 3 5 6 E

(ii) Schleife-Inneres-Überdeckung

Lösungsvorschlag

**Äußere Pfade (äußere Pfade):** (ohne Ausführung der Wiederholung)

**p1** S 1 E  
**p2** S 1 2 3 4 E

**Grenzpfade (boundary test)** (alle Pfade, die die Wiederholung betreten, aber nicht wiederholen; innerhalb des Schleifenrumpfes alle Pfade!)**p4** S 1 2 3 5 6 E**Innere Pfade (interior test)** (alle Pfade mit einer Wiederholung des Schleifenrumpfes; innerhalb des Schleifenrumpfes wieder alle Pfade!)

**p5** S 1 2 3 5 7 8 3 5 7 8 3 4 E  
**p6** S 1 2 3 5 7 8 3 5 6 8 3 4 E  
**p7** S 1 2 3 5 6 8 3 5 6 8 3 4 E  
**p8** S 1 2 3 5 6 8 3 5 7 8 3 4 E  
**p9** S 1 2 3 5 7 8 3 5 7 8 3 5 6 E

**p10 = p3** S 1 2 3 5 **7** 8 3 5 **6** 8 3 5 **6** E  
**p11** S 1 2 3 5 **6** 8 3 5 **6** 8 3 5 **6** E  
**p12** S 1 2 3 5 **6** 8 3 5 **7** 8 3 5 **6** E

mit minimaler Testfallanzahl genügen würden.

- (b) Welche der vorangehend ermittelten Pfade sind mittels Testfälle tatsächlich überdeckbar („feasible“)? Falls der Pfad ausführbar ist, geben Sie bitte den Testfall an, andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

Erweitere Methode, die die Knotennamen ausgibt:

```
public static final String RESET = "\u001B[0m";
public static final String ROT = "\u001B[31m";
public static final String GRÜN = "\u001B[32m";

static int binToIntLog(String bin) {
 System.out.println("\nInput: " + bin);
 System.out.print("S");
 System.out.print(1);
 if (bin.isEmpty()) {
 System.out.print("E");
 System.out.println("\nOutput: " + -1);
 return -1;
 }
 System.out.print(2);
 int place = 1, value = 0;
 int length = bin.length() - 1;
 System.out.print(3);
 for (int i = length; i >= 0; --i) {
 char ch = bin.charAt(i);
 System.out.print(5);
 if (ch == '1') {
 System.out.print(ROT + 7 + RESET);
 value += place;
 } else {
 System.out.print(GRÜN + 6 + RESET);
 if (ch == '0') {
 // do nothing
 } else {
 System.out.print("E");
 System.out.println("\nOutput: " + -1);
 return -1;
 }
 }
 }
 System.out.print(8);
 place *= 2;
 System.out.print(3);
}
System.out.print(4);
System.out.print("E");
System.out.println("\nOutput: " + value);
return value;
```

```

}

public static void main(String[] args) {
 binToIntLog(""); // p1
 binToIntLog("??"); // p2 not feasible
 binToIntLog("x01"); // p3
 binToIntLog("x"); // p4

 binToIntLog("11"); // p5
 binToIntLog("01"); // p6
 binToIntLog("00"); // p7
 binToIntLog("10"); // p8

 binToIntLog("x11"); // p9
 binToIntLog("x01"); // p10
 binToIntLog("x00"); // p11
 binToIntLog("x10"); // p12
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/soey/ab\\_7/Aufgabe5.java](https://github.com/bschlangaul/aufgaben/soey/ab_7/Aufgabe5.java)

Alle mit Ausnahme von p2.

**p2** ist nicht überdeckbar. Passiert ein Wert der Variable `bin` die erste if-Verzweigung, dann hat der Wert eine Länge größer 0 und betritt deshalb die Wiederholung mit fester Anzahl.

|          |                               |                               |
|----------|-------------------------------|-------------------------------|
| p1       | S 1 E                         | <code>binToInt("");</code>    |
| p2       | S 1 2 3 4 E                   | not feasible                  |
| p3       | S 1 2 3 5 7 8 3 5 6 8 3 5 6 E | <code>binToInt("x01");</code> |
| p4       | S 1 2 3 5 6 E                 | <code>binToInt("x");</code>   |
| p5       | S 1 2 3 5 7 8 3 5 7 8 3 4 E   | <code>binToInt("11");</code>  |
| p6       | S 1 2 3 5 7 8 3 5 6 8 3 4 E   | <code>binToInt("01");</code>  |
| p7       | S 1 2 3 5 6 8 3 5 6 8 3 4 E   | <code>binToInt("00");</code>  |
| p8       | S 1 2 3 5 6 8 3 5 7 8 3 4 E   | <code>binToInt("10");</code>  |
| p9       | S 1 2 3 5 7 8 3 5 7 8 3 5 6 E | <code>binToInt("x11");</code> |
| p10 = p3 | S 1 2 3 5 7 8 3 5 6 8 3 5 6 E | <code>binToInt("x01");</code> |
| p11      | S 1 2 3 5 6 8 3 5 6 8 3 5 6 E | <code>binToInt("x00");</code> |
| p12      | S 1 2 3 5 6 8 3 5 7 8 3 5 6 E | <code>binToInt("x10");</code> |

- (c) Bestimmen Sie anhand des Kontrollflussgraphen die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach Mc-Cabe.

**Binärverzweigungen** 4**Knoten** 10**Kanten** 13

Anhand der Binärverzweigungen:

$$\begin{aligned}M &= b + p \\&= 4 + 1 \\&= 5\end{aligned}$$

oder durch Anzahl Kanten  $e$  und Knoten  $n$ 

$$\begin{aligned}M &= e - n + 2p \\&= 13 - 10 + 2 \cdot 1 \\&= 5\end{aligned}$$

- (d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

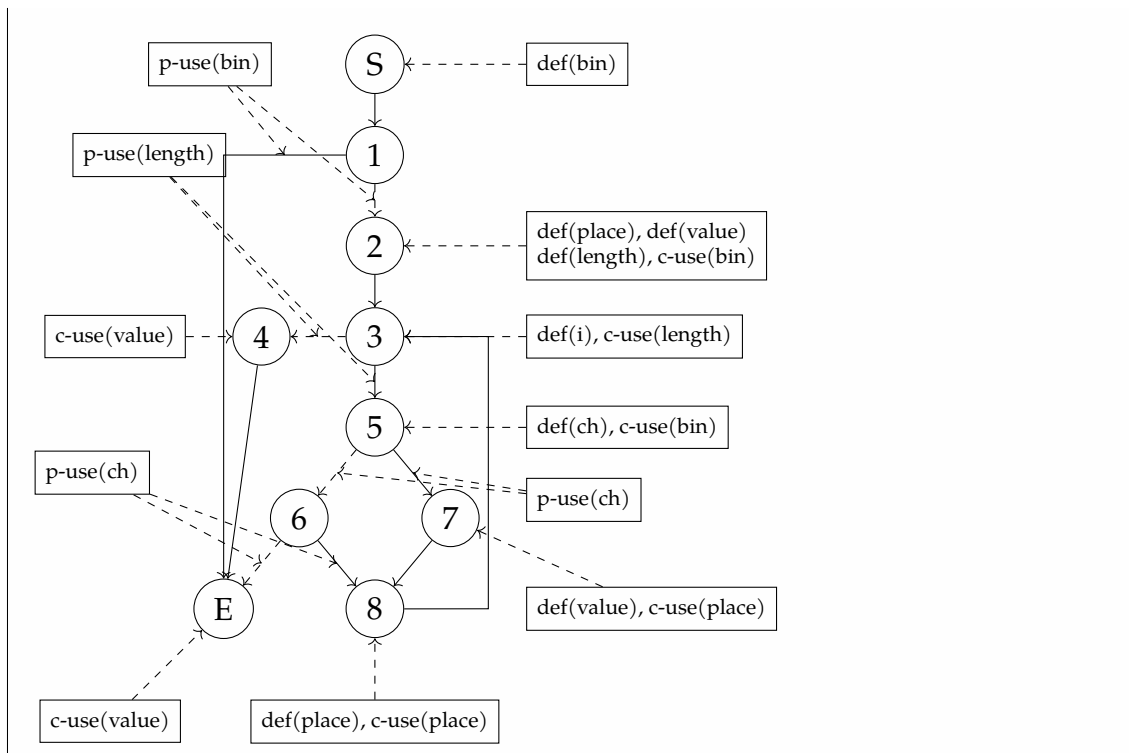
Lösungsvorschlag

Nein, da **p2** nicht überdeckbar ist.

- (e) Geben Sie zu jedem Knoten die jeweilige Datenfluss-Annotation (def s bzw. uses) für jede betroffene Variable in der zeitlichen Reihenfolge ihres Auftretens zur Laufzeit an.

Lösungsvorschlag





## 66116 / 2017 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 4

Sie dürfen im Folgenden davon ausgehen, dass keinerlei Under- oder Overflows auftreten.

Gegeben sei die folgende Methode mit Vorbedingung  $P := x \geq 0 \wedge y \geq 0$  und Nachbedingung  $Q := x \cdot y = z$ .

```
int mul (int x , int y) {
 /* P */
 int z = 0, i = 0;
 while (i++ != x)
 z += y;
 /* Q */
 return z;
}
```

Betrachten Sie dazu die folgenden drei Prädikate:

- $I_1 := z + i \cdot y = x \cdot y$
- $I_2 := \text{false}$
- $I_3 := z + (x - i) \cdot y = x \cdot y$

(a) Beweisen Sie formal für jedes der drei Prädikate, ob es unmittelbar vor Betreten der Schleife in `mul` gilt oder nicht.

$$\begin{aligned}
\text{wp}(\text{"Code vor der Schleife"}, I_1) &\equiv \text{wp}(\text{"int } z = 0, i = 0; ", z + i \cdot y = x \cdot y) \\
&\equiv \text{wp}("", 0 + 0 \cdot y = x \cdot y) \\
&\equiv 0 = x \cdot y \\
&\equiv \text{falsch}
\end{aligned}$$

$$\begin{aligned}
\text{wp}(\text{"Code vor der Schleife"}, I_2) &\equiv \text{wp}(\text{"int } z = 0, i = 0; ", \text{false}) \\
&\equiv \text{wp}("", \text{false}) \\
&\equiv \text{false} \\
&\equiv \text{falsch}
\end{aligned}$$

$$\begin{aligned}
\text{wp}(\text{"Code vor der Schleife"}, I_3) &\equiv \text{wp}(\text{"int } z = 0, i = 0; ", z + (x - i) \cdot y = x \cdot y) \\
&\equiv \text{wp}("", 0 + (x - 0) \cdot y = x \cdot y) \\
&\equiv x \cdot y = x \cdot y \\
&\equiv \text{wahr}
\end{aligned}$$

- (b) Weisen Sie formal nach, welche der drei Prädikate Invarianten des Schleifenrumpfs in `mul` sind oder welche nicht.

Für den Nachweis muss der Code etwas umformuliert werden:

```

int mul (int x , int y) {
 /* P */
 int z = 0, i = 0;
 while (i != x) {
 i = i + 1;
 z = z + y;
 }
 /* Q */
 return z;
}

```

$$\begin{aligned}
\text{wp}(\text{"Code Schleife"}, I_1 \wedge i \neq x) &\equiv \text{wp}(\text{"i = i + 1; z = z + y;"}, z + i \cdot y = x \cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\text{"i = i + 1;"}, z + y + i \cdot y = x \cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\text{"", } z + y + (i + 1) \cdot y = x \cdot y \wedge i + 1 \neq x) \\
&\equiv z + y + (i + 1) \cdot y = x \cdot y \wedge i + 1 \neq x \\
&\equiv z + i \cdot y + 2 \cdot y = x \cdot y \wedge i + 1 \neq x \\
&\equiv \text{falsch} \wedge i + 1 \neq x \\
&\equiv \text{falsch}
\end{aligned}$$

$$\begin{aligned}
\text{wp}(\text{"Code Schleife"}, I_2 \wedge i \neq x) &\equiv \text{wp}(\text{"i = i + 1; z = z + y;"}, \text{false} \wedge i \neq x) \\
&\equiv \text{wp}(\text{"", } \text{false} \wedge i \neq x) \\
&\equiv \text{falsch} \wedge i \neq x \\
&\equiv \text{falsch}
\end{aligned}$$

$$\begin{aligned}
\text{wp}(\text{"Code Schleife"}, I_3 \wedge i \neq x) &\equiv \text{wp}(\text{"i = i + 1; z = z + y;"}, z + (x - i) \cdot y = x \cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\text{"i = i + 1;"}, z + y + (x - i) \cdot y = x \cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\text{"", } z + y + (x - i + 1) \cdot y = x \cdot y \wedge i + 1 \neq x) \\
&\equiv z + y + x \cdot y - i \cdot y + y = x \cdot y \wedge i + 1 \neq x \\
&\equiv z + 2 \cdot y + x \cdot y - i \cdot y = x \cdot y \wedge i + 1 \neq x \\
&\equiv \text{wahr}
\end{aligned}$$

- (c) Beweisen Sie formal, aus welchen der drei Prädikate die Nachbedingung gefolgt werden darf bzw. nicht gefolgt werden kann.

Lösungsvorschlag

$$I_1 := z + i \cdot y = x \cdot y \quad I_2 := \text{false} \quad I_3 := z + (x - i) \cdot y = x \cdot y$$

$$\begin{aligned}
\text{wp}(\text{"Code nach Schleife"}, I_1 \wedge i = x) &\equiv \text{wp}(\text{"", } z + i \cdot y = x \cdot y \wedge i = x) \\
&\equiv z + i \cdot y = x \cdot y \wedge i = x \\
&\equiv z + x \cdot y = x \cdot y \\
&\equiv \text{wahr}
\end{aligned}$$

$$\begin{aligned}
\text{wp}(\text{"Code nach Schleife"}, I_2 \wedge i = x) &\equiv \text{wp}(\text{"", false} \wedge i = x) \\
&\equiv \text{false} \wedge i = x \\
&\equiv \text{falsch} \\
&\neq Q
\end{aligned}$$

$$\begin{aligned}
\text{wp}(\text{"Code nach Schleife"}, I_3 \wedge i = x) &\equiv \text{wp}(\text{"", } z + (x - i) \cdot y = x \cdot y \wedge i = x) \\
&\equiv z + (x - i) \cdot y = x \cdot y \wedge i = x \\
&\equiv z + (x - x) \cdot y = x \cdot y \\
&\equiv z + 0 \cdot y = x \cdot y \\
&\equiv z + 0 = x \cdot y \\
&\equiv z = x \cdot y \\
&\equiv Q
\end{aligned}$$

- (d) Skizzieren Sie den Beweis der totalen Korrektheit der Methode `mul`. Zeigen Sie dazu auch die Terminierung der Methode.

Lösungsvorschlag

Aus den Teilaufgaben folgt der Beweis der partiellen Korrektheit mit Hilfe der Invariante  $i_3$ .  $i$  steigt streng monoton von 0 an so lange gilt  $i \neq x$ .  $i = x$  ist die Abbruchbedingung für die bedingte Wiederholung. Dann terminiert die Methode. Die Methode `mul` ist also total korrekt.

## 66116 / 2017 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 5

Gegeben ist die folgende Definition zweier Tabellen:

```

CREATE TABLE R2 (
 b integer not null,
 c integer unique,
 primary key (b)
);

CREATE TABLE R1 (
 a integer not null,
 b integer references R2,
 primary key (a)
);

```

Geben Sie jeweils an, ob das Statement syntaktisch korrekt ist und ob es von der gegebenen Datenbank ausgeführt werden kann.

Beantworten Sie jede der folgenden Fragen unabhängig von allen anderen, des liegt immer das hier gezeigte Schema vor und alle Relationen sind leer.

(a) `DELETE FROM R1;`

DELETE  
INSERT  
DROP TABLE

Lösungsvorschlag

korrekt

(b) `INSERT INTO R2 VALUES (1,1);`  
`INSERT INTO R1 VALUES (1,1);`  
`INSERT INTO R1 VALUES (2,1);`  
`INSERT INTO R1 VALUES (3,1);`

Lösungsvorschlag

korrekt

(c)

```
INSERT INTO R2 VALUES (1,1);
INSERT INTO R2 VALUES (2,2);
INSERT INTO R1 VALUES (1,1);
DELETE FROM R2 WHERE b=a;
```

Lösungsvorschlag

falsch: Fehlermeldung column a "does not exist"

```
INSERT INTO R2 VALUES (1,1);
INSERT INTO R2 VALUES (2,2);
INSERT INTO R1 VALUES (1,1);
-- Wir löschen von R1 weil R2 auf R1 referenziert
-- b kann nur mit Integer verglichen werden.
DELETE FROM R1 WHERE b=1;
```

(d)

```
INSERT INTO R1 SELECT * FROM R1;
```

Lösungsvorschlag

korrekt

(e)

```
DROP TABLE R2 FROM DATABASE;
```

Lösungsvorschlag

falsch: Fehlermeldung ERROR: syntax error at or near "FROM"

Müsste so lauten:

```
-- Zuerst R1 löschen, wegen der Referenz
DROP TABLE R1;
DROP TABLE R2;
```

**66116 / 2017 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 6**

Folgende Tabellen veranschaulichen eine Ausprägung eines Fluginformationssystems:

**Flughäfen**

| Code | Stadt         | Transferzeit (min) |
|------|---------------|--------------------|
| LHR  | London        | 30                 |
| LGW  | London        | 20                 |
| JFK  | New York City | 60                 |
| EWB  | New York City | 35                 |
| MUC  | München       | 30                 |
| FRA  | Frankfurt     | 45                 |

**Verbindungen**

| ID  | Von | Nach | Linie | Abflug (MEZ)        | Ankunft (MEZ)       |
|-----|-----|------|-------|---------------------|---------------------|
| 410 | MUC | FRA  | LH    | 2016-02-24 07:00:00 | 2016-02-24 08:10:00 |
| 411 | MUC | FRA  | LH    | 2016-02-24 08:00:00 | 2016-02-24 09:10:00 |
| 412 | FRA | JFK  | LH    | 2016-02-24 10:50:00 | 2016-02-24 19:50:00 |

**Hinweise**

- Formulieren Sie alle Abfragen in SQL-92 (insbesondere sind LIMIT, TOP, FETCH FIRST, ROWNUM und dergleichen nicht erlaubt).
- Alle Datum/Zeit-Angaben erlauben arithmetische Operationen, beispielsweise wird bei der Operation `ankunft + transferzeit` die transferzeit auf den Zeitstempel `ankunft` addiert.
- Es müssen keine Zeitverschiebungen berücksichtigt werden. Alle Zeitstempel sind in MEZ.

```
CREATE TABLE Flughäfen (
 Code VARCHAR(3) PRIMARY KEY,
 Stadt VARCHAR(20),
 Transferzeit integer
);
```

```
CREATE TABLE Verbindungen (
 ID integer PRIMARY KEY,
 Von VARCHAR(3) REFERENCES Flughäfen(Code),
 Nach VARCHAR(3) REFERENCES Flughäfen(Code),
 Linie VARCHAR(20),
 Abflug timestamp,
```

```
Ankunft timestamp
);
```

```
INSERT INTO Flughäfen VALUES
('LHR', 'London', 30),
('LGW', 'London', 20),
('JFK', 'New York City', 60),
('EWR', 'New York City', 35),
('MUC', 'München', 30),
('FRA', 'Frankfurt', 45);
```

```
INSERT INTO Verbindungen VALUES
(410, 'MUC', 'FRA', 'LH', '2016-02-24 07:00:00', '2016-02-24 08:10:00'),
(411, 'MUC', 'FRA', 'LH', '2016-02-24 08:00:00', '2016-02-24 09:10:00'),
(412, 'FRA', 'JFK', 'LH', '2016-02-24 10:50:00', '2016-02-24 19:50:00'),
(413, 'MUC', 'LHR', 'LH', '2016-02-24 10:00:00', '2016-02-24 12:10:00'),
(414, 'MUC', 'LGW', 'LH', '2016-02-24 11:00:00', '2016-02-24 13:20:00'),
(415, 'MUC', 'LHR', 'LH', '2016-02-24 12:00:00', '2016-02-24 14:00:00');
```

- (a) Ermitteln Sie die Städte, in denen es mehr als einen Flughafen gibt.

Lösungsvorschlag

```
SELECT Stadt FROM Flughäfen
GROUP BY Stadt
HAVING count(Stadt) > 1;
```

- (b) Ermitteln Sie die Städte, in denen man mit der Linie „LH“ an mindestens zwei verschiedenen Flughäfen landen kann.

Lösungsvorschlag

```
SELECT Stadt FROM Flughäfen
WHERE Stadt IN (
 SELECT Stadt FROM Flughäfen, Verbindungen
 WHERE
 Code = Nach AND
 Linie = 'LH'
 GROUP BY Stadt
)
GROUP BY Stadt
HAVING COUNT(Stadt) > 1;
```

- (c) Ermitteln Sie die Flugzeit des kürzesten Direktflugs von München nach London.

Lösungsvorschlag

```
CREATE VIEW Flugdauer AS
 SELECT ID, Ankunft - Abflug AS Dauer FROM Flughäfen v, Flughäfen n,
 ↳ Verbindungen
 WHERE
 n.Code = Nach AND
 v.Code = Von AND
 v.Stadt = 'München' AND
 n.Stadt = 'London';

SELECT a.Dauer FROM Flugdauer a, Flugdauer b
```

```

WHERE a.Dauer >= b.Dauer
GROUP BY a.Dauer
HAVING COUNT(*) <= 1;

```

- (d) Ermitteln Sie die kürzeste Roundtrip-Zeit (nur Direktflüge) zwischen den Flughäfen FRA und JFK (Transferzeit am Flughafen JFK beachten).

## 66116 / 2017 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 4

Die folgende Seite enthält Software-Quellcode, der einen Algorithmus zur binären Suche implementiert. Dieser ist durch Inspektion zu überprüfen. Im Folgenden sind die Regeln der Inspektion angegeben.

|     |                  |                                                                                                                                                |
|-----|------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| RM1 | (Dokumentation)  | Jede Quellcode-Datei beginnt mit einem Kommentar, der den Klassennamen, Versionsinformationen, Datum und Urheberrechtsangaben enthält.         |
| RM2 | (Dokumentation)  | Jede Methode wird kommentiert. Der Kommentar enthält eine vollständige Beschreibung der Signatur so wie eine Design-by-Contract-Spezifikation. |
| RM3 | (Dokumentation)  | Deklarationen von Variablen werden kommentiert.                                                                                                |
| RM4 | (Dokumentation)  | Jede Kontrollstruktur wird kommentiert.                                                                                                        |
| RM5 | (Formatierung)   | Zwischen einem Schlüsselwort und einer Klammer steht ein Leerzeichen.                                                                          |
| RM6 | (Formatierung)   | Zwischen binären Operatoren und den Operanden stehen Leerzeichen.                                                                              |
| RM7 | (Programmierung) | Variablen werden in der Anweisung initialisiert, in der sie auch deklariert werden.                                                            |
| RM8 | (Bezeichner)     | Klassennamen werden groß geschrieben, Variablennamen klein.                                                                                    |

```

/**
 * BinarySearch.java
 *
 * Eine Implementierung der "Binaere Suche"
 * mit einem iterativen Algorithmus
 */
class BinarySearch {

 /**
 * BinaereSuche
 * a: Eingabefeld
 * item: zuzuchendesElement

```



```
* returnValue: der Index des zu suchenden Elements oder -1
*
* Vorbedingung:
* a.length > 0
* a ist ein linear geordnetes Feld:
* For all k: (1 <= k < a.length) ==> (a[k-1] <=a [k])
*
* Nachbedingung:
* Wenn item in a, dann gibt es ein k mit a[k] == item und returnValue == k
* Genau dann wenn returnValue == -1 gibt es kein k mit 0 <= k < a.length
* und a[k]==item.
*/
public static int binarySearch(float a[], float item) {

 int End; // exklusiver Index fuer das Ende des
 // zudurchsuchenden Teils des Arrays
 int start = 1; // inklusiver Index fuer den Anfang der Suche
 End = a.length;

 // Die Schleife wird verlassen, wenn keine der beiden Haelften das
 // Element enthaelt.
 while(start < End) {

 // Teilung des Arrays in zwei Haelften
 // untere Haelfte: [0,mid[
 // obere Haelfte:]mid,End[
 int mid = (start + End) / 2;

 if (item > a[mid]) {
 // Ausschluss der oberen Haelfte
 start = mid + 1;
 } else if(item < a[mid]) {
 // Ausschluss der unteren Haelfte
 End = mid-1;
 } else {
 // Das gesuchte Element wird zurueckgegeben
 return (mid);
 }
 } // end of while

 // Bei Misserfolg der Suche wird -1 zurueckgegeben
 return (-1);
}
```

- (a) Überprüfen Sie durch Inspektion, ob die obigen Regeln für den Quellcode eingehalten wurden. Erstellen Sie eine Liste mit allen Verletzungen der Regeln. Geben Sie für jede Verletzung einer Regel die Zeilennummer, Regelnummer und Kommentar an, z. B. (07, RM4, while nicht kommentiert). Schreiben Sie nicht in den Quellcode.

Lösungsvorschlag

| Zeile | Regel | Kommentar                                                                                                                                                            |
|-------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3-8   | RM1   | Fehlen von Versionsinformationen, Datum und Urheberrechtsangaben                                                                                                     |
| 11-26 | RM2   | Fehlen der Invariante in der Design-by-Contract-Spezifikation                                                                                                        |
| 36,46 | RM5   | Fehlen des Leerzeichens vor der Klammer                                                                                                                              |
| 48    | RM6   | Um einen binären (zweistellige) Operator handelt es sich im Code-Beispiel um den Subtraktionsoperator: <code>mid-1</code> . Hier fehlen die geforderten Leerzeichen. |
| 32    | RM7   | Die Variable <code>End</code> wird in Zeile 32 deklariert, aber erst in Zeile initialisiert <code>End = a.length;</code>                                             |
| 32    | RM8   | Die Variable <code>End</code> muss klein geschrieben werden.                                                                                                         |

- (b) Entspricht die Methode `binarySearch` ihrer Spezifikation, die durch Vor- und Nachbedingungen angegeben ist? Geben Sie gegebenenfalls Korrekturen der Methode an.

Lösungsvorschlag

### Korrektur der Vorbedingung

Die Vorbedingung ist nicht erfüllt, da weder die Länge des Feldes `a` noch die Reihenfolge der Feldeinträge geprüft wurden.

```

if (a.length <= 0) {
 return -1;
}

for (int i = 0; i < a.length; i++) {
 if (a[i] > a[i + 1]) {
 return -1;
 }
}

```

### Korrektur der Nachbedingung

`int start` muss mit `0` initialisiert werden, da sonst `a[0]` vernachlässigt wird.

- (c) Beschreiben alle Kommentare ab Zeile 24 die Semantik des Codes korrekt? Geben Sie zu jedem falschen Kommentar einen korrigierten Kommentar mit Zeilennummer an.

| Zeile | Kommentar im Code                                                                                               | Korrektur                                                                                                                                           |
|-------|-----------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 34-35 | <code>// Die Schleife wird v<br/>erlassen, wenn keine der<br/>beiden Haelften das Elemen<br/>t enthaelt.</code> | <code>// Die Schleife wird v<br/>erlassen, wenn keine der<br/>beiden Haelften das El<br/>ement enthaelt oder das<br/>Element gefunden wurde.</code> |
| 44    | <code>// Ausschluss der oberen<br/>Haelfte</code>                                                               | <code>// Ausschluss der unteren<br/>Haelfte</code>                                                                                                  |
| 47    | <code>// Ausschluss der unteren<br/>Haelfte</code>                                                              | <code>// Ausschluss der oberen<br/>Haelfte</code>                                                                                                   |
| 50    | <code>// Das gesuchte Element<br/>wird zurueckgegeben</code>                                                    | <code>// Der Index des gesuchten<br/>Elements wird zurueckgeb<br/>en</code>                                                                         |

- (d) Geben Sie den Kontrollflussgraphen für die Methode `binarySearch` an.
- (e) Geben Sie maximal drei Testfälle für die Methode `binarySearch` an, die insgesamt eine vollständige Anweisungsüberdeckung leisten.

Die gegebene Methode: `binarySearch(a[], item)`

### Testfall

- (i) Testfall: `a[] = {1, 2, 3}, item = 4`
- (ii) Testfall: `a[] = {1, 2, 3}, item = 2`

## 66116 / 2017 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Gegeben ist das folgende Gantt-Diagramm zur Planung eines hypothetischen Softwareprojekts:

- (a) Konvertieren Sie das Gantt-Diagramm in ein CPM-Netzwerk, das die Aktivitäten und Abhängigkeiten äquivalent beschreibt. Gehen Sie von der Zeiteinheit „Monate“ aus. Definieren Sie im CPM-Netzwerk je einen globalen Start- und Endknoten. Der Start jeder Aktivität hängt dabei vom Projektstart ab, das Projektende hängt vom Ende aller Aktivitäten ab.
- (b) Berechnen Sie für jedes Ereignis (öfür jeden Knoten Ihres CPM-Netzwerks) die früheste Zeit, die späteste Zeit sowie die Pufferzeit. Beachten Sie, dass die Berechnungsreihenfolge einer topologischen Sortierung des Netzwerks entsprechen sollte.
- (c) Geben Sie einen kritischen Pfad durch das CPM-Netzwerk an. Welche Aktivität darf sich demnach wie lange verzögern?

## 66116 / 2017 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Aktivitätsdiagramm  
Klassendiagramm  
Objektdiagramm

Gegeben sei das folgende Glossar, welches die statische Struktur von einfachen Aktivitätsdiagrammen in natürlicher Sprache beschreibt:

**Aktivitätsdiagramm:** Benannter Container für Aktivitäten und Datenflüsse. Eine der definierten Aktivitäten ist als Start-Aktivität ausgezeichnet.

**Aktivität:** Teil des beschriebenen Verhaltens. Man unterscheidet Start-, End-, echte Aktivitäten sowie Entscheidungen. Aktivitäten können generell mehrere ein- und auslaufende Kontrollflüsse haben.

**Startaktivität:** Ist im Aktivitätsdiagramm eindeutig und dient als Einstiegspunkt des beschriebenen Ablaufs.

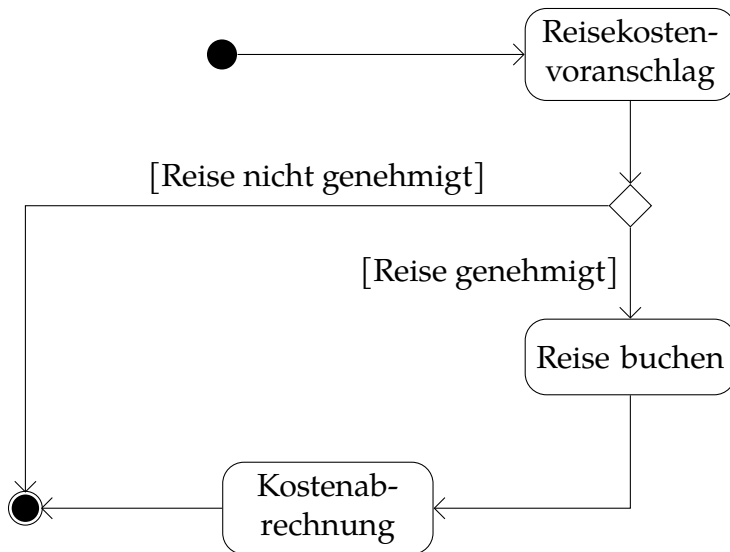
**Endaktivität:** Wird eine solche Aktivität erreicht, ist der beschriebene Ablauf zu Ende.

**Echte Aktivität:** Benannte Aktion, die nach Ausführung zu einer definierten nächsten Aktivität führt.

**Entscheidung:** Aktivität, die mehrere Nachfolger hat. Welche davon als nächstes ausgeführt wird, wird durch entsprechende Bedingungen (s. Kontrollfluss) gesteuert.

**Kontrollfluss:** Verbindet je eine Quell- mit einer Zielaktivität. Kann eine Bedingung enthalten, die erfüllt sein muss, damit die Zielaktivität im Falle einer Entscheidung ausgeführt wird.

- (a) Geben Sie ein UML-Klassendiagramm an, welches die im Glossar definierten Konzepte und Beziehungen formal beschreibt. Geben Sie bei allen Attributen und Assoziationsenden deren Sichtbarkeit, Multiplizität imd Typ an. Benennen Sie alle Assoziationen.
- (b) Nachfolgend ist ein Beispiel eines Aktivitätsdiagramms in der gängigen grafischen Notation abgebildet. Stellen Sie den beschriebenen Kontrollfluss als UML-Objektdiagramm konform zum in Teilaufgabe a erstellten UML-Klassendiagramm dar. Referenzieren Sie die dort definierten Klassen und Assoziationen; auf Objektbezeichner dürfen Sie verzichten.



## 66116 / 2018 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2

Das Fremdenverkehrsamt will sich einen besseren Überblick über Zirkusse verschaffen. In einer Datenbank sollen dazu die Zirkusse, die angebotenen Vorstellungen, die einzelnen Darbietungen in einer Vorstellung sowie die zugehörigen Dompteure und Tiere verwaltet werden.

Ein Zirkus wird durch seinen Namen gekennzeichnet und hat einen Besitzer. Vorstellungen haben eine VorstellungID und ein Datum. Darbietungen haben neben der eindeutigen ProgrammNr eine Uhrzeit. Ein Dompteur hat eine eindeutige AngestelltenNr sowie einen Künstlernamen. Tiere sind eindeutig durch eine TierNr bestimmt und haben außerdem eine Bezeichnung der Tierart.

Ein Zirkus bietet Vorstellungen an und stellt Dompteure an. Eine Darbietung findet in einer Vorstellung statt. Des weiteren trainiert ein Dompteur Tiere. In einer Darbietung tritt ein Dompteur mit Tieren auf.

(a) 1.1

(i) Listen Sie die Entity-Typen und die zugehörigen Attribute auf.

- \* Zirkusse (Zirkus-Nummer, Namen)
  - \* Besitzer
  - \* Namen
- \* Vorstellungen (VorstellungID)
  - \* VorstellungID
  - \* Datum
- \* Darbietungen (ProgrammNr)
  - \* ProgrammNr
  - \* Datum
- \* Dompteure (AngestelltenNr)
  - \* AngestelltenNr
  - \* Künstlernamen

```
* Tiere (TierNr)
* TierNr
* Tierart
```

- (ii) Bestimmen Sie zu jedem Entity-Typen einen Schlüssel. Fügen Sie, wenn nötig einen künstlichen Schlüssel hinzu.

```
Zirkus (ZID, Besitzer, Name)
Vorstellung (VorstellungsID, Datum, ZID[Zirkus])
Darbietung (ProgrammNr, VorstellungsID[Vorstellung], Uhrzeit)
Dompteur (AngestelltenNr, Kuenstlername, ZID[Zirkus])
Tier (TierNr, Tierart)

trainiert (AngestelltenNr[Dompteur], TierNr[Tier])
trittAuf (AngestelltenNr[Dompteur], TierNr[Tier], ProgrammNr[Darbietung],
 → VorstellungsID[Vorstellung])
```

- (iii) Erstellen Sie das ER-Diagramm!

Vorstellungen werden von genau einem Zirkus angeboten. Ein Zirkus bietet mehrere Vorstellungen an und stellt mehrere Dompteure an. Ein Dompteur ist genau bei einem Zirkus angestellt. Eine Darbietung findet in einer bestimmten Vorstellung statt. Des weiteren trainiert ein Dompteur mehrere Tiere, ein Tier kann allerdings auch von mehreren Dompteuren trainiert werden. In einer Darbietung tritt genau ein Dompteur mit mindestens einem Tier auf.

- (b) 1.2 Ergänzen Sie die Funktionalitäten im ER-Diagramm.

Vorstellungen werden von genau einem Zirkus angeboten. Ein Zirkus bietet mehrere Vorstellungen an und stellt mehrere Dompteure an. Ein Dompteur ist genau bei einem Zirkus angestellt. Eine Darbietung findet in einer bestimmten Vorstellung statt. Des weiteren trainiert ein Dompteur mehrere Tiere, ein Tier kann allerdings auch von mehreren Dompteuren trainiert werden. In einer Darbietung tritt genau ein Dompteur mit mindestens einem Tieren auf.

- (c) 1.3

- (i) Was bedeutet „mehrere“?

- (ii) Ergänzen Sie die Kardinalitäten in min-max Notation im ER-Diagramm.

## 66116 / 2018 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 4

Gegeben sei das folgende Datenbank-Schema, das für die Speicherung der Daten einer Universität entworfen wurde, zusammen mit einem Teil seiner Ausprägung. Die Primärschlüssel-Attribute sind jeweils unterstrichen. Die Relation *Dozent* enthält allgemeine Daten zu den Dozentinnen und Dozenten. Dozentinnen und Dozenten halten Vorlesungen, die in der Relation *Vorlesung* abgespeichert sind. Wir gehen davon aus, dass es zu jeder Vorlesung genau einen Dozenten (und nicht mehrere) gibt. Zusätzlich wird in der Relation *Vorlesung* das *Datum* gespeichert, an dem die Klausur stattfindet. In der Relation *Student* werden die Daten der teilnehmenden Studierenden verwaltet,

während die Relation *besucht* Auskunft darüber gibt, welche Vorlesung von welchen Studierenden besucht wird.

Dozent (DNR, DVorname, DNachname, DTitel)  
Vorlesung (VNR, VTitel, Klausurtermin, Dozent)  
Student (Matrikelnummer, SVorname, SNachname, Semesterzahl)  
*besucht* (Student, Vorlesung)

Formulieren Sie die folgenden Anfragen im Tupelkalkül. Datumsvergleiche können Sie mit  $>$ ,  $\geq$ ,  $<$ ,  $\leq$  oder  $=$  angeben:

- (a) Geben Sie die Vornamen aller Studierenden aus, die die Vorlesung „Datenbanksysteme“ besuchen oder besucht haben.

Lösungsvorschlag

$$\{s.SVorname \mid s \in Student \wedge \forall v \in Vorlesung (v.VTitel = 'Datenbanksysteme' \Rightarrow \exists b \in besucht (b.Vorlesung = v.VNR \wedge b.Student = s.Matrikelnummer))\}$$

oder

Lösungsvorschlag

$$\{s.SVorname \mid s \in Student \wedge s.Matrikelnummer = b.Student \wedge b \in besucht \wedge b.Vorlesung = v.VNR \wedge v.VNR \in Vorlesung \wedge v.VTitel = 'Datenbanksysteme'\}$$

- (b) Geben Sie die Matrikelnummern der Studierenden an, die keine Vorlesung mit einem Klausurtermin nach dem 31.12.2017 besuchen oder besucht haben.

Lösungsvorschlag

$$\{s.Matrikelnummer \mid \\ s \in Student \wedge \forall v \in Vorlesung ( \\ v.Klausurtermin > '31.12.2017' \Rightarrow \\ b \in besucht ( \\ b.Vorlesung = v.VNR \wedge b.Student = s.Matrikelnummer \\ ) \\ )\}$$

- (c) Geben Sie die Matrikelnummern der Studierenden aus, die alle Vorlesungen von Prof. Dr. Schulz hören oder gehört haben.

## 66116 / 2018 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 6

Gegeben sei das Relationenschema  $R(A, B, C, D, E, F)$ , sowie die Menge der zugehörigen funktionalen Abhängigkeiten  $F$ .

$$FA = \left\{ \begin{array}{l} \{ C \} \rightarrow \{ B \}, \\ \{ B \} \rightarrow \{ A \}, \\ \{ C, E \} \rightarrow \{ D \}, \\ \{ E \} \rightarrow \{ F \}, \\ \{ C, E \} \rightarrow \{ F \}, \\ \{ C \} \rightarrow \{ A \}, \end{array} \right\}$$

- (a) Bestimmen Sie den Schlüsselkandidaten der Relation  $R$  und begründen Sie, warum es keine weiteren Schlüsselkandidaten gibt.

Lösungsvorschlag

$C$  und  $E$  kommen auf keiner rechten Seite vor. Sie müssen deshalb immer Teil des Schlüsselkandidaten sein.

$$\text{AttrHülle}(F, \{C, E\}) = \{A, B, C, D, E, F\}$$

Daraus folgt, dass  $\{C, E\}$  ein Superschlüssel ist.

$$\text{AttrHülle}(F, \{C, E \setminus E\}) = \{A, B, C\} \neq R$$

$$\text{AttrHülle}(F, \{C, E \setminus C\}) = \{E, F\} \neq R$$

$\{C, E\}$  kann nicht weiter minimiert werden.

- (b) Überführen Sie das Relationenschema  $R$  mit Hilfe des Synthesealgorithmus in die dritte Normalform. Führen Sie hierfür jeden der vier Schritte durch und kennzeichnen Sie Stellen, bei denen nichts zu tun ist.

Lösungsvorschlag

### - Kanonische Überdeckung

— Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden. —

### - Linksreduktion

— Führe für jede funktionale Abhängigkeit  $\alpha \rightarrow \beta \in F$  die Linksreduktion durch, überprüfe also für alle  $A \in \alpha$ , ob  $A$  überflüssig ist, d. h. ob  $\beta \subseteq \text{AttrHülle}(F, \alpha - A)$ . —

$$\{C, E\} \rightarrow \{D\}$$

$$D \notin \text{AttrHülle}(F, \{C, E \setminus E\}) = \{A, C, B\}$$

$$D \notin \text{AttrHülle}(F, \{C, E \setminus C\}) = \{E, F\}$$

$$\{C, E\} \rightarrow \{F\}$$

$$F \notin \text{AttrHülle}(F, \{C, E \setminus E\}) = \{A, C, B\}$$

$$F \in \text{AttrHülle}(F, \{C, E \setminus C\}) = \{E, F\}$$

$$FA = \left\{ \right.$$



$$\begin{aligned}
 & \{C\} \rightarrow \{B\}, \\
 & \{B\} \rightarrow \{A\}, \\
 & \{C, E\} \rightarrow \{D\}, \\
 & \{E\} \rightarrow \{F\}, \\
 & \{E\} \rightarrow \{F\}, \\
 & \{C\} \rightarrow \{A\},
 \end{aligned}
 \}$$

### - Rechtsreduktion

— Führe für jede (verbliebene) funktionale Abhängigkeit  $\alpha \rightarrow \beta$  die Rechtsreduktion durch, überprüfe also für alle  $B \in \beta$ , ob  $B \in \text{AttrHülle}(F - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B)), \alpha)$  gilt. In diesem Fall ist  $B$  auf der rechten Seite überflüssig und kann eliminiert werden,  $\alpha \rightarrow \beta$  wird durch  $\alpha \rightarrow (\beta - B)$  ersetzt. —

#### A

$$A \notin \text{AttrHülle}(F \setminus \{B\} \rightarrow \{A\}, \{B\}) = \{B\}$$

$$A \in \text{AttrHülle}(F \setminus \{C\} \rightarrow \{A\}, \{C\}) = \{A, B, C\}$$

$$\begin{aligned}
 \text{FA} = \{ & \\
 & \{C\} \rightarrow \{B\}, \\
 & \{B\} \rightarrow \{A\}, \\
 & \{C, E\} \rightarrow \{D\}, \\
 & \{E\} \rightarrow \{F\}, \\
 & \{E\} \rightarrow \{F\}, \\
 & \{C\} \rightarrow \{\emptyset\},
 \end{aligned}
 \}$$

#### F

$$F \in \text{AttrHülle}(F \setminus \{E\} \rightarrow \{F\}, \{E\}) = \{E, F\}$$

$$\begin{aligned}
 \text{FA} = \{ & \\
 & \{C\} \rightarrow \{B\}, \\
 & \{B\} \rightarrow \{A\}, \\
 & \{C, E\} \rightarrow \{D\}, \\
 & \{E\} \rightarrow \{\emptyset\}, \\
 & \{E\} \rightarrow \{F\}, \\
 & \{C\} \rightarrow \{\emptyset\},
 \end{aligned}
 \}$$

### - Löschen leerer Klauseln

— Entferne die funktionalen Abhängigkeiten der Form  $\alpha \rightarrow \emptyset$ , die im 2. Schritt möglicherweise entstanden sind. —

$$FA = \left\{ \begin{array}{l} \{C\} \rightarrow \{B\}, \\ \{B\} \rightarrow \{A\}, \\ \{C, E\} \rightarrow \{D\}, \\ \{E\} \rightarrow \{F\}, \end{array} \right\}$$

**- Vereinigung**

— Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form  $\alpha \rightarrow \beta_1, \dots, \alpha \rightarrow \beta_n$ , so dass  $\alpha \rightarrow \beta_1 \cup \dots \cup \beta_n$  verbleibt. —————

∅ Nichts zu tun

**- Relationsschemata formen**

— Erzeuge für jede funktionale Abhängigkeit  $\alpha \rightarrow \beta \in F_c$  ein Relationenschema  $\mathcal{R}_\alpha := \alpha \cup \beta$ .

$R_1(\underline{C}, B)$

$R_2(\underline{B}, A)$

$R_3(\underline{C}, \underline{E}, D)$

$R_4(\underline{E}, F)$

**- Schlüssel hinzufügen**

— Falls eines der in Schritt 2. erzeugten Schemata  $\mathcal{R}_\alpha$  einen Schlüsselkandidaten von  $\mathcal{R}$  bezüglich  $F_c$  enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten  $\mathcal{K} \subseteq \mathcal{R}$  aus und definiere folgendes zusätzliche Schema:  $\mathcal{R}_\mathcal{K} := \mathcal{K}$  und  $\mathcal{F}_\mathcal{K} := \emptyset$  —————

∅ Nichts zu tun

**- Entfernung überflüssiger Teilschemata**

— Eliminiere diejenigen Schemata  $\mathcal{R}_\alpha$ , die in einem anderen Relationenschema  $\mathcal{R}_{\alpha'}$  enthalten sind, d. h.  $\mathcal{R}_\alpha \subseteq \mathcal{R}_{\alpha'}$ . —————

∅ Nichts zu tun

## 66116 / 2018 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Gegeben sei das folgende Java-Programm:

```
class M {
 private boolean b;
 private F f;
 private A a;

 public void m() {
 f = new F();
 a = new A(f);
 b = true;
 }
}

class A {
```

```
private R r;
public A(I i) {
 r = i.createX();
}

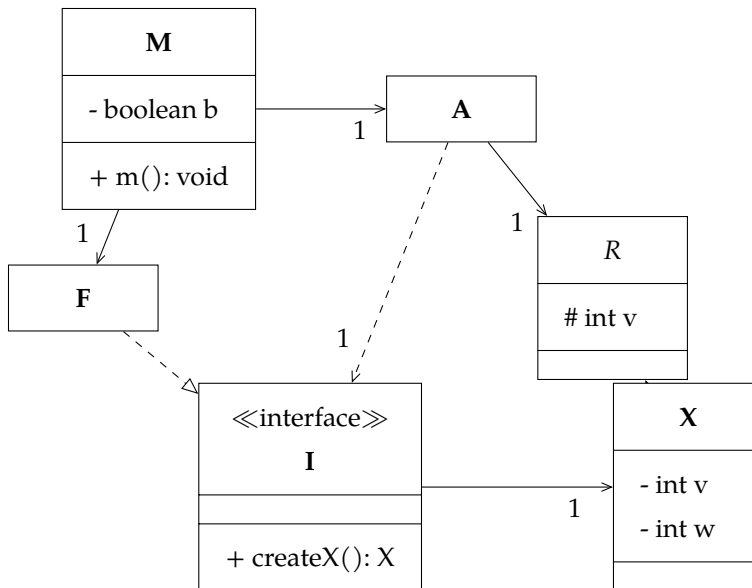
interface I {
 public X createX();
}

class F implements I {
 public X createX() {
 return new X(0, 0);
 }
}

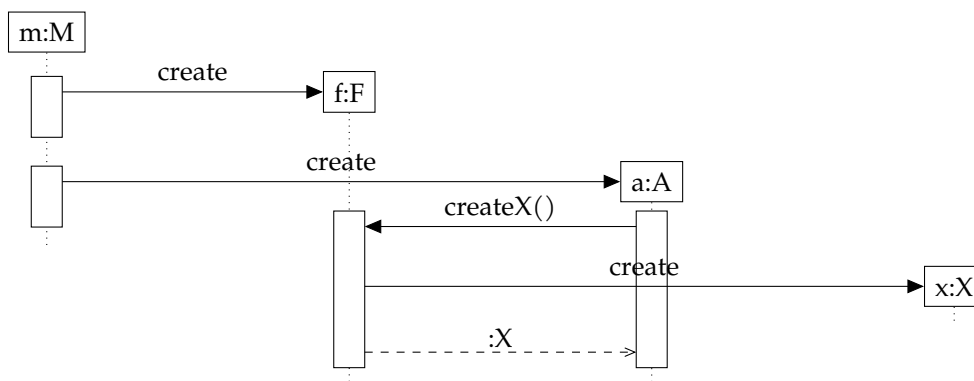
abstract class R {
 protected int v;
}

class X extends R {
 private int v, w;
 public X(int v, int w) {
 this.v = v;
 this.w = w;
 }
}
```

- (a) Das Subtypprinzip der objektorientierten Programmierung wird in obigem Programmcode zweimal ausgenutzt. Erläutern Sie wo und wie dies geschieht.
- (b) Zeichnen Sie ein UML-Klassendiagramm, das die statische Struktur des obigen Programms modelliert. Instanzvariablen mit einem Klassentyp sollen durch gerichtete Assoziationen mit Rollennamen und Multiplizität am gerichteten Assoziationsende modelliert werden. Alle aus dem Programmcode ersichtlichen statischen Informationen (insbesondere Interfaces, abstrakte Klassen, Zugriffsrechte, benutzerdefinierte Konstruktoren und Methoden) sollen in dem Klassendiagramm abgebildet werden.



- (c) Es wird angenommen, dass ein Objekt der Klasse M existiert, für das die Methode `m()` aufgerufen wird. Geben Sie ein Instanzendiagramm (Objektdiagramm) an, das alle nach der Ausführung der Methode `m` existierenden Objekte und deren Verbindungen (Links) zeigt.
- (d) Wie in Teil c) werde angenommen, dass ein Objekt der Klasse M existiert, für das die Methode `m()` aufgerufen wird. Diese Situation wird in Abb. 1 dargestellt. Zeichnen Sie ein Sequenzdiagramm, das Abb. 1 so ergänzt, dass alle auf den Aufruf der Methode `m()` folgenden Objekterzeugungen und Interaktionen gemäß der im Programmcode angegebenen Konstruktor- und Methodenrumpfe dargestellt werden. Aktivierungsphasen von Objekten sind durch längliche Rechtecke deutlich zu machen.

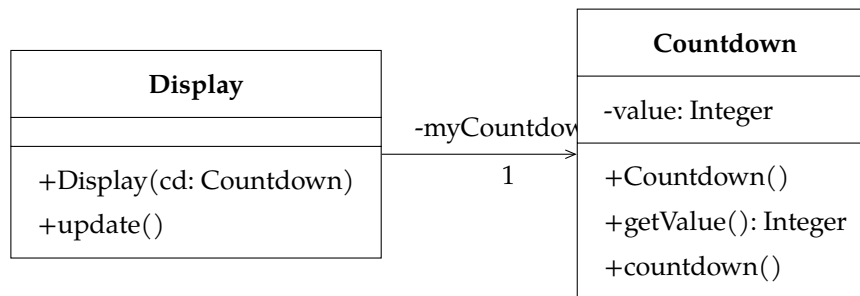


## 66116 / 2018 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 2

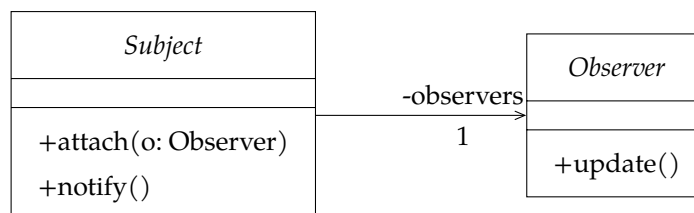
Es soll eine (kleine) Anwendung entwickelt werden, in der ein Zähler in 1-er Schritten von 5000 bis 0 herunterzählt. Der Zähler soll als Objekt der Klasse `Countdown` realisiert werden, die in UML-Notation dargestellt ist. Das Attribut `value` soll den aktuellen Zählerstand speichern, der mit dem Konstruktor zu initialisieren ist. Die Methode `getValue` soll den aktuellen Zählerstand liefern und die Methode `countdown` soll den

Zähler von 5000 bis 0 herunterzählen.

Der jeweilige Zählerstand soll von einem Objekt der in untenstehender Abbildung angegebenen Klasse `Display` am Bildschirm ausgegeben werden. Bei der Konstruktion eines `Display`-Objekts soll es mit einem `Countdown`-Objekt verbunden werden, indem dessen Referenz unter `myCountdown` abgespeichert wird. Die Methode `update` soll den aktuellen Zählerstand vom `Countdown`-Objekt holen und mit `System.out.println` am Bildschirm ausgeben. Dies soll zu Beginn des Zählprozesses und nach jeder Änderung des Zählerstands erfolgen.



Damit das `Display`-Objekt über Zählerstände des `Countdown`-Objekts informiert wird, soll das Observer-Pattern angewendet werden. Untenstehende Abbildung zeigt die im Observer-Pattern vorkommenden abstrakten Klassen. (Kursivschreibweise bedeutet abstrakte Klasse bzw. abstrakte Methode.)



- (a) Welche Wirkung haben die Methoden `attach` und `notify` gemäß der Idee des Observer-Patterns?

Lösungsvorschlag

Das beobachtete Objekt bietet mit der Methode `attach` einen Mechanismus, um Beobachter anzumelden und diese über Änderungen zu informieren. Mit der Methode `notify` werden alle Beobachter benachrichtigt, wenn sich das beobachtete Objekt ändert.

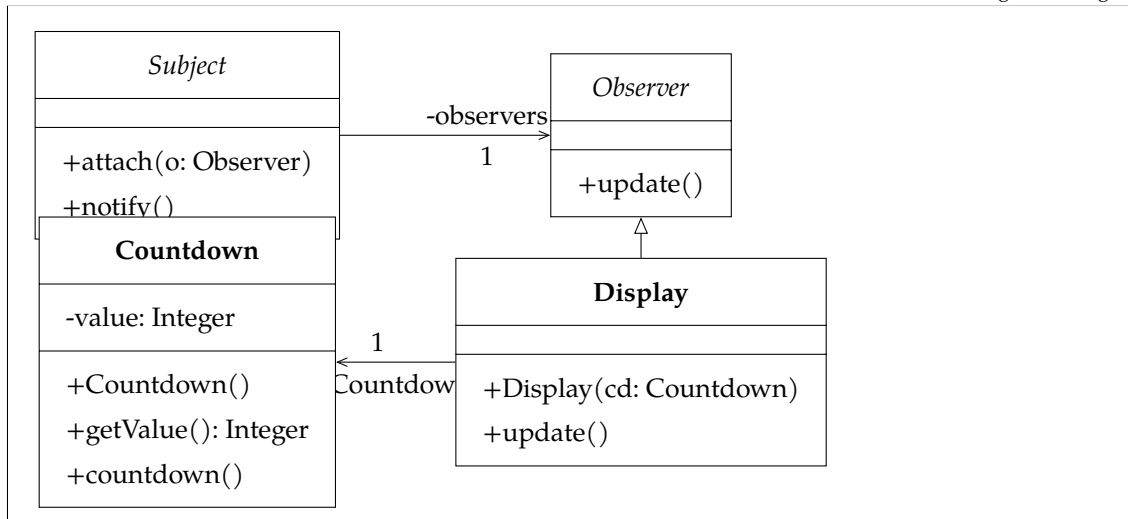
- (b) Welche der beiden Klassen `Display` und `Countdown` aus obenstehender Abbildung spielt die Rolle eines `Subject` und welche die Rolle eines `Observer`?

Lösungsvorschlag

Die Klasse `Countdown` spielt die Rolle des `Subject`s, also des Gegenstands, der beobachtet wird. Die Klasse `Display` spielt die Rolle eines `Observer`, also die Rolle eines Beobachters.

- (c) Erstellen Sie ein Klassendiagramm, das die beiden obenstehenden gegebenen Diagramme in geeigneter Weise, öentsprechend der Idee des Observer-Patterns, zusammenfügt. Es reicht die Klassen und deren Beziehungen anzugeben. Eine nochmalige Nennung der Attribute und Methoden ist nicht notwendig.

Lösungsvorschlag



- (d) Unsere Anwendung soll nun in einer objektorientierten Programmiersprache Ihrer Wahl (z. B. Java oder C++) implementiert werden. Dabei soll von folgenden Annahmen ausgegangen werden:

- Das Programm wird mit einer main-Methode gestartet, die folgenden Rumpf hat:

```

public static void main(String[] args){
 Countdown cd = new Countdown();
 new Display(cd);
 cd.countdown();
}

```

- Die beiden Klassen Subject und Observer sind bereits gemäß der Idee des Observer-Patterns implementiert.

Geben Sie auf dieser Grundlage eine Implementierung der beiden Klassen `Display` und `Countdown` an, so dass das gewünschte Verhalten, öAnzeige der Zählerstände und Herunterzählen des Zählers, realisiert wird. Die Methoden der Klassen `Subject` und `Observer` sind dabei auf geeignete Weise zu verwenden bzw. zu implementieren. Geben Sie die verwendete Programmiersprache an.

Lösungsvorschlag

```

public class Client {
 public static void main(String[] args){
 Countdown cd = new Countdown();
 new Display(cd);
 cd.countdown();
 cd.countdown();
 cd.countdown();
 }
}

```

```
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2018/fruehjahr/Client.java](#)

```
import java.util.ArrayList;
import java.util.List;

public abstract class Subject {
 private final List<Observer> observers = new ArrayList<Observer>();

 public void attach(Observer o) {
 observers.add(o);
 }

 public void notifyObservers() {
 for (Observer o : observers) {
 o.update();
 }
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2018/fruehjahr/Subject.java](#)

```
public abstract class Observer {
 public abstract void update();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2018/fruehjahr/Observer.java](#)

```
public class Countdown extends Subject {

 private int value;

 public Countdown() {
 value = 5000;
 }

 public int getValue() {
 return value;
 }

 public void countdown() {
 if (value > 0) {
 notifyObservers();
 value--;
 }
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2018/fruehjahr/Countdown.java](#)

```
public class Display extends Observer {
 Countdown myCountdown;
 public Display(Countdown cd) {
 myCountdown = cd;
 myCountdown.attach(this);
 }

 public void update() {
 System.out.println(myCountdown.getValue());
 }
}
```

Zustand (State)

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2018/fruehjahr/Display.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2018/fruehjahr/Display.java)

## 66116 / 2018 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Gegeben sei das Java-Programm:

```
public class Music {
 private Beatle state;

 public Music() {
 state = new Paul();
 }

 public void help() {
 this.state.help(this);
 }

 public void obladi() {
 this.state.obladi(this);
 }

 public void yesterday() {
 this.state.yesterday(this);
 }

 public void setBeatle(Beatle b) {
 state = b;
 }
}

abstract class Beatle {
 public abstract void help(Music m);

 public abstract void obladi(Music m);

 public abstract void yesterday(Music m);
}

class George extends Beatle {
 public void help(Music m) {
 System.out.println("help");
 m.setBeatle(new John());
 }
}
```



```
}

public void obladi(Music m) {
}

public void yesterday(Music m) {
 System.out.println("yesterday");
 m.setBeatle(new Paul());
}
}

class John extends Beatle {
 public void help(Music m) {
 System.out.println("help");
 m.setBeatle(new Paul());
 }

 public void obladi(Music m) {
 System.out.println("obladi");
 m.setBeatle(new Ringo());
 }

 public void yesterday(Music m) {
 }
}

class Paul extends Beatle {
 public void help(Music m) {
 System.out.println("help");
 m.setBeatle(new George());
 }

 public void obladi(Music m) {
 }

 public void yesterday(Music m) {
 System.out.println("yesterday");
 }
}

class Ringo extends Beatle {
 public void help(Music m) {
 System.out.println("help");
 m.setBeatle(new John());
 }

 public void obladi(Music m) {
 }

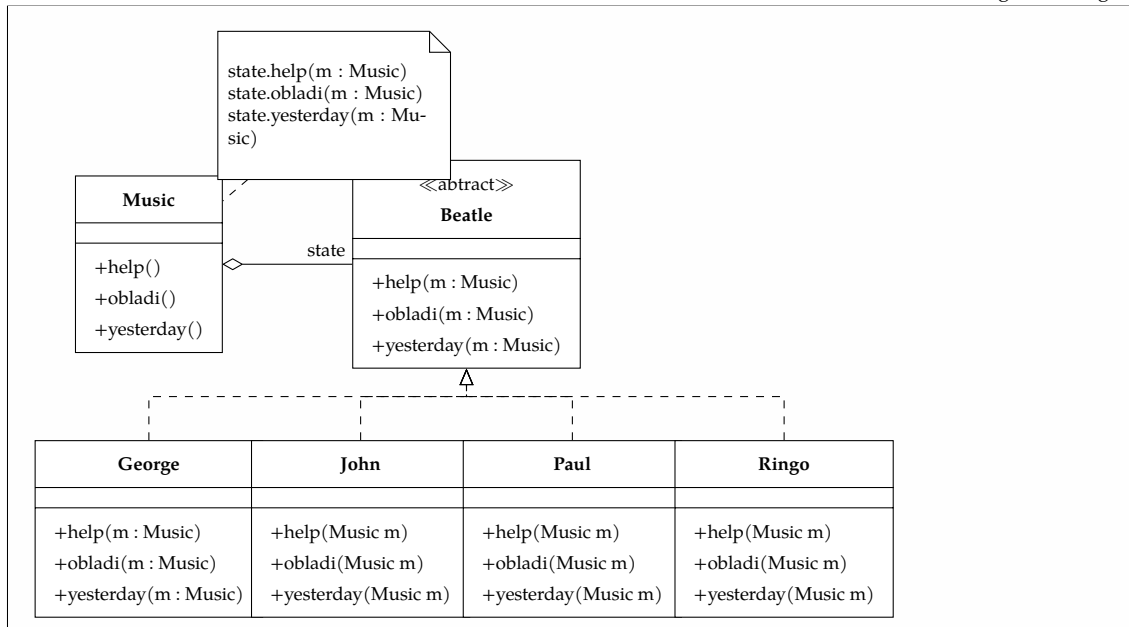
 public void yesterday(Music m) {
 System.out.println("yesterday");
 m.setBeatle(new Paul());
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2018/herbst/beatles/Music.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2018/herbst/beatles/Music.java)

- (a) Zeichnen Sie ein UML-Klassendiagramm, das die statische Struktur des Programms modelliert. Instanzvariablen mit einem Klassentyp sollen durch gerichtete Assoziationen mit Rollennamen und passender Multiplizität am gerichteten Assoziationsende modelliert werden. Alle aus dem Programmcode ersichtlichen statischen Informationen sollen in dem Klassendiagramm dargestellt werden.

Klassendiagramm zeichnen  
zustand (State)

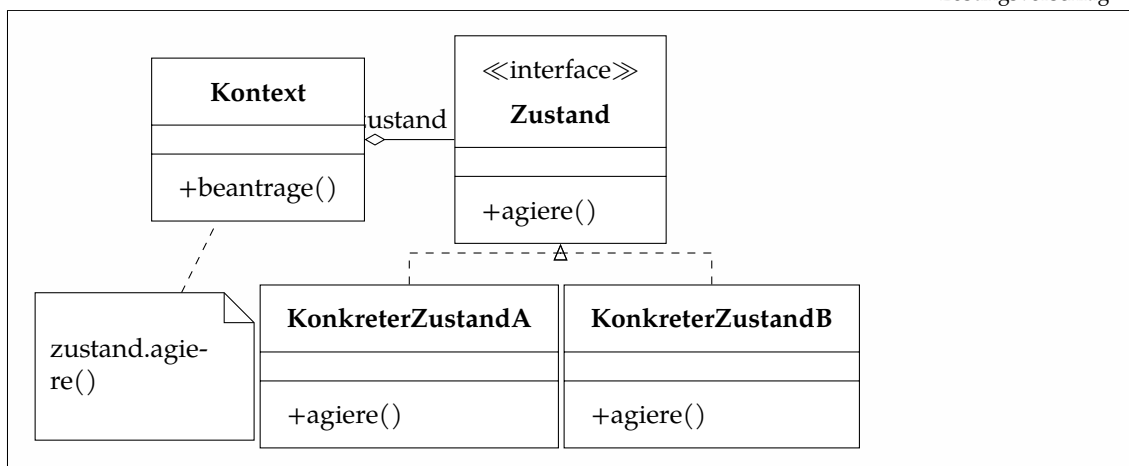
Lösungsvorschlag



Das Programm implementiert ein Zustandsdiagramm, das das Verhalten von Objekten der Klasse **Music** beschreibt. Für die Implementierung wurde das Design-Pattern STATE angewendet.

- (c) Geben Sie die statische Struktur des STATE-Patterns an und erläutern Sie, welche Rollen aus dem Entwurfsmuster den Klassen des gegebenen Programms dabei zufallen und welche Operationen aus dem Entwurfsmuster durch (ggf. mehrere) Methoden in unserem Beispielprogramm implementiert werden. Es ist von den z. B. im Design-Pattern-Katalog von Gamma et al. verwendeten Namen auszugehen, das heißt von Klassen mit Namen **Context**, **State**, **ConcreteStateA**, **ConcreteStateB** und von Operationen mit Namen **request** und **handle**.

Lösungsvorschlag



**Kontext (Context)** definiert die clientseitige Schnittstelle und verwaltet die separaten Zustandsklassen.

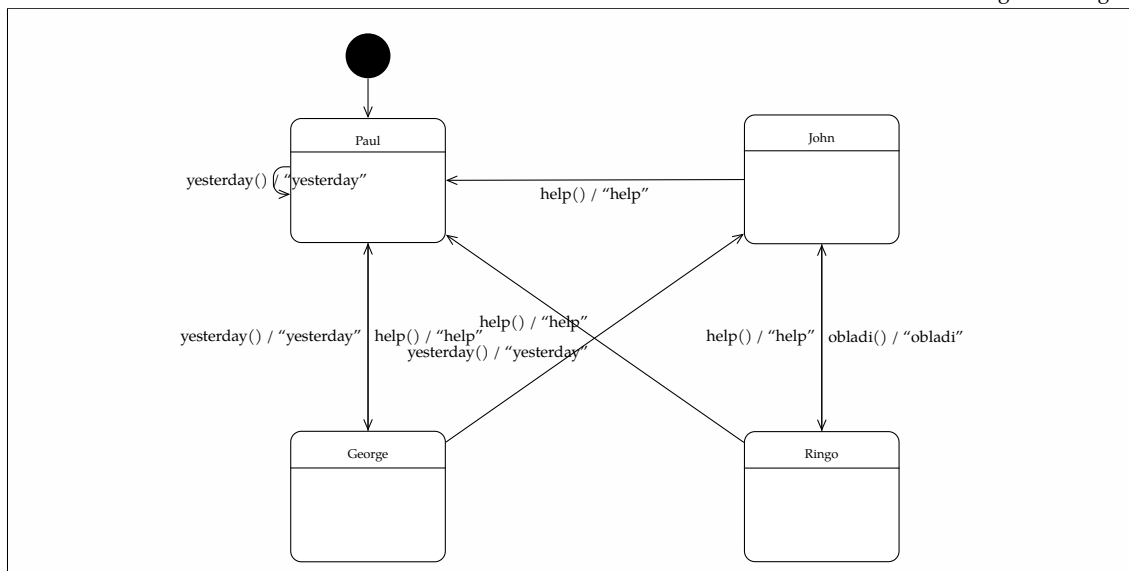
**State (Zustand)** definiert eine einheitliche Schnittstelle aller Zustandsobjekte und implementiert gegebenenfalls ein Standardverhalten.

**KontreterZustand (ConcreteState)** implementiert das Verhalten, das mit dem Zustand des Kontextobjektes verbunden ist.

Zustandsdiagramm zeichnen

- (d) Zeichnen Sie das UML-Zustandsdiagramm (mit Anfangszustand), das von dem Programm implementiert wird. Dabei muss - gemäß der UML-Notation - unterscheidbar sein, was Ereignisse und was Aktionen sind. In dem Diagramm kann zur Vereinfachung statt `System.out.println ("x")` einfach "x" geschrieben werden.

Lösungsvorschlag



## 66116 / 2018 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 4

Gegeben sind folgende Relationen aus einer Datenbank zur Verwaltung von Triathlon-Wettbewerben.

Athlet : {[ ID, Vorname, Nachname ]}

Ergebnis : {[ Athlet[Athlet], Wettbewerb[Wettbewerb], Schwimmzeit, Radzeit, Laufzeit ]}

Wettbewerb : {[ Name, Jahr ]}

```

CREATE TABLE Athlet (
 ID INTEGER PRIMARY KEY,
 Vorname VARCHAR(20),
 Nachname VARCHAR(20)
);

```

```

CREATE TABLE Wettbewerb (

```

```
Name VARCHAR(40) PRIMARY KEY,
Jahr INTEGER
);
```

```
CREATE TABLE Ergebnis (
 Athlet INTEGER REFERENCES Athlet(ID),
 Wettbewerb VARCHAR(40) REFERENCES Wettbewerb(Name),
 Schwimmzeit INTEGER NOT NULL,
 Radzeit INTEGER,
 Laufzeit INTEGER,
 PRIMARY KEY (Athlet, Wettbewerb)
);
```

```
INSERT INTO Athlet VALUES
(1, 'Boris', 'Stein'),
(2, 'Trevor', 'Wurtele'),
(3, 'Reichelt', 'Horst'),
(12, 'Mitch', 'Kibby');
```

```
INSERT INTO Wettbewerb VALUES
('Zürichsee', 2018),
('Ironman Vichy', 2018),
('Challenge Walchsee', 2018),
('Triathlon Alpe d'Huez', 2017);
```

```
INSERT INTO Ergebnis VALUES
(1, 'Zürichsee', 14, 10, 11),
(2, 'Zürichsee', 13, 10, 11),
(3, 'Zürichsee', 12, 10, 11),
(12, 'Zürichsee', 11, 10, 11),
(2, 'Challenge Walchsee', 12, 10, 11),
(3, 'Challenge Walchsee', 11, 10, 11),
(12, 'Triathlon Alpe d'Huez', 9, 10, 11);
```

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz, also eine Entity, nicht mehrfach aus.

- (a) Schreiben Sie eine SQL-Anweisung, die die Tabelle „Ergebnis“ anlegt. Gehen Sie davon aus, dass die Tabellen „Athlet“ und „Wettbewerb“ bereits existieren. Verwenden Sie sinnvolle Datentypen.

Lösungsvorschlag

```
CREATE TABLE IF NOT EXISTS Ergebnis (
 Athlet INTEGER REFERENCES Athlet(ID),
 Wettbewerb INTEGER REFERENCES Wettbewerb(Name),
 Schwimmzeit INTEGER NOT NULL,
 Radzeit INTEGER,
 Laufzeit INTEGER,
 PRIMARY KEY (Athlet, Wettbewerb)
);
```

- (b) Schreiben Sie eine SQL-Anweisung, die die Radzeit des Teilnehmers mit der ID 12 beim Wettbewerb „Zürichsee“ um eins erhöht.

Lösungsvorschlag

DESC  
AVG  
GROUP BY  
HAVING  
EXCEPT

```
-- Nur für Test-Zwecke
SELECT * FROM Ergebnis WHERE Athlet = 12 AND Wettbewerb = 'Zürichsee';

UPDATE Ergebnis
SET Radzeit = Radzeit + 1
WHERE Athlet = 12 AND Wettbewerb = 'Zürichsee';

-- Nur für Test-Zwecke
SELECT * FROM Ergebnis WHERE Athlet = 12 AND Wettbewerb = 'Zürichsee';
```

- (c) Schreiben Sie eine SQL-Anweisung, die die Namen aller Wettbewerbe des Jahres 2018 ausgibt, absteigend sortiert nach Name.

Lösungsvorschlag

```
SELECT Name
FROM Wettbewerb
WHERE Jahr = 2018
ORDER BY Name DESC;
```

- (d) Schreiben Sie eine SQL-Anweisung, die die Namen aller Wettbewerbe ausgibt, in der die durchschnittliche Schwimmzeit größer als 10 ist.

Lösungsvorschlag

```
SELECT Wettbewerb, AVG(Schwimmzeit)
FROM Ergebnis
GROUP BY Wettbewerb
HAVING AVG(Schwimmzeit) > 10;
```

- (e) Schreiben Sie eine SQL-Anweisung, die die IDs aller Athleten ausgibt, die im Jahr 2017 an keinem Wettbewerb teilgenommen haben.

Lösungsvorschlag

```
(SELECT DISTINCT Athlet FROM Ergebnis)
EXCEPT
(SELECT DISTINCT Athlet FROM Ergebnis e, Wettbewerb w
WHERE e.Wettbewerb = w.name AND w.Jahr = 2017);
```

- (f) Schreiben Sie eine SQL-Anweisung, die die Nachnamen aller Athleten ausgibt, die mindestens 10 Wettbewerbe gewonnen haben, das heißt im jeweiligen Wettbewerb die kürzeste Gesamtzeit erreicht haben. Die Gesamtzeit ist die Summe aus Schwimmzeit, Radzeit und Laufzeit. Falls zwei Athleten in einem Wettbewerb die gleiche Gesamtzeit erreichen, sind beide Sieger.

Lösungsvorschlag

```
vermutlich falsch

CREATE VIEW Gesamtzeiten AS
SELECT e.Athlet AS NameAthlet, e.Radzeit + e.Schwimmzeit + e.Laufzeit
AS Gesamtzeit, w.NameWettbewerb
FROM Ergebnis e, Wettbewerb w
WHERE e.Wettbewerb = w.Name
```

```
CREATE VIEW Sieger AS
SELECT g1.NameAthlet
FROM Gesamtzeiten g1, Gesamtzeiten g2
GROUP BY g1.NameWettbewerb
HAVING g1.Gesamtzeit < g2.Gesamtzeit
SELECT NameAthlet
FROM Sieger
GROUP BY NameAthlet
HAVING count(*) > 10;
```

- (g) Schreiben Sie eine SQL-Anweisung, die die Top-Ten der Athleten mit der schnellsten Schwimmzeit des Wettbewerbs „Paris“ ausgibt. Ausgegeben werden sollen die Platzierung (1 bis 10) und der Nachname des Athleten, aufsteigend sortiert nach Platzierung. Gehen Sie davon aus, dass keine zwei Athleten die gleiche Schwimmzeit haben und verwenden Sie keine produktspezifischen Anweisungen wie beispielsweise rownum, top oder limit.

Lösungsvorschlag

```
CREATE VIEW AthletenParis AS
SELECT a.Nachname, e.Schwimmzeit
FROM Athlet a, Ergebnis e INNER JOIN Wettbewerb W ON e.Wettbewerb = w.Name
WHERE w.Name = "Paris" AND a.ID = e.Athlet
SELECT a.Nachname COUNT(*) + 1 AS Platzierung
FROM AthletenParis a, AthletenParis b
WHERE a.Schwimmzeit < b.Schwimmzeit
GROUP BY a.Nachname
HAVING Platzierung <= 10;
```

- (h) Schreiben Sie einen Trigger, der beim Einfügen neuer Tupel in die Tabelle „Ergebnis“ die Schwimmzeit auf den Wert 0 setzt, falls diese negativ ist.

Lösungsvorschlag

```
CREATE TRIGGER update_Ergebnis AFTER UPDATE ON Ergebnis AS
IF(UPDATE Schwimmzeit AND Schwimmzeit < 0) BEGIN UPDATE Ergebnis
SET Schwimmzeit = 0
WHERE (Athlet, Wettbewerb) IN (SELECT DISTINCT (Athlet, Wettbewerb) FROM
↪ inserted)
END;
```

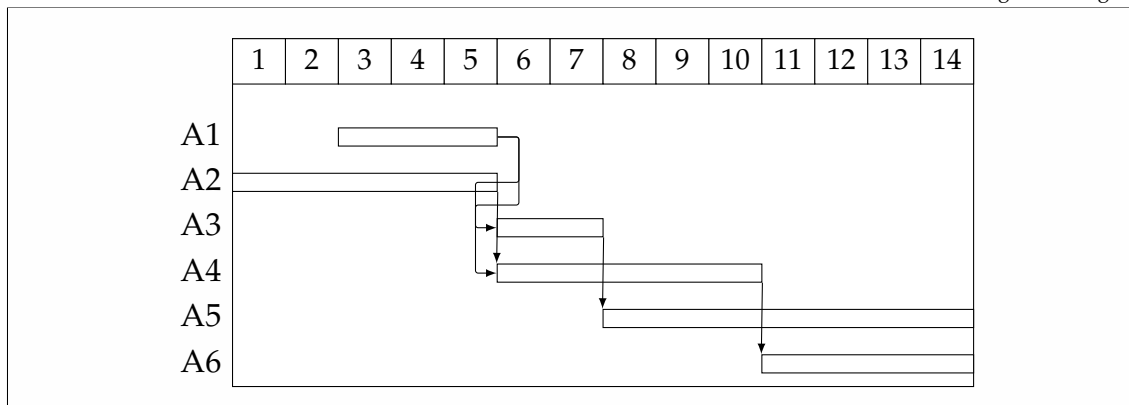
## 66116 / 2018 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1

Ein Team von zwei Softwareentwicklern soll ein Projekt umsetzen, das in sechs Arbeitspakete unterteilt ist. Die Dauer der Arbeitspakete und ihre Abhängigkeiten können Sie aus folgender Tabelle entnehmen:

| Name | Dauer in Wochen | Abhängig von |
|------|-----------------|--------------|
| A1   | 2               | -            |
| A2   | 5               | -            |
| A3   | 2               | A1           |
| A4   | 5               | A1, A2       |
| A5   | 7               | A3           |
| A6   | 4               | A4           |

- (a) Zeichnen Sie ein Gantt-Diagramm, das eine kürzestmögliche Projektabwicklung beinhaltet.

Lösungsvorschlag



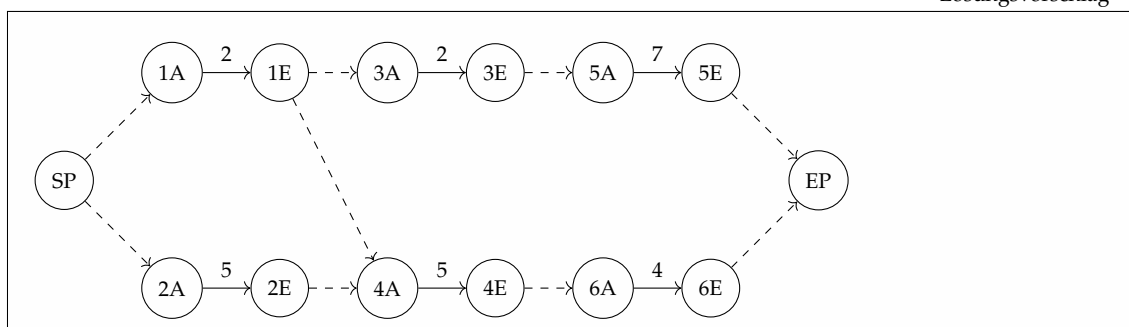
- (b) Bestimmen Sie die Länge des kritischen Pfades und geben Sie an, welche Arbeitspakete an ihm beteiligt sind.

Lösungsvorschlag

Auf dem kritischen Pfad befinden die Arbeitspakete A2, A4 und A6. Die Länge des kritischen Pfades ist 14.

- (c) Wandeln Sie das Gantt-Diagramm in ein CPM-Netzplan um.

Lösungsvorschlag



- (d) Berechnen Sie für jedes Ereignis den *frühesten Termin* und den *spätesten Termin* sowie die *Gesamtpufferzeiten*.

| <i>i</i>              | SP | 1A | 1E | 2A | 2E | 3A | 3E | 4A | 4E | 5A | 5E | 6A | 6E | EP |
|-----------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| <i>FZ<sub>i</sub></i> | 0  | 0  | 2  | 0  | 5  | 2  | 4  | 5  | 10 | 4  | 11 | 10 | 14 | 14 |
| <i>SZ<sub>i</sub></i> | 0  | 3  | 5  | 0  | 5  | 5  | 7  | 5  | 10 | 7  | 14 | 10 | 14 | 14 |
| <i>GP</i>             | 0  | 3  | 3  | 0  | 0  | 3  | 3  | 0  | 0  | 3  | 3  | 0  | 0  | 0  |

## 66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Gegeben sei der folgende Ausschnitt des Schemas für die Verwaltung der Einnahme von Medikamenten:

Person : {[ ID : INTEGER, Name : VARCHAR(255), Wohnort : VARCHAR(255) ]}

Hersteller : {[ ID : INTEGER, Name : VARCHAR(255), Standort : VARCHAR(255), AnzahlMitarbeiter : INTEGER ]}

Medikament : {[ ID : INTEGER Name : VARCHAR(255), Kosten : INTEGER, Wirkstoff : VARCHAR(255), produziert\_von : INTEGER ]}

nimmt : {[ Person : INTEGER, Medikament : INTEGER, von : DATE, bis : DATE ]}

hat\_Unverträglichkeit\_gegen : {[ Person : INTEGER, Medikament : INTEGER ]}

```
CREATE TABLE Person (
 ID INTEGER PRIMARY KEY,
 Name VARCHAR(255),
 Wohnort VARCHAR(255)
);
```

```
CREATE TABLE Hersteller (
 ID INTEGER PRIMARY KEY,
 Name VARCHAR(255),
 Standort VARCHAR(255),
 AnzahlMitarbeiter INTEGER
);
```

```
CREATE TABLE Medikament (
 ID INTEGER PRIMARY KEY,
 Name VARCHAR(255),
 Kosten INTEGER,
 Wirkstoff VARCHAR(255),
 produziert_von INTEGER REFERENCES Hersteller(ID)
);
```

```
CREATE TABLE nimmt (
 Person INTEGER,
 Medikament INTEGER,
 von DATE,
 bis DATE,
 PRIMARY KEY (Person, Medikament, von, bis)
);
```



```
);

CREATE TABLE hat_Unverträglichkeit_gegen (
 Person INTEGER REFERENCES Person(ID),
 Medikament INTEGER REFERENCES Medikament(ID)
);

INSERT INTO Person VALUES
(1, 'Walter Müller', 'Holzapfelkreuth'),
(2, 'Dilara Yildiz', 'Fürth');

INSERT INTO Hersteller VALUES
(1, 'Hexal', 'Holzkirchen', 3700),
(2, 'Ratiopharm', 'Ulm', 563);

INSERT INTO Medikament VALUES
(1, 'IbuHexal', 3, 'Ibuprofen', 1),
(2, 'Ratio-Paracetamol', 2, 'Paracetamol', 2),
(3, 'BudeHexal', 4, 'Budesonid', 1),
(4, 'Ratio-Budesonid', 5, 'Budesonid', 2);

INSERT INTO nimmt VALUES
(1, 1, '2021-07-12', '2021-07-23'),
(1, 3, '2021-07-12', '2021-07-23'),
(2, 4, '2021-02-13', '2021-03-24');

INSERT INTO hat_Unverträglichkeit_gegen VALUES
(1, 1),
(1, 3),
(2, 2);
```

Die Tabelle `Person` beschreibt Personen über eine eindeutige ID, deren Namen und Wohnort. Die Tabelle `Medikament` enthält Informationen über Medikamente, unter anderem deren Namen, Kosten, Wirkstoffe und einen Verweis auf den Hersteller dieses Medikaments. Die Tabelle `Hersteller` verwaltet verschiedene Hersteller von Medikamenten. Die Tabelle `hat_Unverträglichkeit_gegen` speichert die IDs von Personen zusammen mit den IDs von Medikamenten, gegen die diese Person eine Unverträglichkeit hat. Die Tabelle `nimmt` hingegen verwaltet die Einnahme der verschiedenen Medikamente und speichert zudem in welchem Zeitraum eine Person welches Medikament genommen hat bzw. nimmt.

Beachten Sie bei der Formulierung der SQL-Anweisungen, dass die Ergebnisrelationen keine Duplikate enthalten dürfen. Sie dürfen geeignete Views definieren.

- (a) Schreiben Sie SQL-Anweisungen, die für die bereits existierende Tabelle `nimmt` alle benötigten Fremdschlüsselconstraints anlegt. Erläutern Sie kurz, warum die Spalten `von` und `bis` Teil des Primärschlüssels sind.

Lösungsvorschlag

```
ALTER TABLE nimmt
ADD CONSTRAINT FK_Person
 FOREIGN KEY (Person) REFERENCES Person(ID),
```

```
ADD CONSTRAINT FK_Medikament
FOREIGN KEY (Medikament) REFERENCES Medikament(ID);
```

- (b) Schreiben Sie eine SQL-Anweisung, welche sowohl den Namen als auch die ID von Personen und Medikamenten ausgibt, bei denen die Person das jeweilige Medikament nimmt.

Lösungsvorschlag

```
SELECT DISTINCT
 p.ID as Personen_ID,
 p.Name as Personen_Name,
 m.ID as Medikamenten_ID,
 m.Name as Medikamenten_Name
FROM Person p, Medikament m, nimmt n
WHERE
 n.Person = p.ID AND
 n.Medikament = m.ID;
```

- (c) Schreiben Sie eine SQL-Anweisung, welche die ID und den Namen der Medikamente mit den niedrigsten Kosten je Hersteller bestimmt.

Lösungsvorschlag

```
SELECT m.ID, m.Name
FROM Medikament m, Medikament n
WHERE
 m.produziert_von = n.produziert_von AND
 m.Kosten >= n.Kosten
GROUP BY m.ID, m.Name
HAVING COUNT(*) <= 1;
```

- (d) Schreiben Sie eine SQL-Anweisung, die die Anzahl aller Personen ermittelt, die ein Medikament genommen haben, gegen welches sie eine Unverträglichkeit entwickelt haben.

Lösungsvorschlag

```
SELECT COUNT(DISTINCT p.ID)
FROM
 Person p,
 nimmt n,
 hat_Unverträglichkeit_gegen u
WHERE
 p.ID = n.Person AND
 u.Medikament = n.Medikament;
```

- (e) Schreiben Sie eine SQL-Anweisung, die die ID und den Namen derjenigen Personen ermittelt, die weder ein Medikament mit dem Wirkstoff Paracetamol noch ein Medikament mit dem Wirkstoff Ibuprofen genommen haben.

```

SELECT ID, Name FROM Person
EXCEPT
SELECT p.ID, p.Name
FROM Person p, nimmt n, Medikament m
WHERE
 p.ID = n.Person AND
 n.Medikament = m.ID AND
 m.Wirkstoff IN ('Ibuprofen', 'Paracetamol');

```

- (f) Schreiben Sie eine SQL-Anweisung, welche die Herstellernamen und die Anzahl der bekannten Unverträglichkeiten gegen Medikamente dieses Herstellers ermittelt. Das Ergebnis soll aufsteigend nach der Anzahl der bekannten Unverträglichkeiten sortiert werden.

```

SELECT p.Name, (
 SELECT COUNT(h.ID)
 FROM Hersteller h, Medikament m, hat_Unverträglichkeit_gegen u
 WHERE
 m.produziert_von = h.ID AND
 u.Medikament = m.ID AND
 h.ID = p.ID
) AS Unverträglichkeiten
FROM Hersteller p
ORDER BY Unverträglichkeiten ASC;

```

- (g) Formulieren Sie eine Anfrage in relationaler Algebra, welche die Wohnorte aller Personen bestimmt, welche ein Medikament mit dem Wirkstoff Paracetamol nehmen oder genommen haben. Die Lösung kann als Baum- oder als Term-Schreibweise angegeben werden.

$$\pi_{\text{Person.Wohnort}} (\sigma_{\text{Medikament.Wirkstoff} = \text{'Paracetamol'}} (\text{Person} \bowtie_{\text{Person.ID} = \text{nimmt.Person}} \text{nimmt} \bowtie_{\text{nimmt.Medikament}} \text{Medikament}))$$

- (h) Formulieren Sie eine Anfrage in relationaler Algebra, welche die Namen aller Personen bestimmt, die von allen bekannten Herstellern, deren Standort München ist, Medikamente nehmen oder genommen haben. Die Lösung kann als Baum- oder als Term-Schreibweise angegeben werden.

$$\pi_{\text{Person.Name}} (\sigma_{\text{Hersteller.Standort} = \text{'München'}} ((\text{Person} \bowtie_{\text{Person.ID} = \text{nimmt.Person}} \text{nimmt}) \bowtie_{\text{nimmt.Medikament}} \text{Medikament}))$$

## 66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Gegeben sei folgendes relationales Schema R in erster Normalform:

$$R : \{ [ A, B, C, D, E, F ] \}$$

Für  $R$  gelte folgende Menge  $FD$  funktionaler Abhängigkeiten:

$$FA = \left\{ \begin{array}{l} \{ A, D, F \} \rightarrow \{ E \}, \\ \{ B, C \} \rightarrow \{ A, E \}, \\ \{ D \} \rightarrow \{ B \}, \\ \{ D, E \} \rightarrow \{ C, B \}, \\ \{ A \} \rightarrow \{ F \}, \end{array} \right\}$$

- (a) Bestimmen Sie alle Kandidatenschlüssel/Schlüsselkandidaten von  $R$  mit  $FD$ . *Hinweis: Die Angabe von Attributmengen, die keine Kandidatenschlüssel sind, führt zu Abzügen.*

Lösungsvorschlag

- $\{ D, A \}$
- $\{ D, C \}$
- $\{ D, E \}$

- (b) Prüfen Sie, ob  $R$  mit  $FD$  in 2NF bzw. 3NF ist.

Lösungsvorschlag

$R$  ist in 1NF, da  $\{ d \} \rightarrow \{ b \}$

- (c) Bestimmen Sie mit folgenden Schritten eine kanonische Überdeckung  $FD_C$  von  $FD$ :

- (i) Führen Sie eine Linksreduktion von  $FD$  durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Linksreduktion an ( $FD_L$ ).

Lösungsvorschlag

### Linksreduktion

— Führe für jede funktionale Anhängigkeit  $\alpha \rightarrow \beta \in F$  die Linksreduktion durch, überprüfe also für alle  $A \in \alpha$ , ob  $A$  überflüssig ist, d. h. ob  $\beta \subseteq \text{AttrHülle}(F, \alpha - A)$ . —

$$\{ A, D, F \} \rightarrow \{ E \}$$

$$E \notin \text{AttrHülle}(F, \{ A, D, F \setminus A \}) = \{ D, E, B \}$$

$$E \notin \text{AttrHülle}(F, \{ A, D, F \setminus D \}) = \{ A, F \}$$

$$E \in \text{AttrHülle}(F, \{ A, D, F \setminus F \}) = \{ A, B, D, F \}$$

$$\{ B, C \} \rightarrow \{ A, E \}$$

$$\{ A, E \} \notin \text{AttrHülle}(F, \{ B, C \setminus B \}) = \{ C \}$$

$$\{ A, E \} \notin \text{AttrHülle}(F, \{ B, C \setminus C \}) = \{ B \}$$

$$\{D, E\} \rightarrow \{C, B\}$$

$$\{C, B\} \notin \text{AttrHülle}(F, \{D, E \setminus D\}) = \{E\}$$

$$\{C, B\} \notin \text{AttrHülle}(F, \{D, E \setminus E\}) = \{B, D\}$$

$$\text{FA} = \left\{ \right.$$

$$\{A, D\} \rightarrow \{E\},$$

$$\{B, C\} \rightarrow \{A, E\},$$

$$\{D\} \rightarrow \{B\},$$

$$\{D, E\} \rightarrow \{C, B\},$$

$$\{A\} \rightarrow \{F\},$$

$$\left. \right\}$$

- (ii) Führen Sie eine Rechtsreduktion des Ergebnisses der Linksreduktion ( $FD_L$ ) durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Rechtsreduktion an ( $FD_R$ ).

Lösungsvorschlag

### Rechtsreduktion

— Führe für jede (verbliebene) funktionale Abhängigkeit  $\alpha \rightarrow \beta$  die Rechtsreduktion durch, überprüfe also für alle  $B \in \beta$ , ob  $B \in \text{AttrHülle}(F - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B)), \alpha)$  gilt. In diesem Fall ist  $B$  auf der rechten Seite überflüssig und kann eliminiert werden,  $\alpha \rightarrow \beta$  wird durch  $\alpha \rightarrow (\beta - B)$  ersetzt.

### E

$$E \notin \text{AttrHülle}(F \setminus \{A, D\} \rightarrow \{E\}, \{A, D\}) = \{A, B, D, F\}$$

$$E \notin \text{AttrHülle}(F \setminus \{B, C\} \rightarrow \{A, E\} \cup \{B, C\} \rightarrow \{A\}, \{B, C\}) = \{A, B, C, F\}$$

### B

$$B \notin \text{AttrHülle}(F \setminus \{D\} \rightarrow \{B\}, \{D\}) = \{D\}$$

$$B \in \text{AttrHülle}(F \setminus \{D, E\} \rightarrow \{C, B\} \cup \{D, E\} \rightarrow \{C\}, \{D, E\}) = \{B, D, E\}$$

$$\text{FA} = \left\{ \right.$$

$$\{A, D\} \rightarrow \{E\},$$

$$\{B, C\} \rightarrow \{A, E\},$$

$$\{D\} \rightarrow \{B\},$$

$$\{D, E\} \rightarrow \{C\},$$

$$\{A\} \rightarrow \{F\},$$

$$\left. \right\}$$

- (iii) Bestimmen Sie eine kanonische Überdeckung  $FD$ . von  $FD$  auf Basis des Ergebnisses der Rechtsreduktion ( $FD_R$ ).

- **Löschen leerer Klauseln**

— Entferne die funktionalen Abhängigkeiten der Form  $\alpha \rightarrow \emptyset$ , die im 2. Schritt möglicherweise entstanden sind.

∅ Nichts zu tun

- **Vereinigung**

— Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form  $\alpha \rightarrow \beta_1, \dots, \alpha \rightarrow \beta_n$ , so dass  $\alpha \rightarrow \beta_1 \cup \dots \cup \beta_n$  verbleibt.

∅ Nichts zu tun

- (d) Zerlegen Sie  $R$  mit  $FD_C$  mithilfe des Synthesealgorithmus in 3NF. Geben Sie zudem alle funktionalen Abhängigkeiten der erzeugten Relationenschemata an.

- **Relationenschemata formen**

— Erzeuge für jede funktionale Abhängigkeit  $\alpha \rightarrow \beta \in F_C$  ein Relationenschema  $R_\alpha := \alpha \cup \beta$ .

$R_1(\underline{A}, \underline{D}, E)$

$R_2(\underline{B}, \underline{C}, A, E)$

$R_3(\underline{D}, B)$

$R_4(\underline{D}, E, C)$

$R_5(\underline{A}, F)$

- **Schlüssel hinzufügen**

— Falls eines der in Schritt 2. erzeugten Schemata  $R_\alpha$  einen Schlüsselkandidaten von  $\mathcal{R}$  bezüglich  $F_C$  enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten  $\mathcal{K} \subseteq \mathcal{R}$  aus und definiere folgendes zusätzliche Schema:  $\mathcal{R}_\mathcal{K} := \mathcal{K}$  und  $\mathcal{F}_\mathcal{K} := \emptyset$

∅ Nichts zu tun

- **Entfernung überflüssiger Teilschemata**

— Eliminiere diejenigen Schemata  $R_\alpha$ , die in einem anderen Relationenschema  $R_{\alpha'}$  enthalten sind, d. h.  $R_\alpha \subseteq R_{\alpha'}$ .

∅ Nichts zu tun

- (e) Prüfen Sie für alle Relationen der Zerlegung aus d), ob sie jeweils in BCNF sind.

$R_1$  und  $R_4$  sind in BCNF, weil ihre Determinanten Schlüsselkandidaten sind.

## 66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 1

Gegeben sei folgender Sachverhalt: Eine Grafik ist entweder ein Kreis, ein Quadrat oder ein Dreieck. Eine Grafik kann zudem auch eine Kombination aus diesen Elementen sein. Des Weiteren können Sie aus mehreren Grafiken auch neue Grafiken zusammenbauen. Sie denken sich: Ich möchte eine Menge von Grafiken genauso wie eine einzelne Grafik behandeln können.

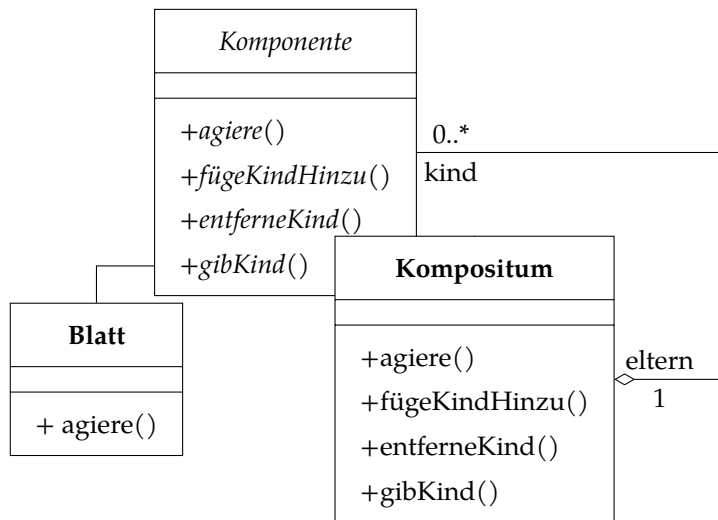
(a) Welches Entwurfsmuster sollten Sie zur Modellierung verwenden?

Lösungsvorschlag

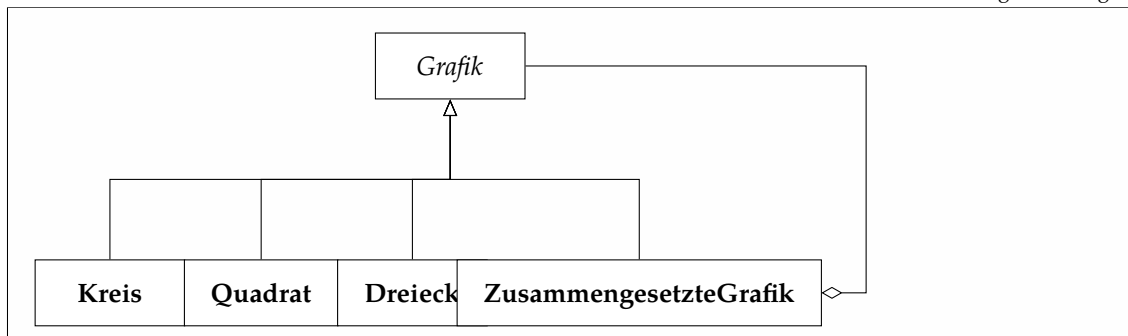
Kompositum

(b) Zeichnen Sie das entsprechende Klassendiagramm. Es reicht, nur die Klassennamen mit ihren Assoziationen und Vererbungsbeziehungen anzugeben; öhne Methoden und Attribute.

#### Exkurs: Kompositum



Lösungsvorschlag



## 66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2

Hintergrundinformationen für die Aufgaben 2-5: Modellierung und Implementierung eines Programms

Ein autonomer Roboter in einer Montagehalle bekommt einen Auftrag, eine Menge von Objekten aus dem Lager (Material, z. B. Schrauben oder Werkzeug, z. B. Bohrer) zu holen und anschließend an seinen Ausgangspunkt zu bringen. Der Roboter hat einen Namen, der ihn eindeutig identifiziert, kennt seine aktuelle Position (x- und y-Koordinate) und hat als weitere Eigenschaft den Auftrag, der aus einer Liste von Auftragspositionen besteht, die er holen soll. Der Roboter besitzt folgende Fähigkeiten (Methoden):

- (a) Er kann sich zu einer angegebenen Position bewegen.
- (b) Er hat eine Methode, um einen Auftrag abzuarbeiten, d.h. alle Auftragspositionen zu holen.

Alle Objekte im Lager haben jeweils einen Namen, einen Standort mit Angabe einer Position (s. oben) und speichern außerdem die jeweils noch vorhandene Stückzahl. Es gibt zwei Typen von Objekten: Werkzeuge mit einer Methode, mit der ein einzelnes Werkzeug ausgeliehen werden kann, und Materialien mit einer Methode, mit der sie verbraucht werden, wobei eine gewünschte Stückzahl angegeben wird. Diese vorgegebenen Methoden aktualisieren die Stückzahlen der Werkzeuge (Reduktion um 1) bzw. Materialien (Reduktion um verbrauchte Stückzahl), wobei hier vereinfachend angenommen wird, dass die Stückzahlen der Werkzeuge und Materialien immer ausreichend groß sind, um die geforderten Mengen bedienen zu können (d.h. Sie können diese beiden Methoden nutzen, ohne deren Implementierung angeben zu müssen).

Ein Auftrag (z. B. „Hole Bohrer Typ B1, 100 × Schrauben M6, 10 × Schrauben M10 und 2 × Blech B72“) besteht aus einer Menge von Auftragspositionen. Eine Auftragsposition besteht aus dem Typ des zu holenden Objekts (Werkzeug oder Material, soll als Enumeration modelliert werden), einem Verweis auf das zu holende Objekt und der zu holenden Stückzahl (Quantität; bei Werkzeugen wird diese ignoriert, da sie immer 1 ist). Der Roboter soll über die Auftragsposition außerdem die Position bestimmen, zu der er fahren muss, um das Objekt zu holen.

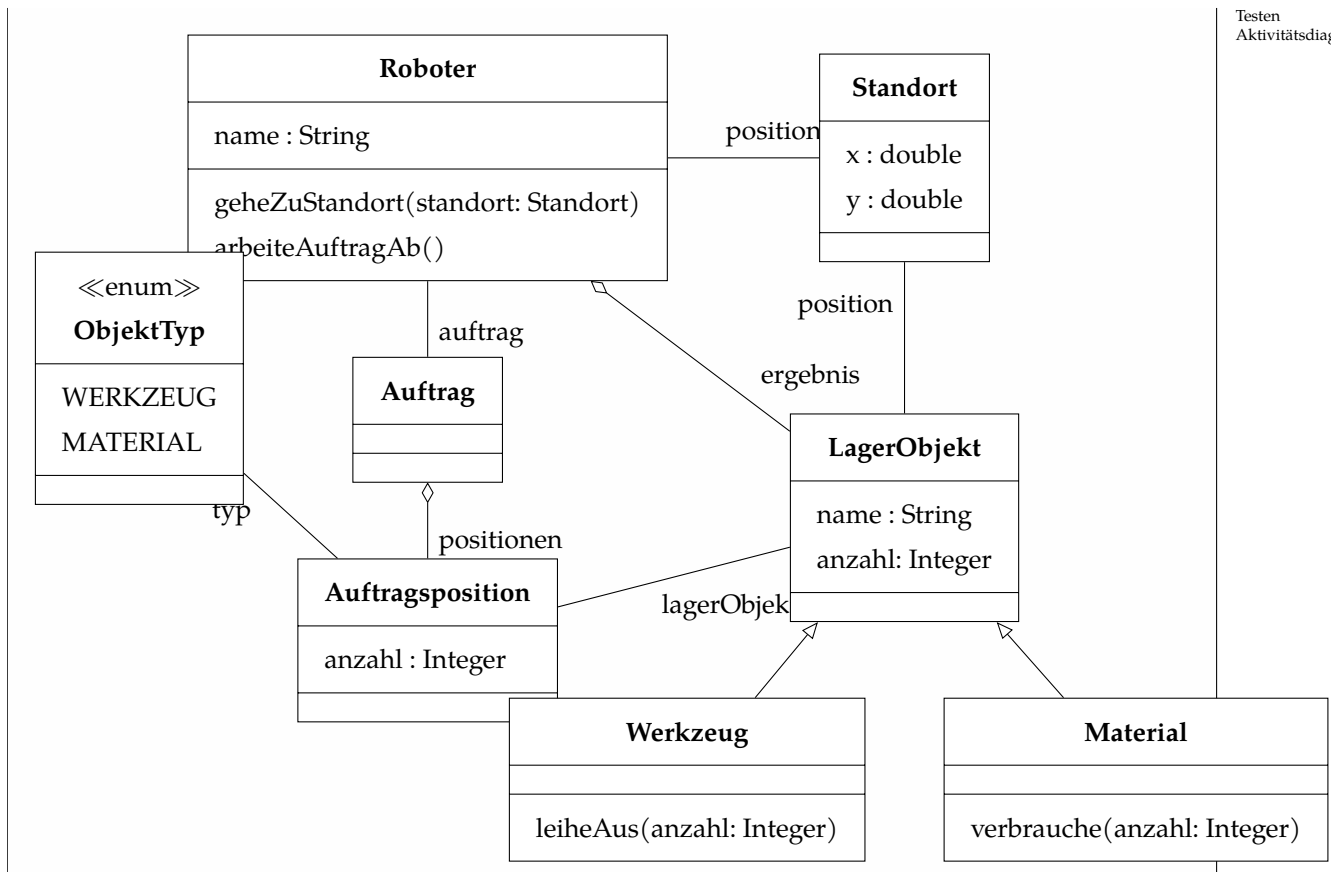
Der Roboter arbeitet die Auftragspositionen in der Reihenfolge ab, indem er sich von seinem aktuellen Standpunkt immer zur am nächsten liegenden Auftragsposition bewegt, um dort das nächste Objekt zu holen. Wir gehen der Einfachheit halber davon aus, dass die Montagehalle gut aufgeräumt ist und der Roboter sich quasi entlang der Luftlinie bewegen kann, d.h. die Entfernung zwischen zwei Positionen entspricht der euklidischen Distanz (Wurzel aus der Summe der Quadrate der Differenzen der x- und y-Koordinaten der Positionen). Der Roboter soll zur Kontrolle als weitere Eigenschaft „Ergebnis“ die Liste der eingesammelten Objekte zu einem Auftrag in der Reihenfolge speichern, in der er sie geholt hat, z.B. (M6 M10 B72 B1).

Geben Sie ein UML-Klassendiagramm zu der Aufgabenstellung an. Hinweis: Bei den Aufgaben 4 und 5 wird Konsistenz des Aktivitätsdiagramms bzw. des Codes mit dem Klassendiagramm verlangt.

Lösungsvorschlag

---





### 66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 3

- (a) Erläutern Sie kurz, was man unter der Methode der testgetriebenen Entwicklung versteht.

Lösungsvorschlag

Bei der testgetriebenen Entwicklung erstellt der/die ProgrammiererIn Softwaretests konsequent vor den zu testenden Komponenten.

- (b) Geben Sie für obige Aufgabenstellung (Abarbeitung der Aufträge durch den Roboter) einen Testfall für eine typische Situation an (d. h. das Einsammeln von 4 Objekten an unterschiedlichen Positionen). Spezifizieren Sie als Input alle für die Abarbeitung eines Auftrages relevanten Eingabe- und Klassen-Parameter sowie den vollständigen und korrekten Output.

### 66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 4

- (a) Geben Sie für alle Methoden, die zur Abarbeitung des Auftrages für den Roboter erforderlich sind, Pseudocode an. Nutzen Sie (im Klassendiagramm definierte) Hilfsmethoden, um den Code übersichtlicher zu gestalten. (Verwenden Sie statt einer komplizierten Methode mehrere einfachere Methoden, die sich aufrufen.) Achten Sie auf die Konsistenz zum Klassendiagramm (Inkonsistenzen führen zu Punktabzügen).

- (b) Überführen Sie den Pseudocode der Hauptmethode zur Abarbeitung eines Auftrags in ein syntaktisch korrektes UML-Aktivitätsdiagramm. Schleifen und Verzweigungen müssen durch strukturierte Knoten dargestellt werden (d. h. kein graphisches "goto").

## 66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 5

Geben Sie korrekten Code für die Abarbeitung eines Auftrages durch den Roboter in einer objektorientierten Programmiersprache Ihrer Wahl an. Sie sollen nur den Code für die Methoden angeben. Sie brauchen keinen Code für Klassendefinitionen angeben, sondern können sich auf das UML-Klassendiagramm aus Aufgabe 2 beziehen.

```
import java.util.ArrayList;

public class Auftrag {
 ArrayList<AuftragsPosition> positionen;

 public Auftrag() {
 this.positionen = new ArrayList<AuftragsPosition>();
 }

 public void fügePositionHinzu(LagerObjekt lagerObjekt, int anzahl) {
 positionen.add(new AuftragsPosition(lagerObjekt, anzahl));
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/fruehjahr/roboter/Auftrag.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/Auftrag.java)

```
public class AuftragsPosition {
 ObjektTyp typ;
 LagerObjekt lagerObjekt;
 int anzahl;

 public AuftragsPosition(LagerObjekt lagerObjekt, int anzahl) {
 if (lagerObjekt instanceof Werkzeug) {
 typ = ObjektTyp.WERKZEUG;
 } else {
 typ = ObjektTyp.MATERIAL;
 }
 this.lagerObjekt = lagerObjekt;
 this.anzahl = anzahl;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/fruehjahr/roboter/AuftragsPosition.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/AuftragsPosition.java)

```
public class LagerObjekt {
 String name;
 Standort position;
 int anzahl;

 public LagerObjekt(String name, int x, int y, int anzahl) {
 this.name = name;
 this.position = new Standort(x, y);
 }
}
```

```
 this.anzahl = anzahl;
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/fruehjahr/roboter/LagerObjekt.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/LagerObjekt.java)

```
public class Material extends LagerObjekt {
 public Material(String name, int x, int y, int anzahl) {
 super(name, x, y, anzahl);
 }

 void verbrauche(int anzahl) {
 }

}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/fruehjahr/roboter/Material.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/Material.java)

```
public enum ObjektTyp {
 WERKZEUG,
 MATERIAL
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/fruehjahr/roboter/ObjektTyp.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/ObjektTyp.java)

```
import java.util.ArrayList;

public class Roboter {
 String name;
 Standort position;
 ArrayList<LagerObjekt> ergebnis;

 void geheZuStandort(Standort standort) {
 }

 void arbeiteAuftragAb() {
 }

 public static void main(String[] args) {
 Werkzeug bohrerB1 = new Werkzeug("Bohrer Typ B1", 1, 0, 1);
 Material schraubenM6 = new Material("Schrauben M6", 1, 1, 100);
 Material schraubenM10 = new Material("Schrauben M10", 1, 2, 10);
 Material blechB72 = new Material("Blech B72", 1, 3, 2);

 Auftrag auftrag = new Auftrag();
 auftrag.fügePositionHinzu(bohrerB1, 1);
 auftrag.fügePositionHinzu(schraubenM6, 100);
 auftrag.fügePositionHinzu(schraubenM10, 10);
 auftrag.fügePositionHinzu(blechB72, 2);
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/fruehjahr/roboter/Roboter.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/Roboter.java)

```
public class Standort {
 double x;
 double y;

 public Standort(int x, int y) {
 this.x = x;
 this.y = y;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/fruehjahr/roboter/Standort.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/Standort.java)

```
public class Werkzeug extends LagerObjekt {

 public Werkzeug(String name, int x, int y, int anzahl) {
 super(name, x, y, anzahl);
 }

 public void leiheAus(int anzahl) {

 }

}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/fruehjahr/roboter/Werkzeug.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/Werkzeug.java)

## 66116 / 2019 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1

- (a) Das ACID-Prinzip fordert unter anderem die Atomarität und die Isolation einer Transaktion. Beschreiben Sie kurz die Bedeutung dieser Begriffe.

Lösungsvorschlag

**Atomicity** Eine Transaktion ist atomar, d.h. von den vorgesehenen Änderungsoperationen auf die Datenbank haben entweder alle oder keine eine Wirkung auf die Datenbank.

**Isolation** Eine Transaktion bemerkt das Vorhandensein anderer (parallel ablaufender) Transaktionen nicht und beeinflusst auch andere Transaktionen nicht.

- (b) Was versteht man unter physischer Datenunabhängigkeit?

Lösungsvorschlag

Änderungen an der physischen Speicher- oder der Zugriffsstruktur (beispielsweise durch das Anlegen einer Indexstruktur) haben keine Auswirkungen auf die logische Struktur der Datenbasis, also auf das Datenbankschema.

- (c) In welche drei Ebenen unterteilt die 3-Ebenen-Architektur Datenbanksysteme?

**Externe Ebene / (Benutzer)sichten (Views)** Die externe Ebene stellt den Benutzern und Anwendungen individuelle Benutzersichten bereit, wie beispielsweise Formulare, Masken-Layouts, Schnittstellen.

**Konzeptionelle / logische Ebene** Die konzeptionelle Ebene beschreibt, welche Daten in der Datenbank gespeichert sind, sowie deren Beziehungen. Ziel des Datenbankdesigns ist eine vollständige und redundanzfreie Darstellung aller zu speichernden Informationen. Hier findet die Normalisierung des relationalen Datenbankschemas statt.

**Interne / physische Ebene** Die interne Ebene stellt die physische Sicht der Datenbank im Computer dar. Sie beschreibt, wie und wo die Daten in der Datenbank gespeichert werden. Oberstes Designziel ist ein effizienter Zugriff auf die gespeicherten Informationen, der meist durch bewusst in Kauf genommene Redundanz (z. B. Index) erreicht wird.

- (d) Definieren Sie, was ein Schlüssel ist.

Ein Schlüssel ist ein Attribut oder eine Attributkombination, von der alle Attribute einer Relation funktional abhängen.

- (e) Gegeben ist eine Relation  $R(A, B, C)$ , deren einziger Schlüsselkandidat  $A$  ist. Nennen Sie zwei Superschlüssel.

$\{A, B\}$  oder  $\{A, C\}$  oder  $\{A, B, C\}$

- (f) Gegeben seien 2 Relationen  $R(A, B, C)$  und  $S(C, D, E)$ . Die Relation  $R$  enthalte 9 Tupel und die Relation  $S$  enthalte 7 Tupel. Sei  $p$  ein beliebiges Selektionsprädikat. Gegeben seien außerdem folgende Anfragen:

$Q_1$ :

```
SELECT *
FROM R NATURAL JOIN S
WHERE p;
```

$Q_2$ :

```
SELECT DISTINCT A
FROM R LEFT OUTER JOIN S ON R.C=S.C;
```

- (i) Wieviele Ergebnistupel liefert die Anfrage  $Q_1$  mindestens?

0

- (ii) Wieviele Ergebnistupel liefert die Anfrage  $Q_2$  höchstens?

Lösungsvorschlag

7

(iii) Wieviele Ergebnistupel liefert die Anfrage  $Q_2$  mindestens?

Lösungsvorschlag

9

(iv) Wieviele Ergebnistupel liefert die Anfrage  $Q_2$  höchstens?

Lösungsvorschlag

9

(g) Erläutern Sie, was es bedeutet, wenn ein Attribut prim ist.

Lösungsvorschlag

Sei  $R$  ein Relationenschema. Ein Attribut  $A \in R$  heißt *prim*, falls  $A$  Teil eines Schlüsselkandidaten von  $R$  ist. Andernfalls heißt  $A$  *nichtprim*.

(h) Nennen Sie die drei Armstrong-Axiome, die dazu dienen aus einer Menge von funktionalen Abhängigkeiten, die auf einer Relation gelten, weitere funktionale Abhängigkeiten abzuleiten.

Lösungsvorschlag

**Reflexivität:** Eine Menge von Attributen bestimmt eindeutig die Werte einer Teilmenge dieser Attribute (triviale Abhängigkeit), das heißt,  $\beta \subseteq \alpha \Rightarrow \alpha \rightarrow \beta$ .

**Verstärkung:** Gilt  $\alpha \rightarrow \beta$ , so gilt auch  $\alpha\gamma \rightarrow \beta\gamma$  für jede Menge von Attributen  $\gamma$  der Relation.

**Transitivität:** Gilt  $\alpha \rightarrow \beta$  und  $\beta \rightarrow \gamma$ , so gilt auch  $\alpha \rightarrow \gamma$ .

(i) Nennen Sie die höchste Normalform, der die Relation  $R(A, B, C, D)$  mit den Abhängigkeiten  $\{A\} \rightarrow \{C\}$  und  $\{A, B\} \rightarrow \{D\}$  entspricht. Begründen Sie Ihre Antwort.

Lösungsvorschlag

1NF.  $\{A\} \rightarrow \{C\}$  verletzt die 2NF, da das Nichtprimärattribut  $C$  funktional von einer echten Teilmenge ( $A$ ) des Schlüsselkandidaten ( $\{A, B\} \rightarrow \{D\}$ ) abhängt.

(j) Gegeben seien die Transaktionen  $T_1, T_2$  und  $T_3$ . Wie viele mögliche verschiedene serialisierbare Schedules gibt es mindestens? (Geben Sie Ihren Rechenweg an.)

Lösungsvorschlag

Vergleiche Kemper Seite 341 Kapitel „Theorie der Serialisierbarkeit“.

**66116 / 2019 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 3**

Gegeben seien folgende Relationen:

**X**

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 3 | 4 |
| 4 | 3 | 3 |
| 1 | 2 | 2 |
| 2 | 3 | 2 |
| 3 | 3 | 2 |
| 3 | 1 | 2 |
| 2 | 2 | 1 |
| 1 | 1 | 1 |

**Y**

| B | C | D |
|---|---|---|
| 2 | 3 | 1 |
| 1 | 1 | 3 |
| 2 | 1 | 3 |
| 3 | 2 | 3 |
| 2 | 2 | 3 |
| 1 | 3 | 2 |

Geben Sie die Ergebnisrelationen folgender Ausdrücke der relationalen Algebra als Tabellen an; machen Sie Ihren Rechenweg kenntlich.

(a)  $\sigma_{A=2}(X) \bowtie Y$

Lösungsvorschlag

| A | B | C | D | E |
|---|---|---|---|---|
| 2 | 3 | 2 | 3 | 1 |
| 2 | 3 | 2 | 1 | 3 |
| 2 | 3 | 2 | 2 | 3 |
| 2 | 2 | 1 | 1 | 3 |
| 2 | 2 | 1 | 3 | 2 |

$$(b) (\pi_{B,C}(X) - \pi_{B,C}(Y)) \bowtie X$$

SQL

Lösungsvorschlag

(c)

|   |
|---|
| A |
| 1 |
| 2 |
| 3 |

## 66116 / 2019 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 5

Wir betrachten erneut das gegebene Schema aus Aufgabe 4. Primärschlüssel sind unterstrichen, Fremdschlüssel sind überstrichen und der Text in den darauf folgenden eckigen Klammern benennt die Relation, auf die verwiesen wird.

Der Notenschnitt, der Preis und die Bewertung werden als Kommazahl dargestellt, wobei die Bewertung die Anzahl der Sterne angibt, also maximal 5 und mindestens 0. Die Modellnummer kann sowohl aus Zahlen und Buchstaben bestehen, ist jedoch nie länger als 50 Zeichen. ID, RAM, Bildschirmdiagonale und Datum sind ganze Zahlen. Die restlichen Attribute sind Strings der Länge 15.

Lehrer (Name, Fach1, Fach2, Fach3)

Schüler (Vorname, Nachname, Notenschnitt)

Smartphone (ID, Modellnr, RAM, Bildschirmdiagonale)

App (Name, Bewertung, Preis)

Eingesammelt (Vorname [Schüler], Nachname [Schüler], Name [Lehrer], ID [Smartphone], Datum)

Installiert (ID [Smartphone], Name [App])

- (a) Geben Sie die Anweisung in SQL-DDL an, die notwendig ist, um die Relation 'App' zu erzeugen.

Lösungsvorschlag

```
CREATE TABLE App(
 Name VARCHAR(50) PRIMARY KEY,
 Bewertung VARCHAR(255),
 Preis INTEGER);
```

- (b) Geben Sie die Anweisung in SQL-DDL an, die notwendig ist, um die Relation 'Installiert' zu erzeugen.

Lösungsvorschlag

```
CREATE TABLE Installiert(
 ID INTEGER REFERENCES Smartphone(ID),
 Name VARCHAR(50) App(Name),
 PRIMARY KEY(ID, Name));
```



- (c) Formulieren Sie die folgende Anfrage in SQL: Gesucht sind die Namen der Apps zusammen mit ihrer Bewertung, die auf den Smartphones installiert sind, die Lehrer Keating eingesammelt hat. Sortieren Sie das Ergebnis nach Bewertung absteigend. Hinweis: Achten Sie auf gleichnamige Attribute.

Lösungsvorschlag

```
SELECT DISTINCT a.Name, a.Bewertung
FROM Eingesammelt e, Installiert i, App a
WHERE e.ID=i.ID AND i.Name = a.Name AND e.Name = "Keating"
ORDER BY a.Bewertung DESC;
```

- (d) Formulieren Sie die folgende Anfrage in SQL:

Gesucht ist der durchschnittliche Notenschnitt der Schüler, denen ein iPhone 65 abgenommen wurde. Ein iPhone 6s kann A1633 als Modellnummer haben oder A1688.

Lösungsvorschlag

```
SELECT DISTINCT AVG(s.Notenschnitt)
FROM Schüler s, Eingesammelt e, Smartphone sm
WHERE s.Vorname = e.Vorname AND s.Nachname = e.Nachname
AND e.ID=sm.ID AND (sm.Modellnr = 'A1633' OR sm.Modellnr = 'A1688');
```

- (e) Formulieren Sie die folgende Anfrage in SQL:

Gesucht ist die Modellnummer der Smartphones, die durchschnittlich die meisten Apps installiert haben.

Tipp: Die Verwendung von Views kann die Aufgabe vereinfachen.

Lösungsvorschlag

```
CREATE VIEW NumberApps AS
SELECT s.ID, s.Modellnr, COUNT(i.Name) AS number
FROM Smartphone s, Installiert i
WHERE s.ID = i.ID
GROUP BY s.ID
SELECT ModellNr, AVG(number)
FROM NumberApps
GROUP BY ModellNr
```

## 66116 / 2019 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 6

Gegeben sei das Relationsschema  $R(A, B, C, D, E, F)$ , sowie die Menge der zugehörigen funktionalen Abhängigkeiten:

$$FA = \{$$

$$\begin{aligned}
 &\{ B \} \rightarrow \{ F \}, \\
 &\{ C, D \} \rightarrow \{ E \}, \\
 &\{ C \} \rightarrow \{ A \}, \\
 &\{ C, D \} \rightarrow \{ A \}, \\
 &\{ D \} \rightarrow \{ F \}, \\
 &\{ D \} \rightarrow \{ B \},
 \end{aligned}
 \}$$

- (a) Bestimmen Sie den Schlüsselkandidaten der Relation  $R$  und begründen Sie, warum es keine weiteren Schlüsselkandidaten gibt.

Lösungsvorschlag

Der Schlüsselkandidat ist  $\{ C, D \}$ , da  $\{ C, D \}$  auf keiner rechten Seiten der Funktionalen Abhängigkeiten vorkommt. Außerdem ist  $\{ C, D \}$  ein Super-schlüssel da gilt:  $\text{AttrHülle}(F, \{ C, E \}) = \{ A, B, C, D, E, G \} = R$

- (b) Überführen Sie das Relationsschema  $R$  mit Hilfe des Synthesealgorithmus in die dritte Normalform. Führen Sie hierfür jeden der vier Schritte durch und kennzeichnen Sie Stellen, bei denen nichts zu tun ist. Benennen Sie alle Schritte und begründen Sie eventuelle Reduktionen.

Lösungsvorschlag

(i) **Kanonische Überdeckung**

— Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden. —

i. **Linksreduktion**

— Führe für jede funktionale Anhängigkeit  $\alpha \rightarrow \beta \in F$  die Linksreduktion durch, überprüfe also für alle  $A \in \alpha$ , ob  $A$  überflüssig ist, d. h. ob  $\beta \subseteq \text{AttrHülle}(F, \alpha - A)$ . —

$$\begin{aligned}
 FA = \{ & \\
 &\{ B \} \rightarrow \{ F \}, \\
 &\{ C, D \} \rightarrow \{ E \}, \\
 &\{ C \} \rightarrow \{ A \}, \\
 &\{ C \} \rightarrow \{ A \}, \\
 &\{ D \} \rightarrow \{ F \}, \\
 &\{ D \} \rightarrow \{ B \}, \\
 &\}
 \end{aligned}$$

ii. **Rechtsreduktion**

— Führe für jede (verbliebene) funktionale Abhängigkeit  $\alpha \rightarrow \beta$  die Rechtsreduktion durch, überprüfe also für alle  $B \in \beta$ , ob  $B \in \text{AttrHülle}(F - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B)), \alpha)$  gilt. In diesem Fall ist  $B$  auf der rechten Seite überflüssig und kann eliminiert werden, d.h.  $\alpha \rightarrow \beta$  wird durch  $\alpha \rightarrow (\beta - B)$  ersetzt. —

$$FA = \{$$

$$\left. \begin{array}{l} \{B\} \rightarrow \{F\}, \\ \{C, D\} \rightarrow \{E\}, \\ \{C\} \rightarrow \{\emptyset\}, \\ \{C\} \rightarrow \{A\}, \\ \{D\} \rightarrow \{\emptyset\}, \\ \{D\} \rightarrow \{B\}, \end{array} \right\}$$

**iii. Löschen leerer Klauseln**

— Entferne die funktionalen Abhängigkeiten der Form  $\alpha \rightarrow \emptyset$ , die im 2. Schritt möglicherweise entstanden sind. \_\_\_\_\_

$$\text{FA} = \left\{ \begin{array}{l} \{B\} \rightarrow \{F\}, \\ \{C, D\} \rightarrow \{E\}, \\ \{C\} \rightarrow \{A\}, \\ \{D\} \rightarrow \{B\}, \end{array} \right\}$$

**iv. Vereinigung**

— Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form  $\alpha \rightarrow \beta_1, \dots, \alpha \rightarrow \beta_n$ , so dass  $\alpha \rightarrow \beta_1 \cup \dots \cup \beta_n$  verbleibt. \_\_\_\_\_

∅ Nichts zu tun

**(ii) Relationsschemata formen**

— Erzeuge für jede funktionale Abhängigkeit  $\alpha \rightarrow \beta \in F_c$  ein Relationenschema  $\mathcal{R}_\alpha := \alpha \cup \beta$ .

- $R_1(B, F)$
- $R_2(C, D, E)$
- $R_3(C, A)$
- $R_4(D, B)$

**(iii) Schlüssel hinzufügen**

— Falls eines der in Schritt 2. erzeugten Schemata  $\mathcal{R}_\alpha$  einen Schlüsselkandidaten von  $\mathcal{R}$  bezüglich  $F_c$  enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten  $\mathcal{K} \subseteq \mathcal{R}$  aus und definiere folgendes zusätzliche Schema:  $\mathcal{R}_\mathcal{K} := \mathcal{K}$  und  $\mathcal{F}_\mathcal{K} := \emptyset$  \_\_\_\_\_

∅ Nichts zu tun

**(iv) Entfernung überflüssiger Teilschemata**

— Eliminiere diejenigen Schemata  $\mathcal{R}_\alpha$ , die in einem anderen Relationenschema  $\mathcal{R}_{\alpha'}$  enthalten sind, d. h.  $\mathcal{R}_\alpha \subseteq \mathcal{R}_{\alpha'}$ . \_\_\_\_\_

∅ Nichts zu tun

**66116 / 2019 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1**

Bewerten Sie die folgenden Aussagen und nehmen Sie jeweils Stellung.

(a) Tests zuerst

In test-getriebener Softwareentwicklung wird der Test vor der zu testenden Funktion programmiert.

Lösungsvorschlag

Bei der testgetriebenen Entwicklung erstellt der/die ProgrammiererIn Softwaretests konsequent vor den zu testenden Komponenten.

(b) Komponententests

Komponententests sind immer White-Box-Tests und nie Black-Box-Tests.

Lösungsvorschlag

Falsch: Komponententests (anderes Wort für Unit- oder Modul-Tests) können beides sein. <sup>a</sup>

<sup>a</sup><https://qastack.com.de/software/362746/what-is-black-box-unit-testing>

(c) Akzeptanztests

Akzeptanz - resp. Funktionstests sind immer Black-Box-Tests und nie White-Box-Tests.

Lösungsvorschlag

In systems engineering, it may involve black-box testing performed on a system

Acceptance testing in extreme programming: Acceptance tests are black-box system tests.

<sup>a</sup>

<sup>a</sup>[https://en.wikipedia.org/wiki/Acceptance\\_testing](https://en.wikipedia.org/wiki/Acceptance_testing)

(d) Fehler

Ein fehlgeschlagener Test und ein Testausführungsfehler bezeichnen denselben Sachverhalt.

Lösungsvorschlag

Falsch:

Fehler (Error)

Der Software-Test konnte durchgeführt werden, das ermittelte Ist-Ergebnis und das vorgegebene Soll-Ergebnis weichen jedoch voneinander ab. Ein derartiger Fehler liegt z. B. dann vor, wenn ein Funktionsaufruf einen abweichenden Wert berechnet.

Fehlschlag (Failure)

Während der Testdurchführung wurde ein Systemversagen festgestellt. Ein Fehlschlag liegt z. B. dann vor, wenn das zu testende System einen Programmabsturz verursacht und hierdurch erst gar kein Ist-Ergebnis ermittelt werden konnte.

Das Misslingen kann als Ursache einen Fehler (Error) oder ein falsches Ergebnis (Failure) haben, die beide per Exception signalisiert werden. Der Unterschied zwischen den beiden Begriffen liegt darin, dass Failures erwartet werden, während Errors eher unerwartet auftreten.<sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/JUnit>

(e) Test Suiten

Tests können hierarchisch strukturiert werden, so dass mit einem Befehl das gesamte zu testende System getestet werden kann.

Lösungsvorschlag

Richtig:

test suite (englisch „Testsammlung“, aus französisch suite Folge, Verkettung) bezeichnet eine organisierte Sammlung von Werkzeugen zum Testen technischer Apparate und Vorgänge.<sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/Testsuite>

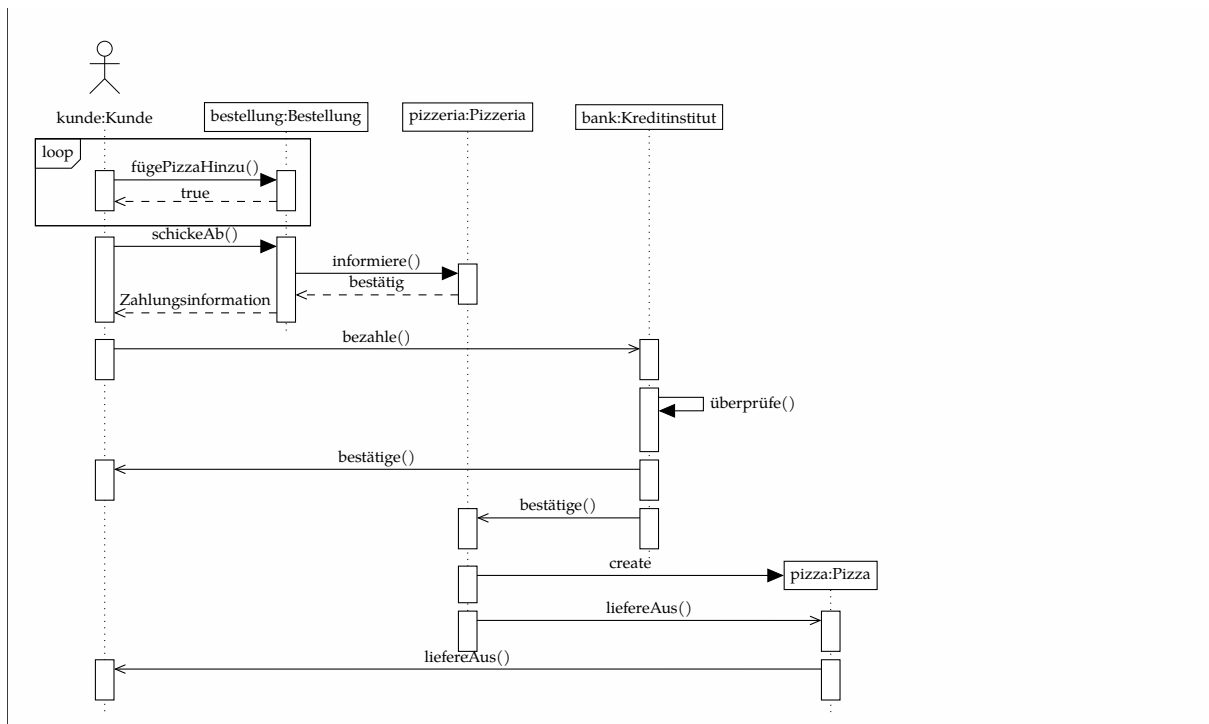
## 66116 / 2019 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Eine Pizzeria möchte mit dem Webdienst PizzaNow anbieten, Pizzen online bestellen zu können. Es gilt die folgende Beschreibung:

*„Ein Kunde kann mehrere Pizzen in PizzaNow zu einer Bestellung zusammenstellen. Nachdem er die Bestellung abgeschickt hat, wird die Pizzeria über diese Bestellung informiert. Dort wird die Bestellung bestätigt und der Kunde erhält eine Bestätigung mit Zahlungsinformationen. Diese müssen vom jeweiligen Kreditinstitut überprüft und bestätigt werden. Danach erhält der Kunde eine Bestätigung über den erfolgreichen Bezahlvorgang und die Pizzeria beginnt nach der Bestätigung der Bezahlung die Herstellung und im Anschluss die Auslieferung der Bestellung.“*

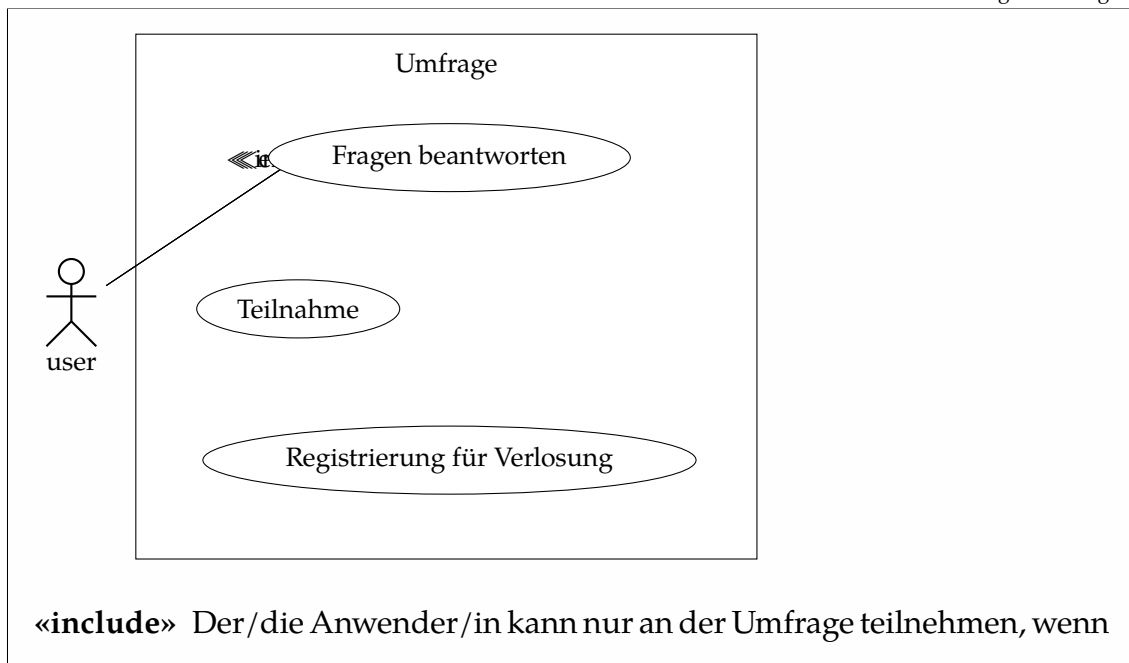
Modellieren Sie den dargestellten Prozess mit Hilfe eines UML-Sequenzdiagramms.

Lösungsvorschlag



## 66116 / 2019 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 3

- (a) Nehmen Sie an, dass in einem System zur Durchführung von Umfragen die drei Anwendungsfälle „Teilnahme“, „Fragen beantworten“ und „Registrierung für Verlosung“ gegeben seien. Alle Anwendungsfälle sind einem Teilnehmer (an einer Umfrage) zugänglich. Stellen Sie die drei Anwendungsfälle in einem UML-Anwendungsfalldiagramm („use case diagram“) dar. Verwenden Sie dabei sowohl die «extend» - als auch die «include»-Beziehung und beschreiben Sie die Bedeutung der Beziehungen für die Existenz des „Teilnahme“-Anwendungsfalls.



er/sie die Fragen beantwortet hat.

«**extend**» Der/die Anwender/in kann zusätzlich zur Teilnahme an der Umfrage sich für die Verlosung registrieren.

- (b) Welche Informationen sollte eine typische Spezifikation eines Anwendungsfalls enthalten?

Lösungsvorschlag

Ein Use-Case-Diagramm zeigt das externe Verhalten eines Systems aus der Sicht der Nutzer, indem es die Nutzer, die Use-Cases und deren Beziehungen zueinander darstellt.

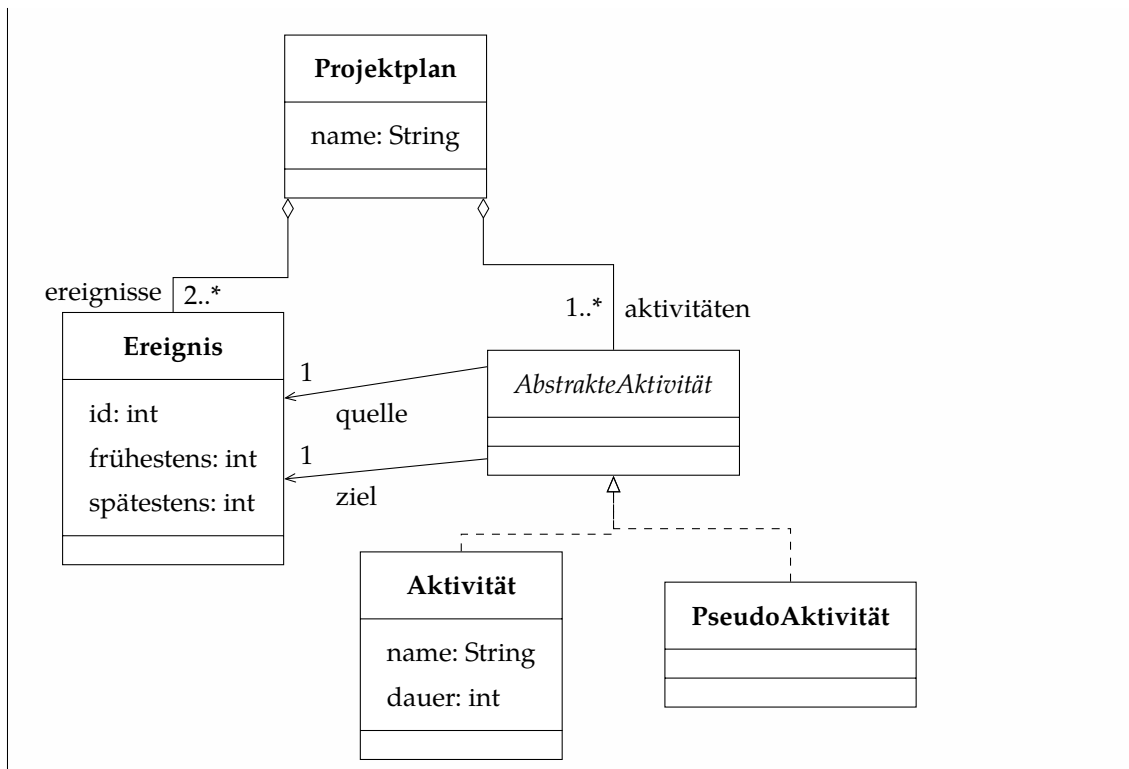
## 66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 1

Ein CPM-Netzwerk („Critical Path Method“) ist ein benannter Projektplan, der aus Ereignissen und Aktivitäten besteht. Ein Ereignis wird durch eine ganze Zahl  $> 0$  identifiziert. Jede Aktivität führt von einem Quellereignis zu einem Zielereignis. Eine reale Aktivität hat einen Namen und eine Dauer (eine ganze Zahl  $> 0$ ). Eine Pseudoaktivität ist anonym. Ereignisse und Pseudoaktivitäten verbrauchen keine Zeit. Zu jedem Ereignis gibt es einen frühesten und einen spätesten Zeitpunkt (eine ganze Zahl  $> 0$ ), deren Berechnung nicht Gegenstand der Aufgabe ist.

- (a) Erstellen Sie ein UML-Klassendiagramm zur Modellierung von CPM-Netzwerken. Geben Sie für Attribute jeweils den Namen und den Typ an. Geben Sie für Assoziationen den Namen und für jedes Ende den Rollennamen und die Multiplizität an. Nutzen Sie ggf. abstrakte Klassen, Vererbung, Komposition oder Aggregation. Verzichten Sie auf Operationen und Sichtbarkeiten.

Lösungsvorschlag





- (b) Erstellen Sie für das Klassendiagramm aus a) und das Beispiel aus der Aufgabenstellung ein Objektdiagramm. Geben Sie Rollennamen nur an, wenn es notwendig ist, um die Enden eines Links (Instanz einer Assoziation) zu unterscheiden.

## 66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Eine Dienstreise an einer Universität wird folgendermaßen abgewickelt:

Ein Mitarbeiter erstellt einen Dienstreiseantrag und legt ihn seinem Vorgesetzten zur Unterschrift vor. Mit seiner Unterschrift befürwortet der Vorgesetzte die Dienstreise. Verweigert er die Unterschrift, wird der Vorgang abgebrochen. Nach der Befürwortung wird der Antrag an die Reisekostenstelle weitergeleitet, die über die Annahme des Antrags entscheidet. Im Falle einer Ablehnung wird der Vorgang abgebrochen; sonst genehmigt die Reisekostenstelle den Antrag. Der Mitarbeiter kann nun einen Abschlag beantragen. Stimmt der Vorgesetzte zu, so entscheidet die Reisekostenstelle über den Abschlag und weist ggf. die Kasse der Universität an, den Abschlag auszuzahlen. Nach Ende der Dienstreise erstellt der Mitarbeiter einen Antrag auf Erstattung der Reisekosten an die Reisekostenstelle. Die Reisekostenstelle setzt den Erstattungsbetrag fest und weist die Kasse an, den Betrag auszuzahlen.

- (a) Erstellen Sie ein Anwendungsfalldiagramm (Use-Case-Diagramm) für Dienstreisen.
- (b) Erstellen Sie ein Aktivitätsdiagramm, das den oben beschriebenen Ablauf modelliert. Ordnen Sie den Aktionen die Akteure gemäß a) zu. Beschränken Sie sich auf den Kontrollfluss (keine Objektflüsse).

**66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3**

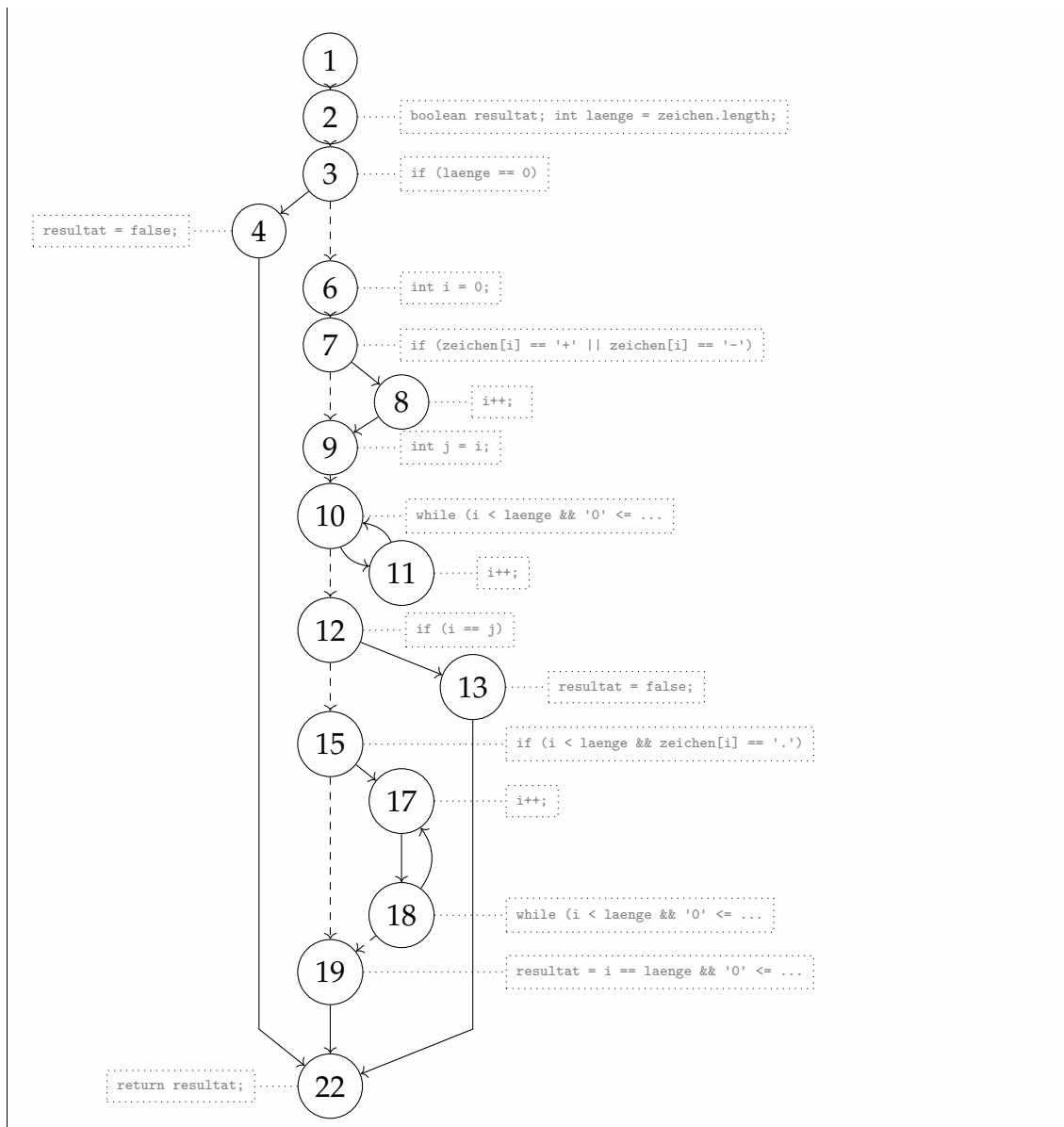
Eine Dezimalzahl hat ein optionales Vorzeichen, dem eine nichtleere Sequenz von Dezimalziffern folgt. Der anschließende gebrochene Anteil ist optional und besteht aus einem Dezimalpunkt, gefolgt von einer nichtleeren Sequenz von Dezimalziffern.

Die folgende Java-Methode erkennt, ob eine Zeichenfolge eine Dezimalzahl ist:

```
public static boolean istDezimalzahl(char[] zeichen) { // 1
 boolean resultat; int laenge = zeichen.length; // 2
 if (laenge == 0) // 3
 resultat = false; // 4
 else { // 5
 int i = 0; // 6
 if (zeichen[i] == '+' || zeichen[i] == '-') // 7
 i++; // 8
 int j = i; // 9
 while (i < laenge && '0' <= zeichen[i] && zeichen[i] <= '9') // 10
 i++; // 11
 if (i == j) // 12
 resultat = false; // 13
 else { // 14
 if (i < laenge && zeichen[i] == '.') // 15
 do // 16
 i++; // 17
 while (i < laenge && '0' <= zeichen[i] && zeichen[i] <= '9'); // 18
 resultat = i == laenge && '0' <= zeichen[i - 1] && zeichen[i - 1] <= '9';
 → // 19
 } // 20
 } // 21
 return resultat; // 22
 }
}
```

- (a) Konstruieren Sie zu dieser Methode einen Kontrollflußgraphen. Markieren Sie dessen Knoten mit Zeilennummern des Quelltexts.

Lösungsvorschlag



- (b) Geben Sie eine minimale Testmenge an, die das Kriterium der Knotenüberdeckung erfüllt. Geben Sie für jeden Testfall den durchlaufenen Pfad in der Notation  $1 \rightarrow 2 \rightarrow \dots$  an.

Lösungsvorschlag

- „“:  
 $1 - 2 - 3 - 4 - 22$
- „1“:  
 $1 - 2 - 3 - 6 - 7 - 9 - 10 - 11 - 10 - 12 - 15 - 19 - 22$
- „+1.0“:  
 $1 - 2 - 3 - 6 - 7 - 8 - 9 - 10 - 11 - 10 - 12 - 15 - 17 - 18 - 19 - 22$
- „X“:  
 $1 - 2 - 3 - 6 - 7 - 9 - 10 - 12 - 13 - 22$

- (c) Verfahren Sie wie in b) für das Kriterium der Kantenüberdeckung.

Lösungsvorschlag

text

- (d) Wie stehen die Kriterien der Knoten- und Kantenüberdeckung zueinander in Beziehung? Begründen Sie Ihre Antwort.

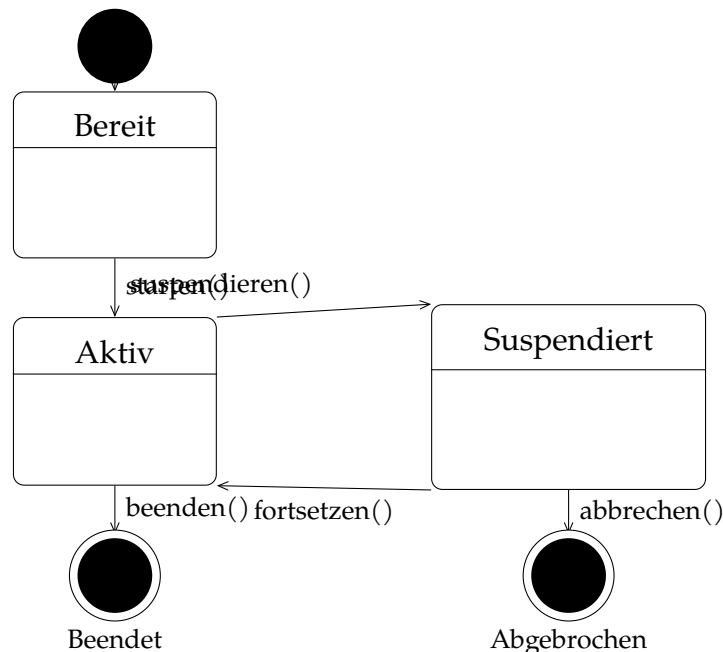
Hinweis: Eine Testmenge ist minimal, wenn es keine andere Testmenge mit einer kleineren Zahl von Testfällen gibt. Die Minimalität braucht nicht bewiesen zu werden.

Lösungsvorschlag

Die Kantenüberdeckung fordert, dass jede *Kante* des Kontrollflussgraphen von mindestens einem Testfall durchlaufen werden muss. Um das Kriterium zu erfüllen, müssen die Testfälle so gewählt werden, dass jede Verzweigungsbedingung mindestens *einmal wahr* und mindestens *einmal falsch* wird. Da hierdurch alle Knoten ebenfalls mindestens einmal besucht werden müssen, ist die *Anweisungsüberdeckung* in der Zweigüberdeckung *vollständig enthalten*.

## 66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4

Zu den Aufgaben eines Betriebssystems zählt die Verwaltung von Prozessen. Jeder Prozess durchläuft verschiedene Zustände; Transitionen werden durch Operationsaufrufe ausgelöst. Folgendes Zustandsdiagramm beschreibt die Verwaltung von Prozessen:

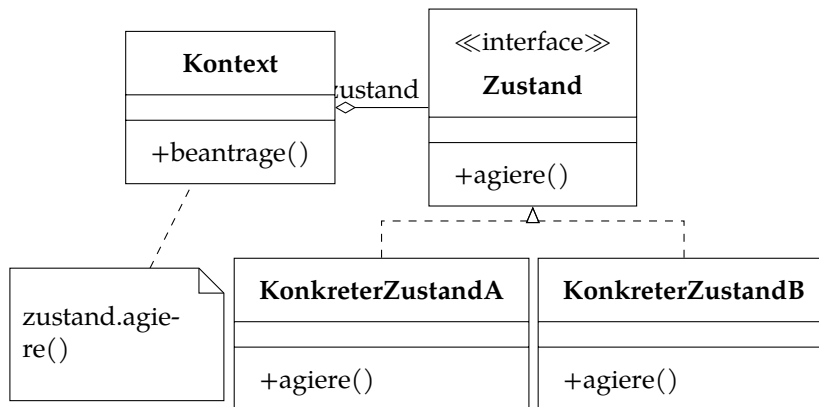


Implementieren Sie dieses Zustandsdiagramm in einer Programmiersprache Ihrer Wahl mit Hilfe des Zustandsmusters; geben Sie die gewählte Sprache an. Die Methoden für

die Transitionen sollen dabei die Funktionalität der Prozessverwaltung simulieren, indem der Methodenaufruf auf der Standardausgabe protokolliert wird. Falls Transitionen im aktuellen Zustand undefiniert sind, soll eine Fehlermeldung ausgegeben werden.

### Exkurs: Zustand-(State)-Entwurfsmuster

#### UML-Klassendiagramm



#### Teilnehmer

**Kontext (Context)** definiert die clientseitige Schnittstelle und verwaltet die separaten Zustandsklassen.

**State (Zustand)** definiert eine einheitliche Schnittstelle aller Zustandsobjekte und implementiert gegebenenfalls ein Standardverhalten.

**KonkreterZustand (ConcreteState)** implementiert das Verhalten, das mit dem Zustand des Kontextobjektes verbunden ist.

Lösungsvorschlag

Implementierung in der Programmiersprache „Java“:

| Methoden                | Zustände    | Klassennamen       |
|-------------------------|-------------|--------------------|
|                         | Bereit      | ZustandBereit      |
| starten(), fortsetzen() | Aktiv       | ZustandAktiv       |
| suspendieren()          | Suspendiert | ZustandSuspendiert |
| beenden()               | Beendet     | ZustandBeendet     |
| abbrechen()             | Abgebrochen | ZustandAbgebrochen |

```

/**
 * Entspricht der „Kontext“-Klasse in der Terminologie der „Gang of
 * Four“.
 */
public class Prozess {

```

```
private ProzessZustand aktuellerZustand;

public Prozess() {
 aktuellerZustand = new ZustandBereit(this);
}

public void setzeZustand(ProzessZustand zustand) {
 aktuellerZustand = zustand;
}

public void starten() {
 aktuellerZustand.starten();
}

public void suspendieren() {
 aktuellerZustand.suspendieren();
}

public void fortsetzen() {
 aktuellerZustand.fortsetzen();
}

public void beenden() {
 aktuellerZustand.beenden();
}

public void abbrechen() {
 aktuellerZustand.abbrechen();
}

public static void main(String[] args) {
 Prozess prozess = new Prozess();
 prozess.starten();
 prozess.suspendieren();
 prozess.fortsetzen();
 prozess.beenden();
 prozess.starten();

 // Ausgabe:
 // Der Prozess ist im Zustand „bereit“
 // Der Prozess wird gestartet.
 // Der Prozess ist im Zustand „aktiv“
 // Der Prozess wird suspendiert.
 // Der Prozess ist im Zustand „suspendiert“
 // Der Prozess wird fortgesetzt.
 // Der Prozess ist im Zustand „aktiv“
 // Der Prozess wird beendet.
 // Der Prozess ist im Zustand „beendet“
 // Im Zustand „beendet“ kann die Transition „starten“ nicht ausgeführt werden!
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/prozess\\_verwaltung/Prozess.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/Prozess.java)

```
/**
 * Entspricht der „Zustand“-Klasse in der Terminologie der "Gang of
 * Four".
 */
abstract class ProzessZustand {

 Prozess prozess;

 String zustand;

 public ProzessZustand(String zustand, Prozess prozess) {
 this.zustand = zustand;
 this.prozess = prozess;
 System.out.println(String.format("Der Prozess ist im Zustand „%s“, zustand));
 }

 private void gibFehlermeldungAus(String transition) {
 System.err.println(
 ↪ String.format("Im Zustand „%s“ kann die Transition „%s“ nicht ausgeführt werden!",
 zustand, transition));
 }

 public void starten() {
 gibFehlermeldungAus("starten");
 }

 public void suspendieren() {
 gibFehlermeldungAus("suspendieren");
 }

 public void fortsetzen() {
 gibFehlermeldungAus("fortsetzen");
 }

 public void beenden() {
 gibFehlermeldungAus("beenden");
 }

 public void abbrechen() {
 gibFehlermeldungAus("abbrechen");
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/prozess\\_verwaltung/ProzessZustand.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/ProzessZustand.java)

```
/**
 * Entspricht der „KonkreterZustand“-Unterklasse in der Terminologie der "Gang of
 * Four".
 */
public class ZustandAbgebrochen extends ProzessZustand {

 public ZustandAbgebrochen(Prozess prozess) {
 super("abgebrochen", prozess);
 }
}
```

```
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/prozess\\_verwaltung/ZustandAbgebrochen.java](#)

```
/**
 * Entspricht der „KonkreterZustand“-Unterklasse in der Terminologie der „Gang of
 * Four“.
 */
public class ZustandAktiv extends ProzessZustand {

 public ZustandAktiv(Prozess prozess) {
 super("aktiv", prozess);
 }

 public void suspendieren() {
 System.out.println("Der Prozess wird suspendiert.");
 prozess.setzeZustand(new ZustandSuspendiert(prozess));
 }

 public void beenden() {
 System.out.println("Der Prozess wird beendet.");
 prozess.setzeZustand(new ZustandBeendet(prozess));
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/prozess\\_verwaltung/ZustandAktiv.java](#)

```
/**
 * Entspricht der „KonkreterZustand“-Unterklasse in der Terminologie der „Gang of
 * Four“.
 */
public class ZustandBeendet extends ProzessZustand {

 public ZustandBeendet(Prozess prozess) {
 super("beendet", prozess);
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/prozess\\_verwaltung/ZustandBeendet.java](#)

```
/**
 * Entspricht der „KonkreterZustand“-Unterklasse in der Terminologie der „Gang of
 * Four“.
 */
public class ZustandBereit extends ProzessZustand {

 public ZustandBereit(Prozess prozess) {
 super("bereit", prozess);
 }

 public void starten() {
 System.out.println("Der Prozess wird gestartet.");
 }
}
```



```
 prozess.setzeZustand(new ZustandAktiv(prozess));
 }
}

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/ZustandBereit.java

/**
 * Entspricht der „KonkreterZustand“-Unterklasse in der Terminologie der „Gang of
 * Four“.
 */
public class ZustandSuspendiert extends ProzessZustand {

 public ZustandSuspendiert(Prozess prozess) {
 super("suspendiert", prozess);
 }

 public void fortsetzen() {
 System.out.println("Der Prozess wird fortgesetzt.");
 prozess.setzeZustand(new ZustandAktiv(prozess));
 }

 public void abbrechen() {
 System.out.println("Der Prozess wird abgebrochen.");
 prozess.setzeZustand(new ZustandAbgebrochen(prozess));
 }
}

Code-Beispiel auf Github ansehen:
src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/ZustandSuspendiert.java
```

## 66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 1

Antworten Sie kurz und prägnant.

(a) Nennen Sie einen Vorteil und einen Nachteil der Schichtenarchitektur.

Lösungsvorschlag

### Vorteil

Physische Datenunabhängigkeit:

Die interne Ebene ist von der konzeptionellen und externen Ebene getrennt.

Physische Änderungen, z. B. des Speichermediums oder des Datenbankprodukts, wirken sich nicht auf die konzeptionelle oder externe Ebene aus.

Logische Datenunabhängigkeit:

Die konzeptionelle und die externe Ebene sind getrennt. Dies bedeutet, dass Änderungen an der Datenbankstruktur (konzeptionelle Ebene) keine Auswirkungen auf die externe Ebene, also die Masken-Layouts, Listen und Schnittstellen haben.

a

**Nachteil**

Overhead durch zur Trennung der Ebenen benötigten Schnittstellen

<sup>a</sup><https://de.wikipedia.org/wiki/ANSI-SPARC-Architektur>

- (b) Wie ermöglicht es ein Datenbankensystem, verschiedene Sichten darzustellen?

Lösungsvorschlag

Die Sichten greifen auf die zwei darunterliegenden Abstraktionsebenen eines Datenbanksystems zu, nämlich auf die logische Ebene und die physische Ebene. Die physische Ebene greift auf die physische Ebene zu.

- (c) Was beschreibt das Konzept der Transitiven Hülle? Erklären Sie dies kurz und nennen Sie ein Beispiel für (1) die Transitive Hülle eines Attributes bei funktionalen Abhängigkeiten und (2) die Transitive Hülle einer SQL-Anfrage.

Lösungsvorschlag

Die transitive Hülle einer Relation  $R$  mit zwei Attributen  $A$  und  $B$  gleichen Typs ist definiert als

Sie enthält damit alle Tupel  $(a, b)$ , für die ein Pfad beliebiger Länge  $k$  in  $R$  existiert.

Berechnung rekursiver Anfragen (z. B. transitive Hülle) über rekursiv definierte Sichten (Tabellen)<sup>a</sup>

<sup>a</sup><https://dbs.uni-leipzig.de/file/dbs2-ss16-kap4.pdf>

- (d) Nennen Sie zwei Indexstrukturen und beschreiben Sie jeweils ihren Vorteil.

Lösungsvorschlag

In Hauptspeicher-Datenbanksystemen werden oft Hashtabellen verwendet, um effiziente Punkt-Abfragen (exact Match) zu unterstützen.

Wenn auch Bereichs-Abfragen (range queries) vorkommen, werden zumeister balancierte Suchbäume - AVL- oder rot/schwarz-Bäume - verwendet.

<sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/Indexstruktur>

## 66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 2

Entwerfen Sie ein ER-Diagramm für eine Schule aus einer imaginären Film-Reihe. Geben Sie alle Attribute an und unterstreichen Sie Schlüsselattribute. Für die Angabe der Kardinalitäten von Beziehungen soll die Min-Max-Notation verwendet werden. Führen Sie wenn nötig einen Surrogatschlüssel ein.

An der Schule werden Schüler ausgebildet. Sie haben einen Namen, ein Geschlecht und ein Alter. Da die Schule klein ist, ist der Name eindeutig. Jeder Schüler ist Teil

seines Jahrgangs, bestimmt durch Jahr und Anzahl an Schüler (Jahrgänge ohne Schüler sind erlaubt), und besucht mit diesem Kurse. Dabei wird jeder Kurs von min. einem Jahrgang besucht und jeder Jahrgang hat zwischen 2 und 5 Kurse. Kurse haben einen Veranstaltungsort und einen Namen. <sup>SQL</sup>

Außerdem wird jeder Schüler einem von vier Häusern zugeordnet. Diese Häuser sind Gryffindor, Slytherin, Hufflepuff und Ravenclaw. Jedes Haus hat eine Anzahl an Mitgliedern.

Um die Organisation an der Schule zu erleichtern, gibt es pro Haus einen Vertrauensschüler und pro Jahrgang einen Jahrgangssprecher. Außerdem können Schüler Quidditch spielen. Dabei können sie die Rollen Sucher, Treiber, Jäger und Hüter spielen. Jedes Haus der Schule hat eine Mannschaft. Diese besteht aus genau einem Sucher, einem Hüter, drei Jäger und zwei Treiber. Jedes Jahr gibt es an der Schule eine Trophäe zu gewinnen. Diese ist abhängig von der Mannschaft und weiterhin durch das Jahr identifiziert.

## 66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 3

Formulieren Sie folgende Anfragen in SQL gegen die angegebene Datenbank aus einer imaginären Serie.

Figur : {[ Id, Name, Schwertkunst, Lebendig, Titel ]}

gehört\_zu : {[ Id, Familie, FK (Id) references Figur(Id), FK (Familie) references Familie(Id) ]}

Familie : {[ Id, Name, Reichtum, Anführer ]}

Drache : {[ Name, Lebendig ]}

besitzt : {[ Id, Name, FK (Id) references Figur(Id), FK (Name) references Drache(Name) ]}

Festung : {[ Name, Ort, Ruine ]}

besetzt : {[ Familie, Festung, FK (Familie) references Familie(Id), FK (Festung) references Festung(Name) ]}

lebt : {[ Id, Name, FK (Id) references Figur(Id), FK (Name) references Festung(Name) ]}

```
CREATE TABLE Figur (
 Id integer PRIMARY KEY,
 Name VARCHAR(20),
 Schwertkunst integer,
 Lebendig boolean,
 Titel VARCHAR(50)
);
```

```
CREATE TABLE Familie (
 Id integer PRIMARY KEY,
 Name VARCHAR(20),
```

```
Reichtum numeric(11,2),
Anführer VARCHAR(20)
);

CREATE TABLE gehört_zu (
 Id integer REFERENCES Figur(id),
 Familie integer REFERENCES Familie(id)
);

CREATE TABLE Drache (
 Name VARCHAR(20) PRIMARY KEY,
 Lebendig boolean
);

CREATE TABLE besitzt (
 Id integer REFERENCES Figur(Id),
 Name VARCHAR(20) REFERENCES Drache(Name)
);

CREATE TABLE Festung (
 Name VARCHAR(20) PRIMARY KEY,
 Ort VARCHAR(30),
 Ruine boolean
);

CREATE TABLE besetzt (
 Familie integer REFERENCES Familie(Id),
 Festung VARCHAR(20) REFERENCES Festung(Name)
);

CREATE TABLE lebt (
 Id integer REFERENCES Figur(Id),
 Name VARCHAR(20) REFERENCES Festung(Name)
);

INSERT INTO Figur VALUES
(1, 'Eddard Stark', 5, FALSE, 'Lord von Winterfell'),
(2, 'Rodd Stark', 4, FALSE, 'Lord von Winterfell'),
(3, 'Tywin Lennister', 5, FALSE, 'Lord von Casterlystein'),
(4, 'Cersei Lennister', 2, TRUE, 'Lady von Casterlystein'),
(5, 'Brandon Stark', 0, TRUE, 'König der Andalen'),
(6, 'Jon Schnee', 5, TRUE, 'König-jenseits-der-Mauer');

INSERT INTO Familie VALUES
(1, 'Haus Stark', 76873.12, 'Eddard Stark'),
(2, 'Haus Lennister', 82345.43, 'Tywin Lennister');

INSERT INTO gehört_zu VALUES
(1, 1),
(2, 1),
(3, 2),
(4, 2),
(5, 1),
(6, 1);
```

```
INSERT INTO Festung VALUES
('Roter Bergfried', 'Westeros', FALSE),
('Casterlystein', 'Westeros', FALSE),
('Winterfell', 'Westeros', FALSE);
```

```
INSERT INTO besetzt VALUES
(1, 'Winterfell'),
(2, 'Roter Bergfried'),
(2, 'Casterlystein');
```

```
INSERT INTO lebt VALUES
(1, 'Winterfell'),
(2, 'Winterfell'),
(3, 'Casterlystein'),
(4, 'Roter Bergfried'),
(5, 'Winterfell'),
(6, 'Winterfell');
```

- (a) Geben Sie für alle Figuren an, wie oft alle vorhandenen Titel vorkommen.

Lösungsvorschlag

```
SELECT count(*) AS Anzahl, f.Titel
FROM Figur f
GROUP BY f.Titel;
```

| anzahl | titel                    |
|--------|--------------------------|
| 1      | König der Andalen        |
| 2      | Lord von Winterfell      |
| 1      | Lord von Casterlystein   |
| 1      | König-jenseits-der-Mauer |
| 1      | Lady von Casterlystein   |

(5 rows)

- (b) Welche Figuren (Name ist gesucht) kommen aus „Kings Landing“?

Lösungsvorschlag

```
SELECT
 f.Name
FROM
 Figur f,
 Festung b,
 lebt l
WHERE
 b.Name = l.Name AND
 f.Id = l.Id AND
 b.Ort = 'Königsmund';
```

- (c) Geben Sie für jede Familie (Name) die Anzahl der zugehörigen Charaktere und Festungen an.

Lösungsvorschlag

DELETE  
DROP COLUMN  
ALTER TABLE

```

SELECT
 f.Name,
 (SELECT COUNT(*) FROM Figur fi, gehört_zu ge WHERE fi.id = ge.id AND
 → ge.Familie = f.id) AS Anzahl_Charaktere,
 (SELECT COUNT(*) FROM Festung fe, besetzt be WHERE fe.Name = be.Festung AND
 → be.Familie = f.id) AS Anzahl_Festungen
FROM
 Familie f;

```

- (d) Gesucht sind die besten fünf Schwertkämpfer aus Festungen aus dem Ort „Westeros“. Es soll der Name, die Schwertkunst und die Platzierung ausgegeben werden. Die Ausgabe soll nach der Platzierung sortiert erfolgen.

Lösungsvorschlag

```

-- Problem: Es gibt 3 mal 3. Platz und nicht 3 mal 1. Platz
SELECT f1.Name, f1.Schwertkunst, COUNT(*) FROM Figur f1, Figur f2
WHERE f1.Schwertkunst <= f2.Schwertkunst
GROUP BY f1.Name, f1.Schwertkunst
ORDER BY COUNT(*)
LIMIT 5;

```

- (e) Schreiben Sie eine Anfrage, die alle Figuren löscht, die tot sind. Das Attribut *Lebendig* kann dabei die Optionen „ja“ und „nein“ annehmen.

Lösungsvorschlag

```

-- Nur zu Testzwecken auflisten:
SELECT * FROM Figur;
SELECT * FROM gehört_zu;

-- PostgreSQL unterstützt kein DELETE JOIN
-- DELETE f, g, b, l FROM Figur AS f
-- JOIN gehört_zu AS g ON f.id = g.id
-- JOIN besitzt AS b ON f.id = b.id
-- JOIN lebt AS l ON f.id = l.id
-- WHERE f.Lebendig = FALSE;

DELETE FROM gehört_zu WHERE id IN (SELECT id FROM Figur WHERE Lebendig =
 → FALSE);
DELETE FROM besitzt WHERE id IN (SELECT id FROM Figur WHERE Lebendig =
 → FALSE);
DELETE FROM lebt WHERE id IN (SELECT id FROM Figur WHERE Lebendig = FALSE);
DELETE FROM Figur WHERE Lebendig = FALSE;

-- Nur zu Testzwecken auflisten:
SELECT * FROM Figur;
SELECT * FROM gehört_zu;

```

- (f) Löschen Sie die Spalten „Lebendig“ aus der Datenbank.

```
ALTER TABLE Figur DROP COLUMN Lebendig;
ALTER TABLE Drache DROP COLUMN Lebendig;
```

- (g) Erstellen Sie eine weitere Tabelle mit dem Namen *Waffen*, welche genau diese auflistet. Eine Waffe ist genau einer Figur zugeordnet, hat einen eindeutigen Namen und eine Stärke zwischen 0 und 5. Wählen Sie sinnvolle Typen für die Attribute.

```
CREATE TABLE Waffen (
 Name VARCHAR(20) PRIMARY KEY,
 Figur integer NOT NULL REFERENCES Figur(Id),
 Stärke integer NOT NULL CHECK(Stärke BETWEEN 0 AND 5)
);

-- Sollte funktionieren:
INSERT INTO Waffen VALUES
 ('Axt', 1, 5);

-- Sollte Fehler ausgeben:
-- INSERT INTO Waffen VALUES
-- ('Schleuder', 1, 6);
```

## 66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 4

Übertragen Sie die folgenden Ausdrücke in die relationale Algebra. Beschreiben Sie diese Ausdrücke umgangssprachlich, bevor Sie die Umformung durchführen. Das Schema der Datenbank entspricht dem Schema aus Aufgabe 3.

- (a)  $\{f \mid f \in \text{Figur} \wedge \neg \exists g \in \text{gehört\_zu}(f.\text{Id} = g.\text{Id})\}$

Gesucht sind alle Figuren, die keiner Familie angehören.

$\text{Figur} - \pi_{\text{Id}, \text{Name}, \text{Schwertkunst}, \text{Lebendig}, \text{Titel}}(\text{Figur} \bowtie \text{gehört\_zu})$

- (b)  $\{f \mid f \in \text{Figur} \wedge \exists l \in \text{lebt}(l.\text{Id} = f.\text{Id}) \wedge \exists f_2 \in \text{Festung}(l.\text{Festung} = f.\text{Name}) \wedge \exists b \in \text{besetzt}(f_2.\text{Name} = b.\text{Festung}) \wedge \exists f_3 \in \text{Familie}(b.\text{Familie} = f_3.\text{Id}) \wedge f_3.\text{Name} = \text{Stark})\}$

In der Angabe steht  $\text{lebt}(l.\text{Id} = c.\text{Id})$  wurde abgeändert in  $\text{lebt}(l.\text{Id} = f.\text{Id})$ . Außerdem kommt  $f$  mehrmals vor. Wir wandeln  $f_2$  in  $f_3$  um und führen ein neues  $f_2$  ein. Diese schließende Klammer muss weg:  $\text{Familie}(b.\text{Familie} = f_3.\text{Id})$   
Übersichtlicher geschrieben

$$\begin{aligned}
&\{f \mid f \in \text{Figur} \wedge \\
&\quad \exists l \in \text{lebt}(l.\text{Id} = f.\text{Id}) \wedge \\
&\quad \exists f_2 \in \text{Festung}(l.\text{Festung} = f.\text{Name}) \wedge \\
&\quad \exists b \in \text{besetzt}(f_2.\text{Name} = b.\text{Festung}) \wedge \\
&\quad \exists f_3 \in \text{Familie}(b.\text{Familie} = f_3.\text{Id} \wedge f_3.\text{Name} = \text{Stark}) \\
&\quad \}
\end{aligned}$$

Gesucht sind alle Figuren, die in einer von der Familie „Stark“ besetzten Festung leben.

## 66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 5

(a) Gegeben sind folgende funktionale Abhängigkeiten.

$$\text{FA} = \left\{ \begin{array}{l} \{X\} \rightarrow \{Y, Z\}, \\ \{Z\} \rightarrow \{W, X\}, \\ \{Q\} \rightarrow \{X, Y, Z\}, \\ \{V\} \rightarrow \{Z, W\}, \\ \{Z, W\} \rightarrow \{Y, Q, V\}, \end{array} \right\}$$

Berechnen Sie die kanonische Überdeckung.

Lösungsvorschlag

### (i) Linksreduktion

— Führe für jede funktionale Anhängigkeit  $\alpha \rightarrow \beta \in F$  die Linksreduktion durch, überprüfe also für alle  $A \in \alpha$ , ob  $A$  überflüssig ist, d. h. ob  $\beta \subseteq \text{AttrHülle}(F, \alpha - A)$ . —

$$\{Z, W\} \rightarrow \{Y, Q, V\}$$

$$\{Y, Q, V\} \in \text{AttrHülle}(F, \{Z, W \setminus W\}) = \{Q, V, W, Y, Z\}$$

$$\text{FA} = \left\{ \begin{array}{l} \{X\} \rightarrow \{Y, Z\}, \\ \{Z\} \rightarrow \{W, X\}, \\ \{Q\} \rightarrow \{X, Y, Z\}, \\ \{V\} \rightarrow \{Z, W\}, \\ \{Z\} \rightarrow \{Y, Q, V\}, \end{array} \right\}$$

### (ii) Rechtsreduktion



— Führe für jede (verbliebene) funktionale Abhängigkeit  $\alpha \rightarrow \beta$  die Rechtsreduktion durch, überprüfe also für alle  $B \in \beta$ , ob  $B \in \text{AttrHülle}(F - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B)), \alpha)$  gilt. In diesem Fall ist  $B$  auf der rechten Seite überflüssig und kann eliminiert werden,  $\delta\alpha \rightarrow \beta$  wird durch  $\alpha \rightarrow (\beta - B)$  ersetzt.

Auf der rechten Seite kommen mehrfach vor: W, X, Y, Z

**Y**

$$Y \in \text{AttrHülle}(F \setminus \{X\} \rightarrow \{Y, Z\} \cup \{X\} \rightarrow \{Z\}, \{X\}) = \{Q, V, W, X, Y, Z\}$$

$$\text{FA} = \left\{ \begin{array}{l} \{X\} \rightarrow \{Z\}, \\ \{Z\} \rightarrow \{W, X\}, \\ \{Q\} \rightarrow \{X, Y, Z\}, \\ \{V\} \rightarrow \{Z, W\}, \\ \{Z\} \rightarrow \{Y, Q, V\}, \end{array} \right\}$$

$$Y \in \text{AttrHülle}(F \setminus \{Q\} \rightarrow \{X, Y, Z\} \cup \{Q\} \rightarrow \{X, Z\}, \{Q\}) = \{Q, V, W, X, Y, Z\}$$

$$\text{FA} = \left\{ \begin{array}{l} \{X\} \rightarrow \{Z\}, \\ \{Z\} \rightarrow \{W, X\}, \\ \{Q\} \rightarrow \{X, Z\}, \\ \{V\} \rightarrow \{Z, W\}, \\ \{Z\} \rightarrow \{Y, Q, V\}, \end{array} \right\}$$

**Z**

$$Z \notin \text{AttrHülle}(F \setminus \{X\} \rightarrow \{Z\}, \{X\}) = \{X\}$$

$$Z \in \text{AttrHülle}(F \setminus \{Q\} \rightarrow \{X, Z\} \cup \{Q\} \rightarrow \{X\}, \{Q\}) = \{Q, V, W, X, Y, Z\}$$

$$\text{FA} = \left\{ \begin{array}{l} \{X\} \rightarrow \{Z\}, \\ \{Z\} \rightarrow \{W, X\}, \\ \{Q\} \rightarrow \{X\}, \\ \{V\} \rightarrow \{Z, W\}, \\ \{Z\} \rightarrow \{Y, Q, V\}, \end{array} \right\}$$

$$Z \notin \text{AttrHülle}(F \setminus \{V\} \rightarrow \{Z, W\} \cup \{V\} \rightarrow \{W\}, \{V\}) = \{V, W\}$$

**W**

$$W \in \text{AttrHülle}(F \setminus \{Z\} \rightarrow \{W, X\} \cup \{Z\} \rightarrow \{X\}, \{Z\}) = \{Q, V, W, X, Y, Z\}$$

$$\text{FA} = \left\{ \begin{array}{l} \{X\} \rightarrow \{Z\}, \\ \{Z\} \rightarrow \{X\}, \\ \{Q\} \rightarrow \{X\}, \\ \{V\} \rightarrow \{Z, W\}, \\ \{Z\} \rightarrow \{Y, Q, V\}, \end{array} \right\}$$

**X**

$$X \in \text{AttrHülle}(F \setminus \{Z\} \rightarrow \{X\}, \{Z\}) = \{Q, V, W, X, Y, Z\}$$

$$\text{FA} = \left\{ \begin{array}{l} \{X\} \rightarrow \{Z\}, \\ \{Z\} \rightarrow \{\emptyset\}, \\ \{Q\} \rightarrow \{X\}, \\ \{V\} \rightarrow \{Z, W\}, \\ \{Z\} \rightarrow \{Y, Q, V\}, \end{array} \right\}$$

(iii) **Löschen leerer Klauseln**

— Entferne die funktionalen Abhängigkeiten der Form  $\alpha \rightarrow \emptyset$ , die im 2. Schritt möglicherweise entstanden sind. \_\_\_\_\_

$$\text{FA} = \left\{ \begin{array}{l} \{X\} \rightarrow \{Z\}, \\ \{Q\} \rightarrow \{X\}, \\ \{V\} \rightarrow \{Z, W\}, \\ \{Z\} \rightarrow \{Y, Q, V\}, \end{array} \right\}$$

(iv) **Vereinigung**

— Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form  $\alpha \rightarrow \beta_1, \dots, \alpha \rightarrow \beta_n$ , so dass  $\alpha \rightarrow \beta_1 \cup \dots \cup \beta_n$  verbleibt. \_\_\_\_\_

$\emptyset$  Nichts zu tun

(b) Gegeben ist folgende Tabelle.

| JedID | Name             | Rasse     | Lichtschwert | Seite der Macht          |
|-------|------------------|-----------|--------------|--------------------------|
| 2     | Yoda             | Unbekannt | Grün         | Gute Seite               |
| 3     | Anakin Skywalker | Mensch    | Blau, Rot    | Gute Seite, Dunkle Seite |
| 4     | Mace Windou      | Mensch    | Lila         | Gute Seite               |
| 5     | Count Dooku      | Mensch    | Rot          | Dunkle Seite             |
| 6     | Ahsoka Tano      | Togruta   | Grün         | Gute Seite               |
| 7     | Yoda             | Mensch    | Rot          | Dunkle Seite             |

Geben Sie zuerst die funktionalen Abhängigkeiten in der Tabelle an.

Lösungsvorschlag

$$FA = \left\{ \begin{array}{l} \{ JedID \} \rightarrow \{ Name, Rasse, Lichtschwert, SeitederMacht \}, \\ \{ Lichtschwert \} \rightarrow \{ SeitederMacht \}, \end{array} \right\}$$

JedID ist ein Surrogat-Schlüssel<sup>a</sup>, ðein künstlich eingeführter Primärschlüssel, von dem alle Attribute abhängen.

Ein rotes Lichtschwert zeigt an, dass der Jedi-Ritter zur dunklen Seite der Macht gehört. Grüne, blaue und lila Lichtschwerter zeigen an, dass der Jedi-Ritter zu guten Seite gehört. Wir können die Funktionale Abhängigkeit nicht umdrehen, weil wir nicht von der guten Seite der Macht auf die Farbe schließen können.

<sup>a</sup><https://de.wikipedia.org/wiki/Surrogatschlüssel>

(c)

(i) Geben Sie die zentrale Eigenschaft der 1. NF an.

Lösungsvorschlag

Eine Relation befindet sich in erster Normalform (1NF), wenn sie ausschließlich atomare Attributwerte enthält.

(ii) Nennen Sie alle Stellen, an denen das Schema die 1. NF verletzt.

Lösungsvorschlag

Im Tupel (JedID = 3) haben die Attribute *Lichtschwert* und *Seite der Macht* mehrwertige Attribute.

(iii) Überführen Sie die Tabelle in die 1. NF.

Lösungsvorschlag

| <u>JedID</u> | Name               | Rasse         | Lichtschwert | Seite der Macht     |
|--------------|--------------------|---------------|--------------|---------------------|
| 2            | Yoda               | Unbekannt     | Grün         | Gute Seite          |
| 3            | Anakin Skywalker   | Mensch        | Blau         | Gute Seite          |
| 4            | Mace Windou        | Mensch        | Lila         | Gute Seite          |
| 5            | Count Dooku        | Mensch        | Rot          | Dunkle Seite        |
| 6            | Ahsoka Tano        | Togruta       | Grün         | Gute Seite          |
| 7            | Yoda               | Mensch        | Rot          | Dunkle Seite        |
| 8            | <b>Darth Vader</b> | <b>Mensch</b> | <b>Rot</b>   | <b>Dunkle Seite</b> |

(d)

(i) Geben Sie die Definition der 2. NF an.

Lösungsvorschlag

Eine Relation in in zweiter Normalform (2NF), wenn sie in 1NF und jedes Nichtschlüsselattribut von jedem Schlüsselkandidaten voll funktional abhängig ist.

(ii) Arbeiten Sie bitte mit folgender, nicht korrekten Zwischenlösung weiter. Erläutern Sie, inwiefern dieses Schema die 2. NF verletzt.

| JedID | Name      | Rasse     | Lichtschwert | Seite der Macht |
|-------|-----------|-----------|--------------|-----------------|
| 2     | Yoda      | Unbekannt | Grün         | Gute Seite      |
| 3     | Skywalker | Mensch    | Blau         | Gute Seite      |
| 4     | Windou    | Mensch    | Lila         | Gute Seite      |
| 5     | Dooku     | Mensch    | Rot          | Dunkle Seite    |
| 6     | Tano      | Togruta   | Grün         | Gute Seite      |
| 2     | Yoda      | Mensch    | Rot          | Dunkle Seite    |

(iii) Überführen Sie die Tabelle in die 2. NF.

(e) 5.

(i) Geben Sie die Definition der 3. NF an.

Lösungsvorschlag

Eine Relation befindet sich in der dritten Normalform (3NF), wenn keine transitiven Abhängigkeiten der Nichtschlüsselattribute existieren.

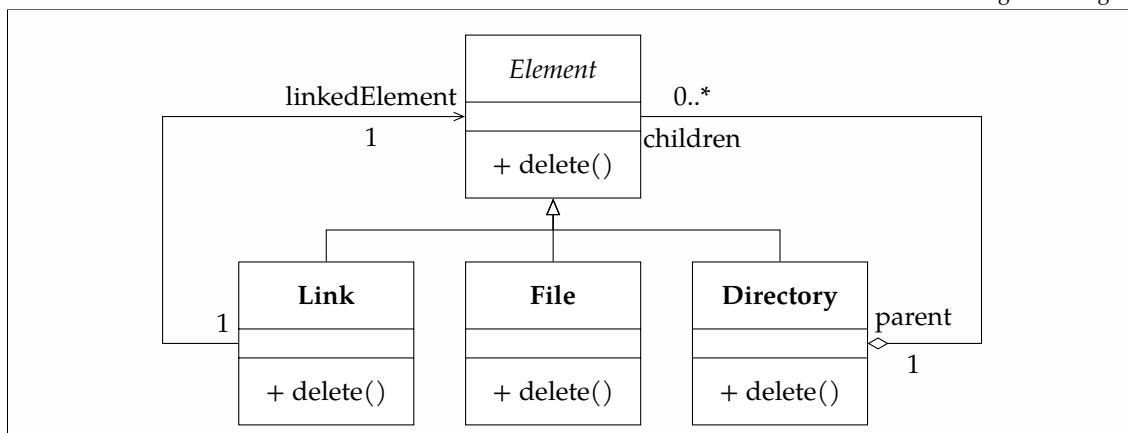
(ii) Erläutern Sie, ob und wenn ja, wie das von Ihnen in 3c) neu erstellte Schema die 3. NF verletzt.

## 66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1

Hierarchische Dateisysteme bestehen aus den FileSystemElements Ordner, Dateien und Verweise. Ein Ordner kann seinerseits Ordner, Dateien und Verweise beinhalten; jedem Ordner ist bekannt, welche Elemente (children) er enthält. Mit Ausnahme des Root-Ordners auf der obersten Hierarchieebene ist jeder Ordner, jede Datei und jeder Verweis Element eines Elternordners. Jedem Element ist bekannt, was sein Elternordner ist (parent). Ein Verweis verweist auf einen Verweis, eine Datei oder einen Ordner (link). Wenn ein Ordner gelöscht wird, werden alle seine Bestandteile ggf. rekursiv ebenfalls gelöscht. Sie dürfen die Lösungen für Aufgabenteil b) und c) in einem gemeinsamen Code kombinieren.

- (a) Modellieren Sie diesen Sachverhalt mit einem UML-Klassendiagramm. Benennen Sie die Rollen von Assoziationen und geben Sie alle Kardinalitäten an. Ihre Lösung soll mindestens eine sinnvolle Spezialisierungsbeziehung enthalten.

Lösungsvorschlag



- (b) Implementieren Sie das Klassendiagramm als Java- oder C++-Programm. Jedes Element des Dateisystems soll mindestens über ein Attribut `name` verfügen. Übergeben Sie den Elternordner jedes Elements als Parameter an den Konstruktor; der Elternordner des Root-Ordners kann dabei als `null` implementiert werden. Dokumentieren Sie Ihren Code.

*a*

```
public abstract class Element {

 protected String name;

 protected Element parent;

 protected Element(String name, Element parent) {
 this.name = name;
 this.parent = parent;
 }

 public String getName() {
 return name;
 }
}
```

```

}

public abstract void delete();

public abstract boolean isDirectory();

public abstract void addChild(Element child);
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/file\\_system/Element.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/file_system/Element.java)

```

import java.util.ArrayList;
import java.util.List;

public class Directory extends Element {
 private List<Element> children;

 public Directory(String name, Element parent) {
 super(name, parent);
 children = new ArrayList<Element>();
 if (parent != null)
 parent.addChild(this);
 }

 public void delete() {
 System.out.println("The directory " + name +
 ↪ " was deleted and it's children were also deleted.");
 for (int i = 0; i < children.size(); i++) {
 Element child = children.get(i);
 child.delete();
 }
 }

 public void addChild(Element child) {
 children.add(child);
 }

 public boolean isDirectory() {
 return true;
 }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/file\\_system/Directory.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/file_system/Directory.java)

```

public class File extends Element {

 public File(String name, Element parent) {
 super(name, parent);
 parent.addChild(this);
 }

 public void delete() {
 System.out.println("The File " + name + " was deleted.");
 }
}

```

```

public boolean isDirectory() {
 return false;
}

/**
 * Eine Datei kann keine Kinder haben. Deshalb eine Methode mit leerem
 * Methodenrumpf.
 */
public void addChild(Element child) {
}
}

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/file_system/File.java

public class Link extends Element {

 private Element linkedElement;

 public Link(String name, Element parent, Element linkedElement) {
 super(name, parent);
 this.linkedElement = linkedElement;
 parent.addChild(this);
 }

 public void delete() {
 System.out.println("The Symbolic Link " + name + " was deleted.");
 linkedElement.delete();
 System.out.println("The linked element " + name + " was deleted too.");
 }

 public void addChild(Element child) {
 if (linkedElement.isDirectory())
 linkedElement.addChild(child);
 }

 public boolean isDirectory() {
 return linkedElement.isDirectory();
 }
}

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/herbst/file_system/Link.java

```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen\_66116/jahr\_2019/herbst/file\_system/Link.java

<sup>a</sup>TU Darmstadt: Dr. Michael Eichberg - Case Study Using the Composite and Proxy Design Patterns

- (c) Ordnen Sie eine Methode `delete`, die Dateien, Ordner und Verweise rekursiv löscht, einer oder mehreren geeigneten Klassen zu und implementieren Sie sie. Zeigen Sie die Löschung jedes Elements durch eine Textausgabe von `name` an. `toString()` müssen Sie dabei nicht implementieren. Gehen Sie davon aus, dass Verweis- und Ordnerstrukturen azyklisch sind und dass jedes Element des Dateisystems höchstens einmal existiert. Wenn ein Verweis gelöscht wird, wird sowohl der Verweis als auch das verwiesene Element bzw. transitiv die Kette der



verwiesenen Elemente gelöscht. Bedenken Sie, dass die Löschung eines Elements immer auch Konsequenzen für den dieses Element beinhaltenden Ordner hat. Es gibt keinen Punktabzug, wenn Sie die Löschung des Root-Ordners nicht zulassen.

Lösungsvorschlag

Siehe Antwort zu Aufgabe b)

- (d) Was kann im Fall von `delete` passieren, wenn die Linkstruktur zyklisch ist oder die Ordnerstruktur zyklisch ist? Kann es zu diesen Problemen auch dann führen, wenn weder die Linkstruktur zyklisch ist, noch die Ordnerstruktur zyklisch ist? Wie kann man im Programm das Problem lösen, falls man Zyklizitäten zulassen möchte?

Lösungsvorschlag

Falls die Link- oder Ordnerstruktur zyklisch ist, kann es aufgrund der Rekursion zu einer Endlosschleife kommen. Diese Problem tritt bei azyklischen Strukturen nicht auf, weil der rekursive Löschvorgang beim letzten Element abgebrochen wird (Abbruchbedingung).

Das Problem kann zum Beispiel durch ein neues Attribut `gelöscht` in der Klasse `Link` oder `Directory` gelöst werden. Dieses Attribut wird auf `true` gesetzt, bevor es in die Rekursion einsteigt. Rufen sich die Klassen wegen der Zyklizität selbst wieder auf, kommt es durch entsprechende IF-Bedingungen zum Abbruch.

Außerdem ist ein Zähler denkbar, der sich bei jeder Rekursion hochsetzt und ab einem gewissen Grenzwert zum Abbruch führt.

- (e) Was ist ein Design Pattern? Nennen Sie drei Beispiele und erläutern Sie sie kurz. Welches Design Pattern bietet sich für die Behandlung von hierarchischen Teil-Ganzes-Beziehungen an, wie sie im Beispiel des Dateisystems vorliegen?

Lösungsvorschlag

Design Pattern sind wiederkehrende, geprüfte, bewährte Lösungsschablonen für typische Probleme.

### Drei Beispiele

**Einzelstück (Singleton)** Stellt sicher, dass nur *genau eine Instanz einer Klasse* erzeugt wird.

**Beobachter (Observer)** Das Observer-Muster ermöglicht einem oder mehreren Objekten, automatisch auf die *Zustandsänderung* eines bestimmten Objekts zu *reagieren*, um den eigenen Zustand anzupassen.

**Stellvertreter (Proxy)** Ein Proxy stellt einen Platzhalter für eine andere Komponente (Objekt) dar und kontrolliert den Zugang zum echten Objekt.

Für hierarchischen Teil-Ganzes-Beziehungen eignet sich das Kompositum (Com-

posite). Es ermöglicht die Gleichbehandlung von Einzelementen und Elementgruppierungen in einer verschachtelten Struktur (z. B. Baum), sodass aus Sicht des Clients keine explizite Unterscheidung notwendig ist.

## 66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2

Methoden in Programmen funktionieren nicht immer für alle möglichen Eingaben - klassische Beispiele sind die Quadratwurzel einer negativen Zahl oder die Division durch Null. Zulässige (bzw. in negierter Form unzulässige) Eingabewerte sollten dann spezifiziert werden, was mit Hilfe sog. assertions geschehen kann. Assertions prüfen zur Laufzeit den Wertebereich der Parameter einer Methode, bevor der Methodenkörper ausgeführt wird. Wenn ein oder mehrere Argumente zur Laufzeit unzulässig sind, wird eine Ausnahme geworfen.

Betrachten Sie das folgende Java-Programm:

```
public static double[][] magic(double[][] A, double[][] B, int m) {
 double[][] C = new double[m][m];
 for (int i = 0; i < m; i++)
 for (int j = 0; j < m; j++)
 for (int k = 0; k < m; k++)
 C[i][j] += A[i][k] * B[k][j];
 return C;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/Assertion.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/Assertion.java)

(a) Beschreiben Sie kurz, was dieses Programm tut.

Lösungsvorschlag

Das Programm erzeugt ein neues zweidimensionales Feld mit  $m \times m$  Einträgen.

Zelle an Kreuzung Zeile Z und Spalte S =

$$\begin{aligned} &A[1. \text{ Wert in } Z] \times B[1. \text{ Wert in } S] + \\ &A[2. \text{ Wert in } Z] \times B[2. \text{ Wert in } S] + \\ &A[3. \text{ Wert in } Z] \times B[3. \text{ Wert in } S] \end{aligned}$$

Zelle an Kreuzung 1. Zeile 1. Spalte =

$$\begin{aligned} &1 \times 11 + \\ &2 \times 16 + \\ &3 \times 17 \end{aligned}$$

**A**

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

**B**

|    |    |    |
|----|----|----|
| 11 | 12 | 13 |
| 16 | 15 | 14 |
| 17 | 18 | 19 |

`magic(A, B, 3);:`

|     |     |     |
|-----|-----|-----|
| 94  | 96  | 98  |
| 226 | 231 | 236 |
| 358 | 366 | 374 |

$C[0][0] = C[0][0] + A[0][0] * B[0][0] = 0.0 + 1.0 * 11.0 = 11.0$   
 $C[0][0] = C[0][0] + A[0][1] * B[1][0] = 11.0 + 2.0 * 16.0 = 43.0$   
 $C[0][0] = C[0][0] + A[0][2] * B[2][0] = 43.0 + 3.0 * 17.0 = 94.0$

$C[0][1] = C[0][1] + A[0][0] * B[0][1] = 0.0 + 1.0 * 12.0 = 12.0$   
 $C[0][1] = C[0][1] + A[0][1] * B[1][1] = 12.0 + 2.0 * 15.0 = 42.0$   
 $C[0][1] = C[0][1] + A[0][2] * B[2][1] = 42.0 + 3.0 * 18.0 = 96.0$

$C[0][2] = C[0][2] + A[0][0] * B[0][2] = 0.0 + 1.0 * 13.0 = 13.0$   
 $C[0][2] = C[0][2] + A[0][1] * B[1][2] = 13.0 + 2.0 * 14.0 = 41.0$   
 $C[0][2] = C[0][2] + A[0][2] * B[2][2] = 41.0 + 3.0 * 19.0 = 98.0$

$C[1][0] = C[1][0] + A[1][0] * B[0][0] = 0.0 + 4.0 * 11.0 = 44.0$   
 $C[1][0] = C[1][0] + A[1][1] * B[1][0] = 44.0 + 5.0 * 16.0 = 124.0$   
 $C[1][0] = C[1][0] + A[1][2] * B[2][0] = 124.0 + 6.0 * 17.0 = 226.0$

$C[1][1] = C[1][1] + A[1][0] * B[0][1] = 0.0 + 4.0 * 12.0 = 48.0$   
 $C[1][1] = C[1][1] + A[1][1] * B[1][1] = 48.0 + 5.0 * 15.0 = 123.0$   
 $C[1][1] = C[1][1] + A[1][2] * B[2][1] = 123.0 + 6.0 * 18.0 = 231.0$

$C[1][2] = C[1][2] + A[1][0] * B[0][2] = 0.0 + 4.0 * 13.0 = 52.0$   
 $C[1][2] = C[1][2] + A[1][1] * B[1][2] = 52.0 + 5.0 * 14.0 = 122.0$   
 $C[1][2] = C[1][2] + A[1][2] * B[2][2] = 122.0 + 6.0 * 19.0 = 236.0$

$C[2][0] = C[2][0] + A[2][0] * B[0][0] = 0.0 + 7.0 * 11.0 = 77.0$   
 $C[2][0] = C[2][0] + A[2][1] * B[1][0] = 77.0 + 8.0 * 16.0 = 205.0$   
 $C[2][0] = C[2][0] + A[2][2] * B[2][0] = 205.0 + 9.0 * 17.0 = 358.0$

$C[2][1] = C[2][1] + A[2][0] * B[0][1] = 0.0 + 7.0 * 12.0 = 84.0$   
 $C[2][1] = C[2][1] + A[2][1] * B[1][1] = 84.0 + 8.0 * 15.0 = 204.0$   
 $C[2][1] = C[2][1] + A[2][2] * B[2][1] = 204.0 + 9.0 * 18.0 = 366.0$

$C[2][2] = C[2][2] + A[2][0] * B[0][2] = 0.0 + 7.0 * 13.0 = 91.0$

```
C[2][2] = C[2][2] + A[2][1] * B[1][2] = 91.0 + 8.0 * 14.0 = 203.0
C[2][2] = C[2][2] + A[2][2] * B[2][2] = 203.0 + 9.0 * 19.0 = 374.0
```

- (b) Implementieren Sie drei nützliche Assertions, die zusammen verhindern, dass das Programm abstürzt.

Lösungsvorschlag

```
private static void assert2DArray(double[][] a, int m) {
 assert a.length < m :
 ↪ "Das 2D-Feld muss mindestens m Zeilen/Felder haben.";
 for (int i = 0; i < a.length; i++) {
 double[] row = a[i];
 assert row.length < m:
 ↪ "Jede Zeile im 2D-Feld muss mindestens m Einträge/Spalten haben.";
 }
}

public static double[][] magicWithAssertions(double[][] A, double[][] B,
 ↪ int m) {
 assert m < 0: "m darf nicht negativ sein.";
 assert2DArray(A, m);
 assert2DArray(B, m);
 double[][] C = new double[m][m];
 for (int i = 0; i < m; i++)
 for (int j = 0; j < m; j++)
 for (int k = 0; k < m; k++) {
 C[i][j] += A[i][k] * B[k][j];
 }
 return C;
}

/**
 * Duplikat der magic-Methode mit einer println-Anweisung, um die
 * Funktionsweise
 * ↪ besser untersuchen zu können.
 *
 * @param A Ein zwei-dimensionales Feld des Datentyps double. Sollte
 * ↪ mindestens
 *
 * m x m Einträge haben.
 */
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/Assertion.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/Assertion.java)

- (c) Wann und warum kann es sinnvoll sein, keine expliziten Assertions anzugeben? Erläutern Sie, warum das potentiell gefährlich ist.

Lösungsvorschlag

Bei sicherheitskritischen Anwendungen, z. B. bei Software im Flugzeug, kann es möglicherweise besser sein, einen Fehler zu ignorieren, als die Software abstürzen zu lassen. In Java beispielsweise sind die Assertions standardmäßig deaktiviert und müssen erst mit `-enableassertions` aktiviert werden.

- (d) Assertions können wie oben sowohl für Vorbedingungen am Anfang einer Methode als auch für Nachbedingungen am Ende einer Methode verwendet werden.

Solche Spezifikationen bilden dann sog. Kontrakte. Kontrakte werden insbesondere auch statisch verwendet, önicht zur Laufzeit. Skizzieren Sie ein sinnvolles Anwendungsgebiet und beschreiben Sie kurz den Vorteil der Verwendung von Kontrakten in diesem Anwendungsgebiet.

Lösungsvorschlag

*a*<sup>a</sup>[https://de.wikipedia.org/wiki/Design\\_by\\_contract](https://de.wikipedia.org/wiki/Design_by_contract)

## 66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3

Die Komponentenarchitektur eines Softwaresystems beschreibt die unterschiedlichen Softwarebausteine, deren Schnittstellen und die Abhängigkeiten von Softwarekomponenten wie beispielsweise Klassen. Wir unterscheiden zwischen der Soll- und der Ist-Architektur eines Systems.

- (a) Nennen und definieren Sie drei Qualitätsattribute von Software, die durch die Architektur beeinflusst werden und charakterisieren Sie jeweils eine „schlechte“ Architektur, die diese Qualitätsattribute negativ beeinflusst.

Lösungsvorschlag

**Skalierbarkeit** Definition: Ob die Software auch für große Zugriffslasten konzipiert wurde. Charakterisierung einer schlechten Architektur: Eine Anwendung läuft nur auf einem Server.

**Modifizierbarkeit** Definition: Ob die Software leicht verändert, erweitert werden kann. Charakterisierung einer schlechten Architektur: Monolithische Architektur, dass kein Laden von Modulen zulässt.

**Verfügbarkeit** Definition: Ob die Software ausfallsicher ist, im Laufenden Betrieb gewartet werden kann. Charakterisierung einer schlechten Architektur: Ein-Server-Architektur, die mehrmal neugestartet werden muss, um ein Update einzuspielen.

- (b) Erläutern Sie, was Information Hiding ist. Wie hängt Information Hiding mit Softwarearchitektur zusammen? Wie wird Information Hiding auf Klassenebene in Java implementiert? Gibt es Situationen, in denen Information Hiding auch negative Effekte haben kann?

Lösungsvorschlag

Das Prinzip der Trennung von Zuständigkeiten (engl. separation of concerns) sorgt dafür, dass jede Komponente einer Architektur nur für eine einzige Aufgabe zuständig ist. Das Innenleben von Komponenten wird durch Schnittstellen verkapselt, was auf das Prinzip des Verbergens von Informationen (engl. information hiding) zurückgeht.

Java: Durch die Sichtbarkeits-Schlüsselwörter: private, protected

Zusätzlicher Overhead durch Schnittstellen API zwischen den Modulen.

(c) Erklären Sie, was Refactoring ist.

Lösungsvorschlag

Verbesserungen des Code durch bessere Lesbarkeit, Wartbarkeit, Performanz, Sicherheit. Keine neuen Funktionen werden programmiert.

(d) Skizzieren Sie die Kernideen von Scrum inkl. der wesentlichen Prozessschritte, Artefakte und Rollen. Beschreiben Sie dann die Rolle von Ist- und Soll-Architektur in agilen Entwicklungskontexten wie Scrum.

## 66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Erstellen Sie ein möglichst einfaches ER-Schema, das alle gegebenen Informationen enthält. Attribute von Entitäten und Beziehungen sind anzugeben, Schlüsselattribute durch Unterstreichen zu kennzeichnen. Verwenden Sie für die Angabe der Kardinalitäten von Beziehungen die Min-MaxNotation. Führen Sie Surrogatschlüssel nur dann ein, wenn es nötig ist und modellieren Sie nur die im Text vorkommenden Elemente.

Ein örtlicher Sportverein möchte seine Vereinsangelegenheiten mittels einer Datenbank verwalten. Der Verein besteht aus verschiedenen Abteilungen, welche eine eindeutige Nummer und einen aussagekräftigen Namen besitzen. Für jede Abteilung soll zudem automatisch die Anzahl der Mitglieder gespeichert werden, wobei ein Mitglied zu mehreren Abteilungen gehören kann. Die Mitglieder des Vereins können keine, eine oder mehrere Rollen (auch Ämter genannt) einnehmen. So gibt es die Ämter: 1. Vorstand, 2. Vorstand, Kassier, Jugendleiter, Trainer sowie einen Abteilungsleiter für jede Abteilung. Es ist dabei auch möglich, dass ein Abteilungsleiter mehrere Abteilungen leitet oder ein Mitglied mehrere Aufgaben übernimmt, mit der Einschränkung, dass die Vorstandsposten und Kassier nicht von der gleichen Person ausgeübt werden dürfen. Zu jedem Trainer wird eine Liste von Lizenzen gespeichert. Jeder Trainer ist zudem in mindestens einer Abteilung eine bestimmte Anzahl von Stunden tätig. Zu allen Mitgliedern werden Mitgliedsnummer, Name (bestehend aus Vor- und Nachname), Geburtsdatum, E-Mail, Eintrittsdatum, Adresse (bestehend aus PLZ, Ort, Straße, Hausnummer), IBAN und die Vereinszugehörigkeit in Jahren gespeichert.

Im Verein fallen Finanztransaktionen an. Zu jeder Transaktion wird ein Zeitstempel, der Betrag und eine eindeutige Transaktionsnummer gespeichert. Die Mitglieder leisten Zahlungen an den Verein. Umgekehrt erstattet der Verein auch bestimmte Kosten. Im Verein existieren drei verschiedene Mitgliedsbeiträge. So gibt es einen Kinder- und Jugendlichen-Tarif, einen Erwachsenentarif und einen Familientarif.

Mitgliedern entstehen des Öfteren Fahrtkosten. Für jede Fahrtkostenabrechnung werden das Datum, die gefahrenen Kilometer und Start und Ziel, der Zweck sowie das Mitglied gespeichert, welches den Antrag gestellt hat. Zu jeder Fahrtkostenabrechnung existiert genau eine Erstattung.

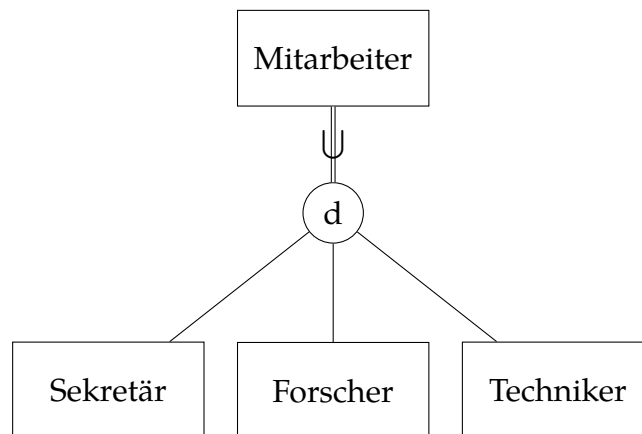
Durch die Teilnahme an verschiedenen Wettbewerben besteht die Notwendigkeit die von einem Team (also mehreren Mitgliedern zusammen) oder Mitgliedern erzielten sportlichen Erfolge, d.h. Platzierungen, zu verwalten. Jeder Wettkampf besitzt eine eindeutige ID, ein Datum und eine Kurzbeschreibung.

Das Vereinsleben besteht aus zahlreichen Terminen, die durch Datum und Uhrzeit innerhalb einer

Abteilung eindeutig identifiziert werden können. Zu jedem Termin wird zusätzlich eine Kurzbeschreibung gespeichert.

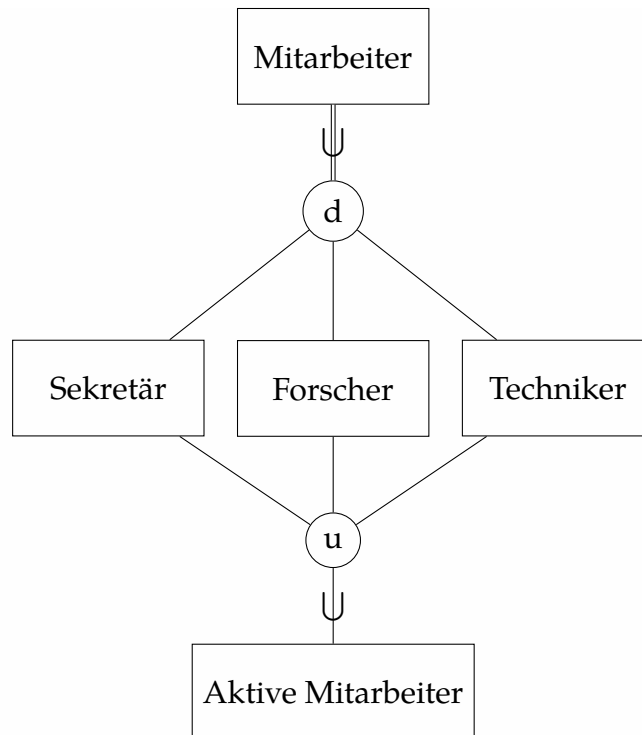
## 66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2

In einer Datenbank zur Mitarbeiterverwaltung werden die Mitarbeiter über ihr Ausscheiden aus dem Betrieb hinaus (z. B. Ruhestand oder Arbeitsplatzwechsel) gespeichert. Im Folgenden ist ein Ausschnitt aus dem ER-Diagramm dargestellt. Erweitern Sie das Diagramm um genau eine Entität, welche die derzeit aktiven Mitarbeiter aus allen Unterklassen umfasst. Benennen und erläutern Sie das von Ihnen verwendete Modellierungskonstrukt.



Lösungsvorschlag

EER-Notation nach Elmasri/Navathe



U = Vereinigungsmenge

### 66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 3

Gegeben seien die folgenden beiden Relationen:

**R1**

| P  | Q | S |
|----|---|---|
| 10 | a | 5 |
| 15 | b | 8 |
| 25 | a | 6 |

**R2**

| A  | B | C |
|----|---|---|
| 10 | b | 6 |
| 25 | c | 3 |
| 10 | b | 5 |

Geben Sie die Ergebnisse der folgenden relationalen Ausdrücke an:

(a)  $R1 \bowtie_{R1.P=R2.A} R2$  (Equi-Join)

Lösungsvorschlag

| P  | Q | S | A  | B | C |
|----|---|---|----|---|---|
| 10 | a | 5 | 10 | b | 6 |
| 10 | a | 5 | 10 | b | 5 |
| 25 | a | 6 | 25 | c | 3 |

(b)  $R1 \bowtie_{R1.Q=R2.B} R2$  (Right-Outer-Join)



| P  | Q | S | A  | B | C |
|----|---|---|----|---|---|
| 15 | b | 8 | 10 | b | 6 |
| 15 | b | 8 | 10 | b | 5 |
|    |   |   | 25 | c | 3 |

- (c) Es ist bekannt, dass die minimale Menge relationaler Operatoren Selektion, Projektion, Vereinigung, Differenz und kartesisches Produkt umfasst. Wie kann die Division zweier Relationen mit diesen Operatoren ausgedrückt werden? Begründen Sie kurz die einzelnen Bestandteile Ihres relationalen Ausdrucks.

Lösungsvorschlag

Seien  $R, S$  Relationen und  $\beta$  die zu  $R$  sowie  $\gamma$  die zu  $S$  dazugehörigen Attributmengen.  $R' := \beta \setminus \gamma$ . Die Division ist dann definiert durch:

$$R \div S := \pi_{R'}(R) - \pi_{R'}((\pi_{R'}(R) \times S) - R)$$

## 66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 4

Gegeben sei das Relationenschema  $R(A, B, C, D, E, F)$  sowie die Menge der zugehörigen funktionalen Abhängigkeiten  $F$ :

$$F = \left\{ \begin{array}{l} \{A, B\} \rightarrow \{C\}, \\ \{A\} \rightarrow \{D\}, \\ \{F\} \rightarrow \{B\}, \\ \{D, E\} \rightarrow \{B\}, \\ \{B\} \rightarrow \{A\}, \end{array} \right\}$$

- (a) Bestimmen Sie sämtliche Schlüsselkandidaten der Relation  $R$  und begründen Sie, warum es keine weiteren Schlüsselkandidaten geben kann.

Lösungsvorschlag

Die Attribute  $E, F$  kommen auf keiner rechten Seite vor.

$$\text{AttrHülle}(F, \{E, F\}) = \{A, B, C, D, E, F\} = R$$

Der Superschlüssel kann nicht weiter minimiert werden:

$$\text{AttrHülle}(F, \{E\}) = \{E\} \neq R$$

$$\text{AttrHülle}(F, \{F\}) = \{A, B, C, D, F\} \neq R$$

Der Schlüsselkandidat ist  $\{E, F\}$

- (b) Ist die gegebene Menge an funktionalen Abhängigkeiten minimal? Fall sie minimal ist begründen Sie diese Eigenschaft ausführlich, anderenfalls minimieren Sie FD schrittweise. Vergessen Sie nicht die einzelnen Schritte entsprechend zu begründen.

**(i) Linksreduktion**

— Führe für jede funktionale Abhängigkeit  $\alpha \rightarrow \beta \in F$  die Linksreduktion durch, überprüfe also für alle  $A \in \alpha$ , ob  $A$  überflüssig ist, d. h. ob  $\beta \subseteq \text{AttrHülle}(F, \alpha - A)$ . —

$$\{A, B\} \rightarrow \{C\}$$

$$C \in \text{AttrHülle}(F, \{A, B \setminus A\}) = \{A, B, C, D\}$$

$$F = \left\{ \begin{array}{l} \{B\} \rightarrow \{C\}, \\ \{A\} \rightarrow \{D\}, \\ \{F\} \rightarrow \{B\}, \\ \{D, E\} \rightarrow \{B\}, \\ \{B\} \rightarrow \{A\}, \end{array} \right\}$$

$$\{D, E\} \rightarrow \{B\}$$

$$B \notin \text{AttrHülle}(F, \{D, E \setminus D\}) = \{E\}$$

$$B \notin \text{AttrHülle}(F, \{D, E \setminus E\}) = \{D\}$$

**(ii) Rechtsreduktion**

— Führe für jede (verbliebene) funktionale Abhängigkeit  $\alpha \rightarrow \beta$  die Rechtsreduktion durch, überprüfe also für alle  $B \in \beta$ , ob  $B \in \text{AttrHülle}(F - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B)), \alpha)$  gilt. In diesem Fall ist  $B$  auf der rechten Seite überflüssig und kann eliminiert werden, d.h.  $\alpha \rightarrow \beta$  wird durch  $\alpha \rightarrow (\beta - B)$  ersetzt. —

**B**

$$B \notin \text{AttrHülle}(F \setminus \{\{F\} \rightarrow \{B\}, \{F\}\}) = \{F\}$$

$$B \notin \text{AttrHülle}(F \setminus \{\{D, E\} \rightarrow \{B\}, \{D, E\}\}) = \{D, E\}$$

$$F = \left\{ \begin{array}{l} \{B\} \rightarrow \{C\}, \\ \{A\} \rightarrow \{D\}, \\ \{F\} \rightarrow \{B\}, \\ \{D, E\} \rightarrow \{B\}, \\ \{B\} \rightarrow \{A\}, \end{array} \right\}$$

**(iii) Löschen leerer Klauseln**

— Entferne die funktionalen Abhängigkeiten der Form  $\alpha \rightarrow \emptyset$ , die im 2. Schritt möglicherweise entstanden sind. \_\_\_\_\_

$\emptyset$  Nichts zu tun

**(iv) Vereinigung**

— Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form  $\alpha \rightarrow \beta_1, \dots, \alpha \rightarrow \beta_n$ , so dass  $\alpha \rightarrow \beta_1 \cup \dots \cup \beta_n$  verbleibt. \_\_\_\_\_

$$F = \left\{ \begin{array}{l} \{A\} \rightarrow \{D\}, \\ \{F\} \rightarrow \{B\}, \\ \{D, E\} \rightarrow \{B\}, \\ \{B\} \rightarrow \{A, C\}, \end{array} \right\}$$

- (c) Überführen Sie falls nötig das Schema in dritte Normalform. Ist die dritte Normalform bereits erfüllt, begründen Sie dies ausführlich.

Lösungsvorschlag

**(i) Kanonische Überdeckung**

— Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden. \_\_\_\_\_

$$F = \left\{ \begin{array}{l} \{A\} \rightarrow \{D\}, \\ \{F\} \rightarrow \{B\}, \\ \{D, E\} \rightarrow \{B\}, \\ \{B\} \rightarrow \{A, C\}, \end{array} \right\}$$

**(ii) Relationsschemata formen**

— Erzeuge für jede funktionale Abhängigkeit  $\alpha \rightarrow \beta \in F_c$  ein Relationenschema  $\mathcal{R}_\alpha := \alpha \cup \beta$ .

$R_1(\underline{A}, D)$

$R_2(\underline{E}, B)$

$R_3(\underline{D}, E, B)$

$R_3(\underline{B}, A, C)$

**(iii) Schlüssel hinzufügen**

— Falls eines der in Schritt 2. erzeugten Schemata  $\mathcal{R}_\alpha$  einen Schlüsselkandidaten von  $\mathcal{R}$  bezüglich  $F_c$  enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten  $\mathcal{K} \subseteq \mathcal{R}$  aus und definiere folgendes zusätzliche Schema:  $\mathcal{R}_\mathcal{K} := \mathcal{K}$  und  $\mathcal{F}_\mathcal{K} := \emptyset$  \_\_\_\_\_

$R_1(\underline{A}, D)$  $R_2(\underline{E}, B)$  $R_3(\underline{D}, \underline{E}, B)$  $R_4(\underline{B}, A, C)$  $R_5(\underline{E}, F)$ (iv) **Entfernung überflüssiger Teilschemata**

— *Eliminiere diejenigen Schemata  $R_\alpha$ , die in einem anderen Relationenschema  $R_{\alpha'}$  enthalten sind, d. h.  $R_\alpha \subseteq R_{\alpha'}$ .*

∅ Nichts zu tun

**66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 5**

Gegeben seien die beiden Relationen Professor und Vorlesung mit folgendem Umfang:  
Relation Professor: 5 Spalten, 164 Datensätze  
Relation Vorlesung: 10 Spalten, 333 Datensätze.

Optimieren Sie auf geeignete Weise folgende SQL-Anweisung möglichst gut und berechnen Sie wie stark sich die Datenmenge durch jede Optimierung reduziert (nehmen Sie für Ihre Berechnung an, dass Herr Mustermann genau zwei Vorlesungen hält). Geben Sie jeweils den Operatorbaum vor und nach Ihren jeweiligen Optimierungen an.

SELECT Titel FROM Professor, Vorlesung WHERE Name = Mustermann' AND PersNr = gelesenVon;

**66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 6**

Begründen oder erläutern Sie Ihre Antworten.

- (a) Erklären Sie kurz den Unterschied zwischen einem Natural-Join und einem Equi-Join.

Lösungsvorschlag

Ein Natural Join ist eine Kombination von zwei Tabellen, in denen Spalten gleichen Namens existieren. Die Werte in diesen Spalten werden sodann auf Übereinstimmungen geprüft (analog Equi-Join). Einige Datenbanksysteme erkennen das Schlüsselwort NATURAL und eliminieren entsprechend automatisch doppelte Spalten.

Während beim Kreuzprodukt keinerlei Anforderungen an die Kombination der Datensätze gestellt werden, führt der Equi-Join eine solche ein: Die Gleichheit von zwei Spalten.

<sup>a</sup>

<sup>a</sup>wiki.selfhtml.org

- (b) Erläutern Sie kurz was man unter einem Theta-Join versteht.

Ein Theta-Join ist eine Verbindung von Relationen bezüglich beliebiger Attribute und mit einem Selektionsprädikat. <sup>a</sup>

<sup>a</sup><https://www.datenbank-grundlagen.de/theta-join.html>

- (c) Was versteht man unter Unionkompatibilität? Nennen Sie drei SQL-Operatoren welche Unionkompatibilität voraussetzen.

Bestimmte Operationen der relationalen Algebra wie Vereinigung, Schnitt und Differenz verlangen Unionkompatibilität. Unionkompatibilität ist eine Eigenschaft des Schemas einer Relation. Zwei Relationen  $R$  und  $S$  sind genau dann union-kompatibel, wenn folgende Bedingungen erfüllt sind:

- (i) Die Relationen  $R$  und  $S$  besitzen dieselbe Stelligkeit  $n$ , d.h. sie haben die selbe Anzahl von Spalten.
- (ii) Für alle Spalten der Relationen gilt, dass die Domäne der  $i$ -ten Spalte der Relation  $R$  mit dem Typ der  $i$ -ten Spalte der Relation  $S$  übereinstimmt ( $0 < i < n$ ).

Die Namen der Attribute spielen dabei keine Rolle. <sup>a</sup>

### SQL-Operatoren mit Unionkompatibilität

- UNION
- INTERSECT
- EXCEPT

<sup>a</sup><https://studylibde.com/doc/1441274/übungstool-für-relationale-algebra>

- (d) Erläutern Sie Backward und Forward Recovery und grenzen Sie diese voneinander ab.
- (e) Erklären Sie das Zwei-Phasen-Freigabe-Protokoll.
- (f) Erläutern Sie Partial Undo / Redo und Global Undo / Redo und deren Bedeutung für die Umsetzung des ACID-Prinzips. Geben Sie zu jeder dieser Konzepte an, ob System-, Programm- oder Gerätefehler damit korrigiert werden können.
- (g) Erklären Sie das WAL-Prinzip (Write ahead logging)!

Das sogenannte write ahead logging (WAL) ist ein Verfahren der Datenbanktechnologie, das zur Gewährleistung der Atomarität und Dauerhaftigkeit von Transaktionen beiträgt. Es besagt, dass Modifikationen vor dem eigentlichen Schreiben (dem Einbringen in die Datenbank) protokolliert werden müssen. Durch das WAL-Prinzip wird ein sogenanntes „update-in-place“ ermöglicht,

Die alte Version eines Datensatzes wird durch die neue Version an gleicher Stelle überschrieben. Das hat vor allem den Vorteil, dass Indexstrukturen bei Änderungsoperationen nicht aktualisiert werden müssen, weil die geänderten Datensätze immer noch an der gleichen Stelle zu finden sind. Die vorherige Protokollierung einer Änderung ist erforderlich, um im Fehlerfall die Wiederholbarkeit der Änderung sicherstellen zu können.

(h) Erklären Sie den Begriff „Datenbankindex“ und nennen Sie zwei häufige Arten.

Lösungsvorschlag

Ein Datenbankindex ist eine von der Datenstruktur getrennte Indexstruktur in einer Datenbank, die die Suche und das Sortieren nach bestimmten Feldern beschleunigt.

### Gruppierte Indizes (Clustered Index)

Bei der Verwendung eines gruppierten Index werden die Datensätze entsprechend der Sortierreihenfolge ihres Index-Schlüssels gespeichert. Wird für eine Tabelle beispielsweise eine Primärschlüssel-Spalte „NR“ angelegt, so stellt diese den Index-Schlüssel dar. Pro Tabelle kann nur ein gruppierter Index erstellt werden. Dieser kann jedoch aus mehreren Spalten zusammengesetzt sein.

### Nicht-gruppierte Indizes (Nonclustered Index)

Besitzt eine Tabelle einen gruppierten Index, so können weitere nicht-gruppierte Indizes angelegt werden. Dabei zeigen die Einträge des Index auf den Speicherbereich des gesamten Datensatzes. Die Verwendung eines nicht-gruppierten Index bietet sich an, wenn regelmäßig nach bestimmten Werten in einer Spalte gesucht wird z.B. dem Namen eines Kunden.<sup>a</sup>

<sup>a</sup><https://www.datenbanken-verstehen.de/datenmodellierung/datenbank-index>

## 66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 7

Gegeben sind folgende Relationen aus einem Verwaltungssystem für die jährlichen Formel-1-Rennen:

Strecke(Strecken\_ID, Streckenname, Land, Länge)

Fahrer(Fahrer\_ID, Fahrername, Nation, Rennstall)

Rennen(Strecken\_ID[Strecke], Jahr, Wetter)

Rennteilnahme(Fahrer\_ID[Fahrer], Strecken\_ID[Rennen], Jahr[Rennen], Rundenbestzeit, Gesamtzeit, disqualifiziert)

FK (Strecken\_ID, Jahr) referenziert Rennen (Strecken\_ID, Jahr)

```
CREATE TABLE Fahrer (
 Fahrer_ID INTEGER PRIMARY KEY,
 Fahrername VARCHAR(100) NOT NULL,
 Nation VARCHAR(100) NOT NULL,
 Rennstall VARCHAR(100) NOT NULL
);

CREATE TABLE Strecke (
 Strecken_ID INTEGER PRIMARY KEY,
 Streckenname VARCHAR(100) NOT NULL,
 Land VARCHAR(100) NOT NULL,
 Länge NUMERIC(5,3) NOT NULL
);

CREATE TABLE Rennen (
 Strecken_ID INTEGER REFERENCES Strecke(Strecken_ID),
 Jahr INTEGER NOT NULL,
 Wetter VARCHAR(10) NOT NULL,
 PRIMARY KEY (Strecken_ID, Jahr)
);

CREATE TABLE Rennteilnahme (
 Fahrer_ID INTEGER REFERENCES Fahrer(Fahrer_ID),
 Strecken_ID INTEGER REFERENCES Strecke(Strecken_ID),
 Jahr INTEGER NOT NULL,
 Rundenbestzeit NUMERIC(5,3) NOT NULL,
 Gesamtzeit NUMERIC(5,3) NOT NULL,
 disqualifiziert BOOLEAN NOT NULL,
 PRIMARY KEY (Fahrer_ID, Strecken_ID, Jahr)
);

INSERT INTO Fahrer VALUES
 (1, 'Kimi Räikkönen', 'Finnland', 'Alfa Romeo'),
 (2, 'Rubens Barrichello', 'Brasilien', 'Brawn'),
 (3, 'Fernando Alonso', 'Spanien', 'Ferrari'),
 (4, 'Michael Schumacher', 'Deutschland', 'Ferrari'),
 (5, 'Jenson Button', 'Vereinigtes Königreich Großbritannien', 'McLaren'),
 (6, 'Felipe Massa', 'Brasilien', 'Ferrari'),
 (7, 'Lewis Hamilton', 'Vereinigtes Königreich Großbritannien', 'Williams'),
 (8, 'Riccardo Patrese', 'Italien', 'Williams'),
 (9, 'Sebastian Vettel', 'Deutschland', 'Ferrari'),
 (10, 'Jarno Trulli', 'Italien', 'Toyota');

INSERT INTO Strecke VALUES
 (1, 'Autodromo Nazionale Monza', 'Italien', 5.793),
 (2, 'Circuit de Monaco', 'Monaco', 3.340),
 (3, 'Silverstone Circuit', 'Vereinigtes Königreich', 5.891),
 (4, 'Circuit de Spa-Francorchamps', 'Belgien', 7.004),
 (5, 'Circuit Gilles-Villeneuve', 'Kanada', 4.361),
 (6, 'Nürburgring', 'Deutschland', 5.148),
 (7, 'Hockenheimring', 'Deutschland', 4.574),
 (8, 'Interlagos', 'Brasilien', 4.309),
 (9, 'Hungaroring', 'Ungarn', 4.381),
 (10, 'Red Bull Ring', 'Österreich', 5.942),
 (11, 'Abu Dhabi', 'Abu Dhabi', 5.554);
```



```
INSERT INTO Rennen VALUES
```

```
(11, 2011, 'sonnig'),
(10, 2006, 'sonnig'),
(9, 2007, 'regnerisch'),
(8, 2008, 'regnerisch'),
(7, 2009, 'sonnig'),
(6, 2010, 'regnerisch'),
(5, 2011, 'sonnig'),
(4, 2012, 'sonnig'),
(3, 2013, 'sonnig'),
(2, 2014, 'regnerisch'),
(1, 2015, 'regnerisch');
```

```
INSERT INTO Rennteilnahme VALUES
```

```
(1, 11, 2011, 2.001, 90.001, FALSE),
(2, 11, 2011, 2.002, 90.002, FALSE),
(3, 11, 2011, 2.003, 90.003, FALSE),
(4, 11, 2011, 2.004, 89.999, FALSE),
(5, 11, 2011, 2.005, 90.005, FALSE),
(6, 11, 2011, 2.005, 99.009, FALSE),
(4, 10, 2006, 2.782, 90.005, TRUE),
(3, 10, 2006, 2.298, 90.005, TRUE),
(3, 9, 2009, 2.253, 90.005, TRUE),
(2, 10, 2006, 2.005, 90.005, TRUE),
(2, 9, 2009, 3.298, 90.342, TRUE),
(2, 8, 2008, 4.782, 78.005, TRUE);
```

Der Einfachheit halber wird angenommen, dass Fahrer den Rennstall nicht wechseln können. Das Attribut „disqualifiziert“ kann die Ausprägungen „ja“ und „nein“ haben. Formulieren Sie folgende Abfragen in SQL. Vermeiden Sie nach Möglichkeit übermäßige Nutzung von Joins und Views.

- (a) Geben Sie für jeden Fahrer seine ID sowie die Anzahl seiner Disqualifikationen in den Jahren 2005 bis 2017 aus. Ordnen Sie die Ausgabe absteigend nach der Anzahl der Disqualifikationen.

Lösungsvorschlag

```
SELECT Fahrer_ID, COUNT(disqualifiziert) as anzahl_disqualifikationen FROM
 ↳ Rennteilnahme
WHERE disqualifiziert = TRUE
GROUP BY Fahrer_ID, disqualifiziert
ORDER BY anzahl_disqualifikationen DESC;
```

- (b) Gesucht sind alle Länder, aus denen noch nie ein Fahrer disqualifiziert wurde.

Lösungsvorschlag

```
SELECT Nation FROM Fahrer GROUP BY Nation
EXCEPT
SELECT f.Nation FROM Fahrer f, Rennteilnahme t
WHERE f.Fahrer_ID = t.Fahrer_ID AND t.disqualifiziert = TRUE
```

```
GROUP BY f.Nation;
```

```
ALTER TABLE
TRIGGER
```

- (c) Gesucht sind die ersten fünf Plätze des Rennens von 2011 in „Abu Dhabi“ (Streckenname). Die Ausgabe soll nach der Platzierung absteigend erfolgen. Geben Sie Fahrer\_ID, Fahrername, Nation und Rennstall mit aus.

Lösungsvorschlag

Mit LIMIT

```
SELECT f.Fahrer_ID, f.Fahrername, f.Nation, f.Rennstall
FROM Fahrer f, Rennteilnahme t, Strecke s
WHERE
 f.Fahrer_ID = t.Fahrer_ID AND
 s.Strecken_ID = t.Strecken_ID AND
 s.Streckenname = 'Abu Dhabi' AND
 t.Jahr = 2011
ORDER BY t.Gesamtzeit ASC LIMIT 5;
```

Als Top-N-Query:

```
CREATE VIEW Rennen_Abu_Dhabi AS
SELECT f.Fahrer_ID, f.Fahrername, f.Nation, f.Rennstall, t.Gesamtzeit
FROM Fahrer f, Rennteilnahme t, Strecke s
WHERE
 f.Fahrer_ID = t.Fahrer_ID AND
 s.Strecken_ID = t.Strecken_ID AND
 s.Streckenname = 'Abu Dhabi' AND
 t.Jahr = 2011
ORDER BY t.Gesamtzeit ASC;

SELECT a.Fahrer_ID, a.Fahrername, a.Nation, a.Rennstall
FROM Rennen_Abu_Dhabi a, Rennen_Abu_Dhabi b
WHERE
 a.Gesamtzeit >= b.Gesamtzeit
GROUP BY a.Fahrer_ID, a.Fahrername, a.Nation, a.Rennstall, a.Gesamtzeit
HAVING COUNT(*) <= 5
ORDER BY a.Gesamtzeit;
```

- (d) Führen Sie eine neue Spalte Gehalt in die Tabelle Fahrer ein. Da sich die Prämien für die Fahrer nach einem Rennstallwechsel ändern, soll ein Trigger geschrieben werden, mit dem das Gehalt des betreffenden Fahrers um 10% angehoben wird.

**Lösung für PostgreSQL:**

```
ALTER TABLE Fahrer ADD Gehalt numeric(12,2);

UPDATE Fahrer SET Gehalt = 1000000 WHERE Fahrer_ID = 1;
UPDATE Fahrer SET Gehalt = 2000000 WHERE Fahrer_ID = 2;
UPDATE Fahrer SET Gehalt = 3000000 WHERE Fahrer_ID = 3;
UPDATE Fahrer SET Gehalt = 4000000 WHERE Fahrer_ID = 4;
UPDATE Fahrer SET Gehalt = 5000000 WHERE Fahrer_ID = 5;
UPDATE Fahrer SET Gehalt = 6000000 WHERE Fahrer_ID = 6;
UPDATE Fahrer SET Gehalt = 7000000 WHERE Fahrer_ID = 7;
```

```

UPDATE Fahrer SET Gehalt = 8000000 WHERE Fahrer_ID = 8;
UPDATE Fahrer SET Gehalt = 9000000 WHERE Fahrer_ID = 9;
UPDATE Fahrer SET Gehalt = 10000000 WHERE Fahrer_ID = 10;

CREATE FUNCTION trigger_function()
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS $$
BEGIN
 UPDATE Fahrer
 SET gehalt = gehalt * 1.1
 WHERE Fahrer_ID = NEW.Fahrer_ID;
 RETURN NEW;
END;
$$;

CREATE TRIGGER mehr_gehalt
AFTER UPDATE OF Rennstall ON Fahrer
FOR EACH ROW EXECUTE PROCEDURE trigger_function();

-- Test:
SELECT * FROM FAHRER WHERE Fahrer_ID = 1;
UPDATE Fahrer SET Rennstall = 'Red Bull' WHERE Fahrer_ID = 1;
SELECT * FROM FAHRER WHERE Fahrer_ID = 1;

```

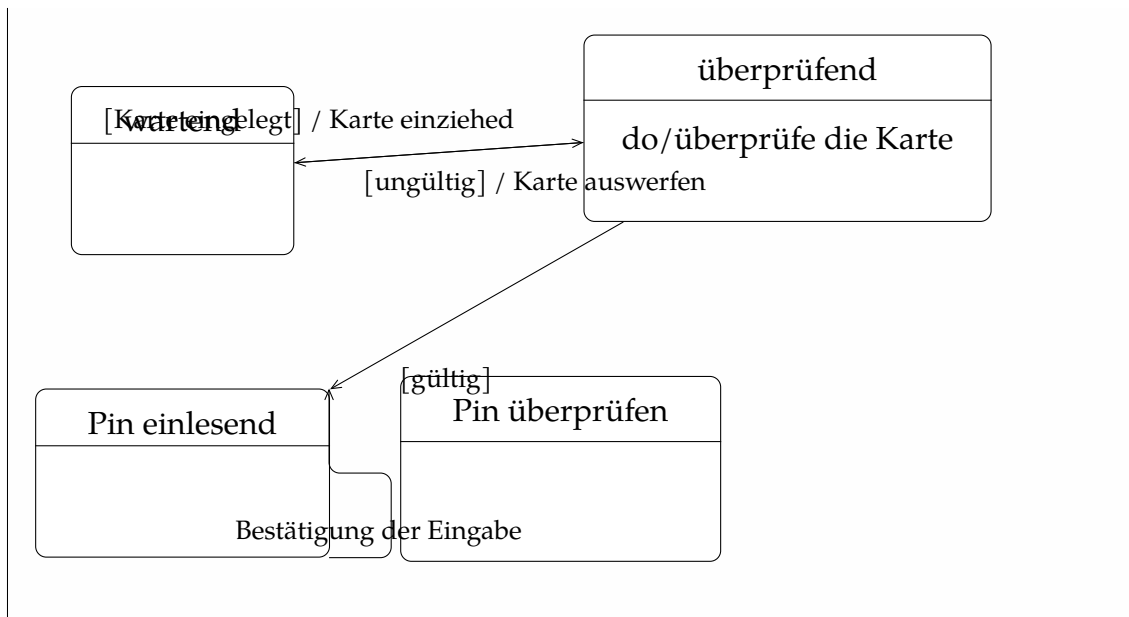
## 66116 / 2020 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1

### (a) Basisfunktion

Erstellen Sie ein Zustandsdiagramm für einen Bankautomat, welcher den im Folgenden beschriebenen Authentifizierungsvorgang von Bankkunden realisiert. Modellieren Sie dazu soweit nötig sowohl Zustände und Transitionsbedingungen als auch die Aktionen der Zustände.

Der Bankautomat startet im Grundzustand und wartet auf das Einlegen einer Bankkarte. Wird eine Karte eingelegt, wird diese eingezogen und der Automat startet die Überprüfung der Bankkarte. Ist die Karte ungültig, wird die Karte ausgeworfen und der Automat wechselt in den Grundzustand. Ist die Karte gültig, kann die vierstellige PIN eingelesen werden. Nach der Bestätigung der Eingabe wird diese überprüft. Ist die PIN gültig, so stoppt der Automat und zeigt eine erfolgreiche Authentifizierung an. Ist die PIN ungültig, zeigt der Automat einen Fehler an und erlaubt eine erneute Eingabe der PIN.

Lösungsvorschlag



## (b) Erweiterung

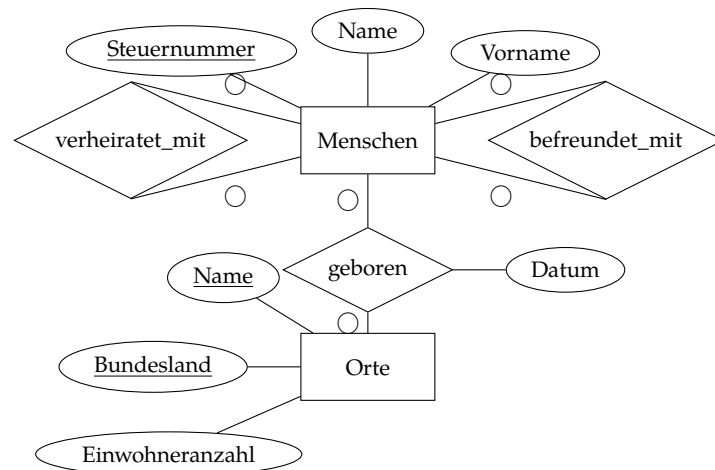
Der Bankautomaten aus Aufgabe a) soll nun so verändert werden, dass ein Bankkunde nach dem ersten fehlerhaften Eingeben der PIN die PIN erneut eingeben muss. Bei erneuter Falscheingabe, wird eine dritte Eingabe möglich. Bei der dritten Falscheingabe der PIN wird die Karte vom Automaten eingezogen und der Automat geht wieder in den Ausgangszustand über.

Hinweis: Für diese Aufgabe dürfen Sie Ihr Zustandsdiagramm aus a) weiter verwenden, wenn Sie eindeutig, z. B. durch den Einsatz von Farben kennzeichnen, was nur zur Aufgabe a) gehört und was Abänderungen des Zustandsdiagramms aus a) sind. Sie können, falls Sie einen neuen Automaten zeichnen, Zustände und Übergänge, die inhaltsgleich zur Lösung des Aufgabenteils a) sind mit einem "W" markieren, statt sie zu beschriften. In diesem Fall wird der Text aus der Lösung zu Aufgabenteil a) an dieser Stelle wiederholt gedacht.

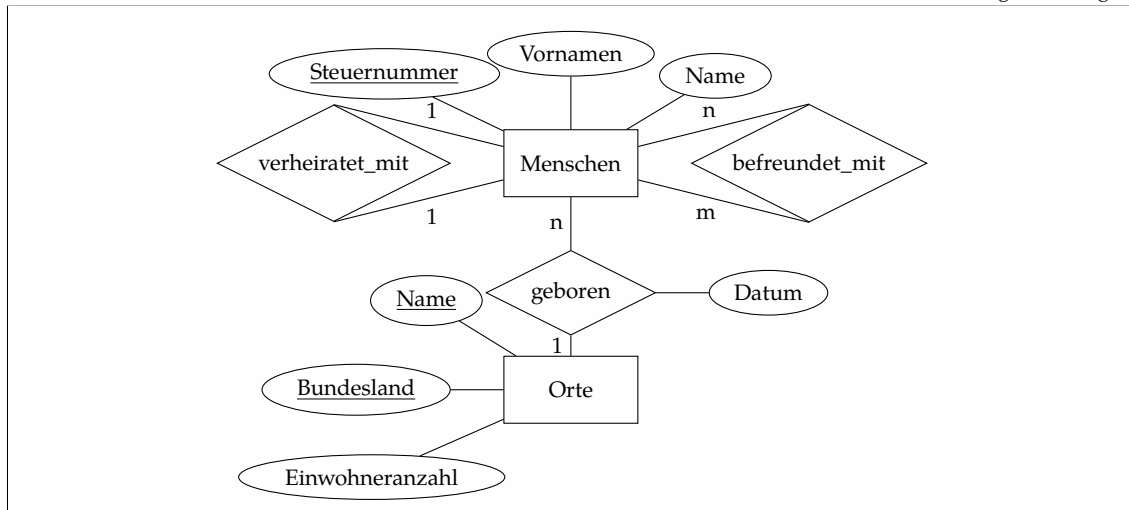
## 66116 / 2020 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 1

Gegeben sei folgendes ER-Diagramm:

- (a) Übernehmen Sie das ER-Diagramm auf Ihre Bearbeitung und ergänzen Sie die Funktionalitätsangaben im Diagramm.



Lösungsvorschlag



- (b) Übersetzen Sie das ER-Diagramm in ein relationales Schema. - Datentypen müssen nicht angegeben werden.

Lösungsvorschlag

```

Menschen(Steuernummer, Name, Vorname)

Orte(Name, Bundesland, Einwohneranzahl)

verheiratet_mit(Mensch[Menschen], Ehepartner[Menschen])

befreundet_mit(Mensch[Menschen], Freund[Menschen])

geboren(Datum, Steuernummer, Geburtsort[Orte], Geburtsbundesland[Orte])

```

- (c) Verfeinern Sie das Schema aus Teilaufgabe b) indem Sie die Relationen zusammenfassen.

```

Menschen(Steuernummer, Name, Vorname, Ehepartner[Menschen], Geburtsdatum, Geburtsort[Orte], Geburtsbundesland[Orte])

Orte(Name, Bundesland, Einwohneranzahl)

befreundet_mit(Mensch[Menschen], Freund[Menschen])

```

(d) Geben Sie sinnvolle SQL Datentypen für Ihr verfeinertes Schema an.

```

CREATE TABLE Menschen (
 Steuernummer BIGINT PRIMARY KEY,
 Name VARCHAR(30),
 Vorname VARCHAR(30),
 Ehepartner BIGINT REFERENCES Steuernummer,
 Geburtsdatum DATE,
 Geburtsort VARCHAR(30) REFERENCES Orte(Name),
 Geburtsbundesland VARCHAR(30) REFERENCES Orte(Bundesland)
);

CREATE TABLE Orte (
 Name VARCHAR(30),
 Bundesland VARCHAR(30),
 Einwohneranzahl INTEGER,
 PRIMARY KEY (Name, Bundesland)
);

CREATE TABLE befreundet_mit (
 Mensch BIGINT REFERENCES Mensch(Steuernummer),
 Freund BIGINT REFERENCES Mensch(Steuernummer),
 PRIMARY KEY (Mensch, Freund)
);

```

## 66116 / 2020 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2

### Relation „Sekretäre“

| PersNr | Name    | Boss | Raum |
|--------|---------|------|------|
| 4000   | Freud   | 2125 | 225  |
| 4000   |         |      | 225  |
| 4020   | Röntgen | 2163 | 6    |
| 4020   | Röntgen |      | 26   |
| 4030   | Galileo | 2127 |      |
|        | Freud   | 2137 | 80   |

Gegeben sei oben stehenden (lückenhafte) Relationenausprägung **Sekretäre** sowie die folgenden funktionalen Abhängigkeiten:

$$FA = \left\{ \begin{array}{l} \{ PersNr \} \rightarrow \{ Name \}, \\ \{ PersNr, Boss \} \rightarrow \{ Raum \}, \end{array} \right\}$$

Geben Sie für alle leeren Zellen Werte an, so dass keine funktionalen Abhängigkeiten verletzt werden. (Hinweis: Es gibt mehrere richtige Antworten.)

Lösungsvorschlag

| PersNr | Name         | Boss              | Raum            |
|--------|--------------|-------------------|-----------------|
| 4000   | Freud        | 2125              | 225             |
| 4000   | <i>Freud</i> | 2143 <sup>a</sup> | 225             |
| 4020   | Röntgen      | 2163              | 6               |
| 4020   | Röntgen      | 2163 <sup>b</sup> | 26              |
| 4030   | Galileo      | 2127              | 27 <sup>c</sup> |
| 4000   | Freud        | 2137              | 80              |

<sup>a</sup>Muss eine andere Boss-ID sein, sonst gäbe es zwei identische Zeilen.

<sup>b</sup>anderer Boss

<sup>c</sup>anderer Raum

### 66116 / 2020 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 3

Gegeben sei ein Universitätsschema.

- (a) Finden Sie alle Studierenden, die keine Vorlesung hören. Formulieren Sie die Anfrage im Tupelkalkül.

Lösungsvorschlag

$$\{s \in \text{Studierende} \wedge h \in \text{hören} \mid \neg \exists s. \text{MatrNr} = h. \text{MatrNr}\}$$

- (b) Geben Sie einen Ausdruck an, der die Relation  $\neg\text{hören}$  erzeugt. Diese enthält für jeden Studierenden und jede Vorlesung, die der Studierende **nicht** hört, einen Eintrag mit Matrikelnummer und Vorlesungsnummer. Formulieren Sie die Anfrage in **relationaler Algebra**.

Lösungsvorschlag

$$\rho_{\neg\text{hören}} \left( \left( \pi_{\text{MatrNr}}(\text{Studierende}) \times \pi_{\text{VorlNr}}(\text{Vorlesungen}) \right) - \text{hören} \right)$$

- (c) Finden Sie alle Studierenden, die **keine** Vorlesung hören. Formulieren Sie die Anfrage in **relationaler Algebra**.



$$\pi_{\text{MatrNr}}(\text{Studierende}) - \pi_{\text{MatrNr}}(\text{hören})$$

## 66116 / 2020 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 4

Gegeben sei die Relation

$$R(A, B, C, D, E, F)$$

mit den FDs

$$\text{FA} = \left\{ \begin{array}{l} \{A\} \rightarrow \{B, C, F\}, \\ \{B\} \rightarrow \{A, B, F\}, \\ \{C, D\} \rightarrow \{E, F\}, \end{array} \right\}$$

(a) Geben Sie alle Kandidatenschlüssel an.

Lösungsvorschlag

- $\{A, D\}$
- $\{B, D\}$

(b) Überführen Sie die Relation mittels Synthesealgorithmus in die 3. NF. Geben Sie alle Relationen in der 3. NF an und **unterstreichen Sie in jeder einen Kandidatenschlüssel**. — Falls Sie Zwischenschritte notieren, machen Sie das Endergebnis **klar kenntlich**.

Lösungsvorschlag

### (i) Kanonische Überdeckung

— Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden. —

#### i. Linksreduktion

— Führe für jede funktionale Anhängigkeit  $\alpha \rightarrow \beta \in F$  die Linksreduktion durch, überprüfe also für alle  $A \in \alpha$ , ob  $A$  überflüssig ist, d. h. ob  $\beta \subseteq \text{AttrHülle}(F, \alpha - A)$ . —

$$\{C, D\} \rightarrow \{E, F\}$$

$$\{E, F\} \notin \text{AttrHülle}(F, \{C, D \setminus D\}) = \{C\}$$

$$\{E, F\} \notin \text{AttrHülle}(F, \{C, D \setminus C\}) = \{D\}$$

$$\text{FA} = \left\{ \begin{array}{l} \{A\} \rightarrow \{B, C, F\}, \\ \{B\} \rightarrow \{A, B, F\}, \\ \{C, D\} \rightarrow \{E, F\}, \end{array} \right\}$$

**ii. Rechtsreduktion**

— Führe für jede (verbliebene) funktionale Abhängigkeit  $\alpha \rightarrow \beta$  die Rechtsreduktion durch, überprüfe also für alle  $B \in \beta$ , ob  $B \in \text{AttrHülle}(F - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B)), \alpha)$  gilt. In diesem Fall ist  $B$  auf der rechten Seite überflüssig und kann eliminiert werden,  $\delta\alpha \rightarrow \beta$  wird durch  $\alpha \rightarrow (\beta - B)$  ersetzt. —

**F**

$$F \in \text{AttrHülle}(F \setminus \{A\} \rightarrow \{B, C, F\} \cup \{A\} \rightarrow \{B, C\}, \{A\}) = \{A, B, C, F\}$$

$$FA = \left\{ \begin{array}{l} \{A\} \rightarrow \{B, C\}, \\ \{B\} \rightarrow \{A, B, F\}, \\ \{C, D\} \rightarrow \{E, F\}, \end{array} \right\}$$

$$F \notin \text{AttrHülle}(F \setminus \{B\} \rightarrow \{A, B, F\} \cup \{B\} \rightarrow \{A, B\}, \{B\}) = \{A, B, C\}$$

$$F \notin \text{AttrHülle}(F \setminus \{C, D\} \rightarrow \{E, F\} \cup \{C, D\} \rightarrow \{E\}, \{C, D\}) = \{C, D, E\}$$

**B**

$$B \notin \text{AttrHülle}(F \setminus \{A\} \rightarrow \{B, C\} \cup \{A\} \rightarrow \{C\}, \{A\}) = \{A, C\}$$

$$B \in \text{AttrHülle}(F \setminus \{B\} \rightarrow \{A, B, F\} \cup \{B\} \rightarrow \{A, F\}, \{B\}) = \{A, B, F\}$$

$$FA = \left\{ \begin{array}{l} \{A\} \rightarrow \{B, C\}, \\ \{B\} \rightarrow \{A, F\}, \\ \{C, D\} \rightarrow \{E, F\}, \end{array} \right\}$$

**iii. Löschen leerer Klauseln**

— Entferne die funktionalen Abhängigkeiten der Form  $\alpha \rightarrow \emptyset$ , die im 2. Schritt möglicherweise entstanden sind. —

$\emptyset$  Nichts zu tun

**iv. Vereinigung**

— Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form  $\alpha \rightarrow \beta_1, \dots, \alpha \rightarrow \beta_n$ , so dass  $\alpha \rightarrow \beta_1 \cup \dots \cup \beta_n$  verbleibt. —

$\emptyset$  Nichts zu tun

**(ii) Relationsschemata formen**

— Erzeuge für jede funktionale Abhängigkeit  $\alpha \rightarrow \beta \in F_c$  ein Relationenschema  $\mathcal{R}_\alpha := \alpha \cup \beta$ .

$$R_1(\underline{A}, \underline{B}, C)$$

$$R_2(\underline{A}, \underline{B}, F)$$

$$R_3(\underline{C}, \underline{D}, E, F)$$

(iii) **Schlüssel hinzufügen**

— Falls eines der in Schritt 2. erzeugten Schemata  $R_\alpha$  einen Schlüsselkandidaten von  $\mathcal{R}$  bezüglich  $F_c$  enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten  $\mathcal{K} \subseteq \mathcal{R}$  aus und definiere folgendes zusätzliche Schema:  $\mathcal{R}_\mathcal{K} := \mathcal{K}$  und  $\mathcal{F}_\mathcal{K} := \emptyset$  —

$R_1(\underline{A}, B, C)$

$R_2(\underline{A}, B, F)$

$R_3(\underline{C}, D, E, F)$

$R_4(\underline{A}, D)$

(iv) **Entfernung überflüssiger Teilschemata**

— Eliminiere diejenigen Schemata  $R_\alpha$ , die in einem anderen Relationenschema  $R_{\alpha'}$  enthalten sind, d. h.  $R_\alpha \subseteq R_{\alpha'}$ . —

$\emptyset$  Nichts zu tun

**66116 / 2020 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 5**

Gegeben sei die Relation  $R(A, B, C)$

- (a) Schreiben Sie eine SQL-Anfrage, mit der sich zeigen lässt, ob das Paar  $A, B$  ein Superschlüssel der Relation  $R$  ist. Beschreiben Sie ggf. textuell - falls nicht eindeutig ersichtlich - wie das Ergebnis Ihrer Anfrage interpretiert werden muss, um zu erkennen ob  $A, B$  ein Superschlüssel ist.

Lösungsvorschlag

Diese Anfrage darf keine Ergebnisse liefern, dann ist das Paar  $A, B$  ein Superschlüssel.

```
SELECT *
FROM R
GROUP BY A, B
HAVING COUNT(*) > 1;
```

- (b) Erläutern Sie den Unterschied zwischen einem Superschlüssel und einem Kandidatenschlüssel. Tipp: Was muss gelten, damit  $A, B$  ein Kandidatenschlüssel ist und nicht nur ein Superschlüssel?

Lösungsvorschlag

Ein Superschlüssel ist ein Attribut oder eine Attributkombination, von der *alle Attribute* einer Relation funktional *abhängen*.

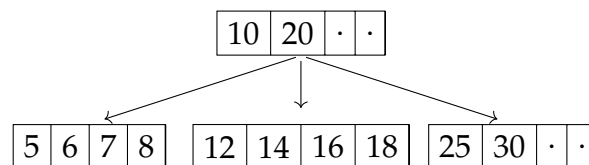
Ein Kandidatenschlüssel ist ein *minimaler* Superschlüssel. Keine Teilmenge dieses Superschlüssels ist ebenfalls Superschlüssels.

- (c) Sei  $A, B$  der Kandidatenschlüssel für die Relation  $R$ . Geben Sie eine minimale Ausprägung der Relation  $R$  an, die diese Eigenschaft erfüllt.

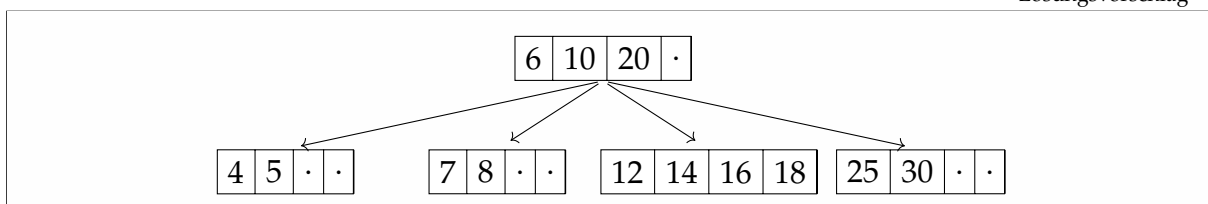
| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 1 | 4 |
| 1 | 1 | 5 |
| 2 | 2 | 5 |

## 66116 / 2020 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 6

Fügen Sie die Zahl 4 in den folgenden B-Baum ein.



Zeichnen Sie den vollständigen, resultierenden Baum.



## 66116 / 2020 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 7

Gegeben sei die Relation *Zehnkampf*, welche die Ergebnisse eines Zehnkampfwettkampfes verwaltet. Eine beispielhafte Ausprägung ist in nachfolgender Tabelle gegeben.

**Hinweise:** Jeder Athlet kann in jeder Disziplin maximal ein Ergebnis erzielen. Außerdem können Sie davon ausgehen, dass jeder Name eindeutig ist.

| Name  | Disziplin  | Leistung | Einheit    | Punkte |
|-------|------------|----------|------------|--------|
| John  | 100m       | 10.21    | Sekunden   | 845    |
| Peter | Hochsprung | 213      | Zentimeter | 812    |
| Peter | 100m       | 10.10    | Sekunden   | 920    |
| Hans  | 100m       | 10.21    | Sekunden   | 845    |
| Hans  | 400m       | 44.12    | Sekunden   | 910    |
| ...   | ...        | ...      | ...        | ...    |

```

CREATE TABLE Zehnkampf (
 Name VARCHAR(30),
 Disziplin VARCHAR(30),
 Leistung FLOAT,
 Einheit VARCHAR(30),
 Punkte INTEGER,
 PRIMARY KEY(Name, Disziplin, Leistung)
);

INSERT INTO Zehnkampf VALUES
('John', '100m', 10.21, 'Sekunden', 845),
('Peter', 'Hochsprung', 213, 'Zentimeter', 812),
('Peter', '100m', 10.10, 'Sekunden', 920),
('Hans', '100m', 10.21, 'Sekunden', 845),
('Hans', '400m', 44.12, 'Sekunden', 910);

```

- (a) Bestimmen Sie alle funktionale Abhängigkeiten, die **sinnvollerweise** in der Relation Zehnkampf gelten.

Lösungsvorschlag

$$\text{FA} = \left\{ \begin{array}{l} \{ \text{Disziplin} \} \rightarrow \{ \text{Einheit} \}, \\ \{ \text{Disziplin}, \text{Leistung} \} \rightarrow \{ \text{Punkte} \}, \\ \{ \text{Name}, \text{Disziplin} \} \rightarrow \{ \text{Leistung} \}, \end{array} \right\}$$

- (b) Normalisieren Sie die Relation Zehnkampf unter Beachtung der von Ihnen identifizierten funktionalen Abhängigkeiten. Unterstreichen Sie alle Schlüssel des resultierenden Schemas.

Lösungsvorschlag

$$\begin{aligned}
 R_1 &: \{ \{ \underline{\text{Disziplin}}, \text{Einheit} \} \} \\
 R_2 &: \{ \{ \underline{\text{Disziplin}}, \underline{\text{Leistung}}, \text{Punkte} \} \} \\
 R_3 &: \{ \{ \underline{\text{Name}}, \underline{\text{Disziplin}}, \text{Leistung} \} \}
 \end{aligned}$$

- (c) Bestimmen Sie in SQL den Athleten (oder bei Punktgleichheit, die Athleten), der in der Summe am meisten Punkte in allen Disziplinen erzielt hat. Benutzen Sie dazu die noch nicht normalisierte Ausgangsrelation *Zehnkampf*.

Lösungsvorschlag

```

CREATE VIEW GesamtPunkte AS
 SELECT Name, SUM(Punkte) AS Punkte
 FROM Zehnkampf
 GROUP BY Name;

SELECT g1.Name, g1.Punkte, COUNT(*) AS Rang
FROM GesamtPunkte g1, GesamtPunkte g2
WHERE g1.Punkte <= g2.Punkte

```

```
GROUP BY g1.Name, g1.Punkte
HAVING COUNT(*) = 1;
```

SQL

```
name | punkte | rang
-----+-----+-----
Hans | 1755 | 1
(1 row)
```

## 66116 / 2020 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 8

Gegeben sei das Universitätsschema. Formulieren Sie folgende Anfragen in SQL-92:

```
CREATE TABLE Studierende (
 MatrNr INTEGER PRIMARY KEY,
 Name VARCHAR(15),
 Semester INTEGER
);
```

```
CREATE TABLE Professoren (
 PersNr INTEGER PRIMARY KEY,
 Name VARCHAR(30),
 Rang VARCHAR(30),
 Raum INTEGER
);
```

```
CREATE TABLE Assistenten (
 PersNr INTEGER PRIMARY KEY,
 Name VARCHAR(20),
 Fachgebiet VARCHAR(30),
 Boss INTEGER,
 FOREIGN KEY (Boss) REFERENCES Professoren(PersNr)
);
```

```
CREATE TABLE Vorlesungen (
 VorlNr INTEGER PRIMARY KEY,
 Titel VARCHAR(30),
 SWS INTEGER,
 gelesenVon INTEGER,
 FOREIGN KEY (gelesenVon) REFERENCES Professoren(PersNr)
);
```

```
CREATE TABLE hören (
 MatrNr INTEGER,
 VorlNr INTEGER,
 PRIMARY KEY(MatrNr, VorlNr),
 FOREIGN KEY (MatrNr) REFERENCES Studierende(MatrNr),
 FOREIGN KEY (VorlNr) REFERENCES Vorlesungen(VorlNr)
);
```

```
CREATE TABLE prüfen (
 MatrNr INTEGER,
 VorlNr INTEGER,
 PersNr INTEGER,
 Note INTEGER,
```

```
PRIMARY KEY(MatrnNr, VorlNr, PersNr),
FOREIGN KEY (MatrnNr) REFERENCES Studierende(MatrnNr),
FOREIGN KEY (VorlNr) REFERENCES Vorlesungen(VorlNr),
FOREIGN KEY (PersNr) REFERENCES Professoren(PersNr)
);

CREATE TABLE voraussetzen (
 Vorgänger INTEGER,
 Nachfolger INTEGER,
 PRIMARY KEY(Vorgänger, Nachfolger),
 FOREIGN KEY (Vorgänger) REFERENCES Vorlesungen(VorlNr),
 FOREIGN KEY (Nachfolger) REFERENCES Vorlesungen(VorlNr)
);

INSERT INTO Studierende
 (MatrnNr, Name, Semester)
VALUES
 (24002, 'Xenokrates', 18),
 (25403, 'Jonas', 12),
 (26120, 'Fichte', 10),
 (26830, 'Aristoxenos', 8),
 (27550, 'Schopenhauer', 6),
 (28106, 'Carnap', 3),
 (29120, 'Theophrastos', 2),
 (29555, 'Feuerbach', 2);

INSERT INTO Professoren
 (PersNr, Name, Rang, Raum)
VALUES
 (2125, 'Sokrates', 'C4', 226),
 (2126, 'Russel', 'C4', 226),
 (2127, 'Kopernikus', 'C3', 226),
 (2133, 'Popper', 'C3', 226),
 (2134, 'Augustinus', 'C3', 226),
 (2136, 'Curie', 'C4', 226),
 (2137, 'Kant', 'C4', 226);

INSERT INTO Assistenten
 (PersNr, Name, Fachgebiet, Boss)
VALUES
 (3002, 'Platon', 'Ideenlehre', 2125),
 (3003, 'Aristoteles', 'Syllogistik', 2125),
 (3004, 'Wittgenstein', 'Sprachtheorie', 2126),
 (3005, 'Rhetikus', 'Planetenbewegung', 2127),
 (3006, 'Newton', 'Kaplorsche Gesetze', 2127),
 (3007, 'Spinoza', 'Gott und Natur', 2134);

INSERT INTO Vorlesungen
 (VorlNr, Titel, SWS, gelesenVon)
VALUES
 (4052, 'Logik', 4, 2125),
 (4630, 'Die 3 Kritiken', 4, 2137),
 (5001, 'Grundzüge', 4, 2137),
 (5022, 'Glaube und Wissen', 2, 2134),
 (5041, 'Ethik', 4, 2125),
```

```
(5043, 'Erkenntnistheorie', 3, 2126),
(5049, 'Mäeutik', 2, 2125),
(5052, 'Wissenschaftstheorie', 3, 2126),
(5216, 'Bioethik', 2, 2126),
(5259, 'Der Wiener Kreis', 2, 2133);
```

```
INSERT INTO hören
(MatrnNr, VorlNr)
```

```
VALUES
(25403, 5022),
(26120, 5001),
(27550, 4052),
(27550, 5001),
(28106, 5041),
(28106, 5052),
(28106, 5216),
(28106, 5259),
(29120, 5001),
(29120, 5041),
(29120, 5049),
(29555, 5001),
(29555, 5022),
(28106, 5001),
(28106, 5022);
```

```
INSERT INTO prüfen
(MatrnNr, VorlNr, PersNr, Note)
```

```
VALUES
(28106, 5001, 2126, 1),
(25403, 5041, 2125, 2),
(27550, 4630, 2137, 2),
(25403, 4630, 2137, 5);
```

```
INSERT INTO voraussetzen
(Vorgänger, Nachfolger)
```

```
VALUES
(5001, 5041),
(5001, 5043),
(5001, 5049),
(5041, 5216),
(5043, 5052),
(5041, 5052),
(5052, 5259);
```

- (a) Welche Vorlesungen liest der Boss des Assistenten *Platon* (nur Vorlesungsnummer und Titel ausgeben)?

Lösungsvorschlag

```
SELECT v.VorlNr, v.Titel
FROM Vorlesungen v, Assistenten a
WHERE a.Boss = v.gelesenVon AND a.Name = 'Platon';

vorlnr | titel
-----+-----
```



```

4052 | Logik
5041 | Ethik
5049 | Mäeutik
(3 rows)

```

- (b) Welche Studierende haben sich schon in mindestens einer direkten Voraussetzung von *Wissenschaftstheorie* prüfen lassen?

Lösungsvorschlag

Wissenschaftstheorie (5052) → Erkenntnistheorie (5043) Ethik (5041) → Jonas (25403)

```

SELECT s.Name
FROM Vorlesungen l, voraussetzen a, prüfen p, Studierende s
WHERE
 l.Titel = 'Wissenschaftstheorie' AND
 l.VorlNr = a.Nachfolger AND
 a.Vorgänger = p.VorlNr AND
 p.MatrNr = s.MatrNr;

name

Jonas
(1 row)

```

- (c) Wie viele Studierende hören *Ethik*?

Lösungsvorschlag

```

SELECT COUNT(*)
FROM Vorlesungen v, hören h
WHERE
 v.Titel = 'Ethik' AND
 v.VorlNr = h.VorlNr;

count

2
(1 row)

```

- (d) Welche Studierende sind im gleichen Semester? — Geben Sie Paare von Studierenden aus.

Achten Sie darauf, dass ein/e Studierende/r mit sich selbst kein Paar bildet. — Achten Sie auch darauf, dass kein Paar doppelt ausgegeben wird: wenn das Paar *StudentA*, *StudentB* im Ergebnis enthalten ist, soll nicht auch noch das Paar *StudentB*, *StudentA* ausgegeben werden.

Lösungsvorschlag

```

SELECT s1.Name, s2.Name
FROM Studierende s1, Studierende s2
WHERE
 s1.Semester = s2.Semester AND
 s1.MatrNr < s2.MatrNr;

```

| name         | name      |
|--------------|-----------|
| Theophrastos | Feuerbach |

(1 row)

Sequenzdiagramm

- (e) In welchen Fächern ist die Durchschnittsnote schlechter als 2? Geben Sie die Vorlesungsnummer und den Titel aus.

Lösungsvorschlag

```
SELECT v.VorlNr, v.Titel
FROM prüfen p, Vorlesungen v
WHERE p.VorlNr = v.VorlNr
GROUP BY v.VorlNr, v.Titel
HAVING AVG(p.Note) > 2;
```

| vorlnr | titel          |
|--------|----------------|
| 4630   | Die 3 Kritiken |

(1 row)

- (f) Finden Sie alle Paare von Studierenden (*MatrNr* duplikatfrei ausgeben), die mindestens zwei Vorlesungen gemeinsam hören.

Lösungsvorschlag

```
SELECT h1.MatrNr, h2.MatrNr
FROM hören h1, hören h2
WHERE
 h1.VorlNr = h2.VorlNr AND
 h1.MatrNr < h2.MatrNr
GROUP BY h1.MatrNr, h2.MatrNr
HAVING COUNT(*) > 1;
```

| matrnr | matrnr |
|--------|--------|
| 28106  | 29120  |
| 28106  | 29555  |

(2 rows)

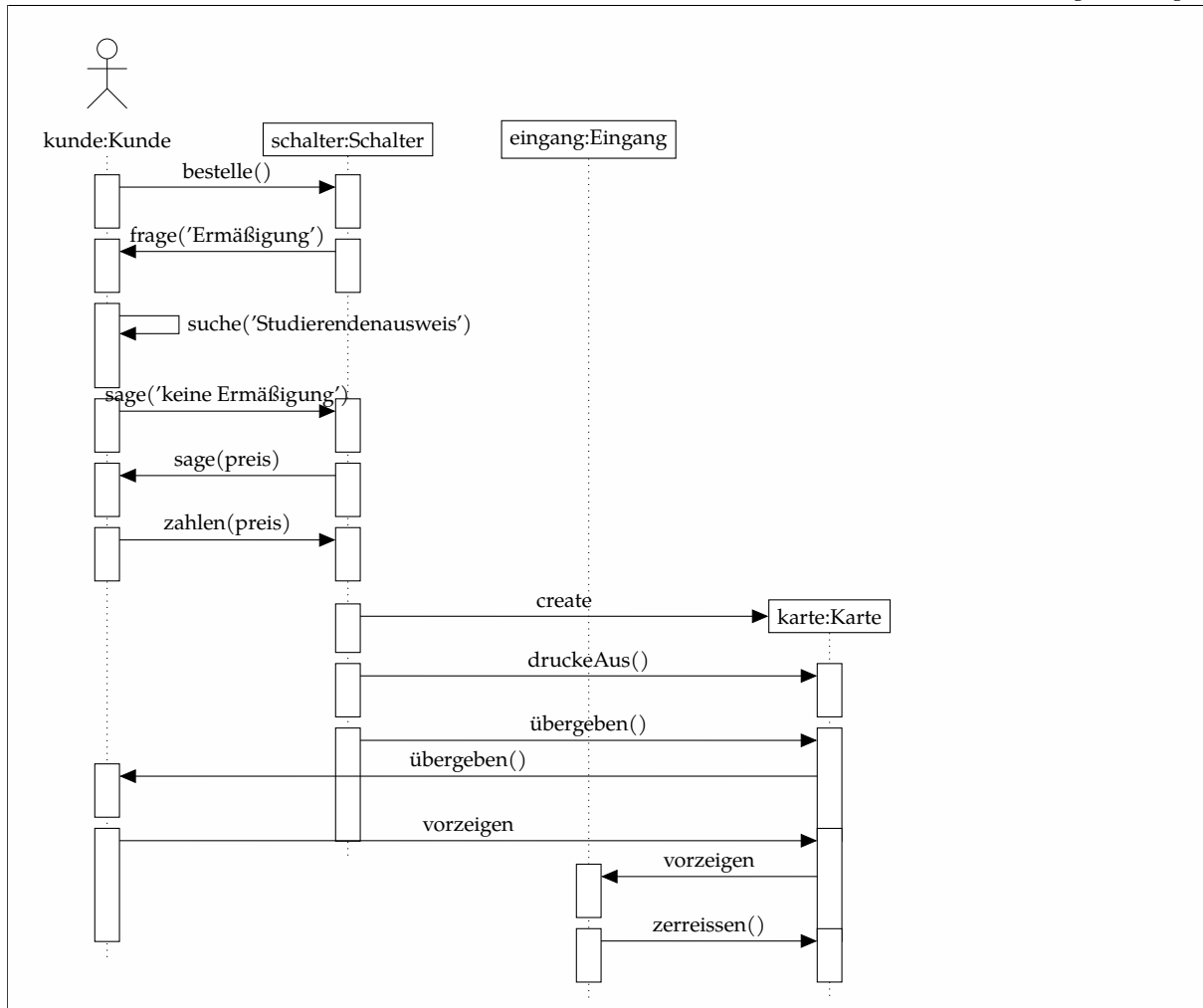
## 66116 / 2020 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 4

Erstellen Sie ein UML-Sequenzdiagramm zur Abbildung des folgenden Szenarios:

- Ein *Kunde* bestellt Karten an einem *Schalter*. Daraufhin wird er vom Schalter gefragt, ob er eine Ermäßigung nachweisen kann.
- Der *Kunde* sucht, leider erfolglos, seinen Studierendenausweis und sagt dem *Schalter*, dass er keinen Ermäßigungsgrund vorweisen kann.
- Der *Schalter* sagt dem Kunden den Preis der *Karten* und der Kunde gibt dem *Schalter* das notwendige Geld.

- (d) Der *Schalter* erstellt die *Karten*, druckt diese aus und übergibt sie dem Kunden.
- (e) Dieser geht mit den *Karten* zum *Eingang*, worauf die *Karten* zum Nachweis des Eintritts zerrissen werden.

Lösungsvorschlag



## 66116 / 2020 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 5

- (a) Implementieren Sie ein Programm in einer objektorientierten Programmiersprache, z. B. Java, für das folgende UML-Klassendiagramm.

Die `shift`-Methode soll die `x`-Position eines Objektes um `xShift` verändern und die `y`-Position um `yShift`. Die `draw`-Methode soll die Werte der Attribute der Klasse auf der Konsole ausgeben (- dies kann in Java mit `System.out.println(...)` erfolgen).

Lösungsvorschlag

```

interface Drawable {
 public void draw();

```

```
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/fruehjahr/object2d/Drawable.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/fruehjahr/object2d/Drawable.java)

```
abstract class Object2D implements Drawable {
 public abstract void shift(int xShift, int yShift);
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/fruehjahr/object2d/Object2D.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/fruehjahr/object2d/Object2D.java)

```
public class Point extends Object2D {
 int xPos;
 int yPos;

 public Point(int x, int y) {
 xPos = x;
 yPos = y;
 }

 public void shift(int xShift, int yShift) {
 xPos += xShift;
 yPos += yShift;
 }

 public void draw() {
 System.out.println(String.format("xPos: %s, yPos: %s", xPos, yPos));
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/fruehjahr/object2d/Point.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/fruehjahr/object2d/Point.java)

```
public class Square extends Object2D {
 Point topLeft;
 Point bottomRight;

 public Square(int top, int left, int bottom, int right) {
 topLeft = new Point(left, top);
 bottomRight = new Point(right, bottom);
 }

 public void shift(int xShift, int yShift) {
 topLeft.shift(xShift, yShift);
 bottomRight.shift(xShift, yShift);
 }

 public void draw() {
 topLeft.draw();
 bottomRight.draw();
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/fruehjahr/object2d/Square.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/fruehjahr/object2d/Square.java)

- (b) Schreiben Sie eine Methode, die ein zweidimensionales Array aus ganzen Zahlen (Datentyp `int`) als Parameter bekommt und ein eindimensionales Array (bestehend aus ganzen Zahlen (Datentyp `int`)) zurückgibt, dessen Elemente jeweils der Summe der Einträge in der entsprechenden Zeile des zweidimensionalen Arrays entsprechen.

Achtung: Die Zeilen des zweidimensionalen Arrays können unterschiedlich lang sein.

Zur Vereinfachung sei die Signatur der Methode gegeben: `public int[] computeSum(int[] [] input)`

Lösungsvorschlag

```
public class ComputeSum {
 public static int[] computeSum(int[] [] input) {
 int[] output = new int[input.length];
 for (int i = 0; i < input.length; i++) {
 int[] numbers = input[i];
 int sum = 0;
 for (int j = 0; j < numbers.length; j++) {
 sum += numbers[j];
 }
 output[i] = sum;
 }
 return output;
 }

 public static void main(String[] args) {
 int[] [] input = new int[3] [];
 input[0] = new int[] { 1, 2, 3 };
 input[1] = new int[] { 4, 5 };
 input[2] = new int[] { 6 };
 int[] output = computeSum(input);
 for (int i = 0; i < output.length; i++) {
 System.out.println(output[i]);
 }
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/fruehjahr/ComputeSum.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/fruehjahr/ComputeSum.java)

- (c) Implementieren Sie eine einfach verkettete Liste in einer Klasse List (z. B. in Java), in der in jedem Listenelement ein String gespeichert wird. Die Klasse soll folgende Methoden bereitstellen:

- `void addFirst (String element)` : Diese Methode fügt ein Element am Anfang einer Liste ein.
- `void addLast (String element)` : Diese Methode hängt ein Element an das Ende der Liste an.
- `boolean exists(String element)` : Diese Methode gibt `true` zurück, wenn die Liste ein Element mit dem Inhalt `element` beinhaltet, andernfalls gibt sie `false` zurück.

Hinweis: Zwei `String`-Objekte können mittels der Funktion `equals(..)` verglichen werden.

```
public class List {

 Element head;

 class Element {
 String value;
 Element next;

 public Element(String value) {
 this.value = value;
 }
 }

 public List() {
 head = null;
 }

 void addFirst(String element) {
 Element newElement = new Element(element);
 if (head != null) {
 newElement.next = head;
 }
 head = newElement;
 }

 void addLast(String element) {
 Element nextElement = head;
 Element lastElement = head;
 while (nextElement != null) {
 lastElement = nextElement;
 nextElement = nextElement.next;
 }
 if (lastElement != null) {
 lastElement.next = new Element(element);
 } else {
 head = new Element(element);
 }
 }

 boolean exists(String element) {
 Element nextElement = head;
 while (nextElement != null) {
 if (nextElement.value.equals(element)) {
 return true;
 }
 nextElement = nextElement.next;
 }
 return false;
 }

 public static void main(String[] args) {
```

```
List list = new List();
list.addLast("two");
list.addFirst("one");
list.addLast("three");

System.out.println(list.exists("one"));
System.out.println(list.exists("four"));
Element nextElement = list.head;
while (nextElement != null) {
 System.out.println(nextElement.value);
 nextElement = nextElement.next;
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/fruehjahr/List.java](https://github.com/orgs/bschlangaul/examen/examen_66116/jahr_2020/fruehjahr/List.java)

## 66116 / 2020 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Hinweis: Bei Wahl dieser Aufgabe wird Wissen über das erweiterte Entity-Relationship-Modell (beispielsweise schwache Entity-Typen, Vererbung) sowie die Verfeinerung eines relationalen Schemas vorausgesetzt.

Gegeben seien folgende Informationen:

- Ein Wetterdienst - identifiziert durch einen eindeutigen Namen - betreibt mehrere Wetterstationen, die mit einer eindeutigen Nummer je Wetterdienst identifiziert werden können. Jede Wetterstation hat zudem mehrere Messgeräte, die wiederum pro Wetterstation einen eindeutigen Code besitzen und zudem eine Betriebsdauer.
- Ein Wetterdienst hat eine Adresse.
- Bei den Messgeräten wird unter anderem zwischen manuellen und digitalen Messgeräten unterschieden. Dabei kann ein Messgerät immer nur zu einer Kategorie gehören.
- Meteorologen sind Mitarbeiter eines Wetterdienstes, haben einen Namen und werden über eine Personalnummer identifiziert. Zudem soll gespeichert werden, in welcher Wetterstation welcher Mitarbeiter zu welchem Zeitpunkt arbeitet.
- Meteorologen können für eine Menge an Messgeräten verantwortlich sein. Manuelle Messgeräte werden von Meteorologen abgelesen.
- Ein Wettermoderator präsentiert das vorhergesagte Wetter eines Wetterdienstes für einen Fernsehsender.
- Für einen Wettermoderator wird der eindeutige Name und die Größe gespeichert. Ein Fernsehsender wird ebenfalls über den eindeutigen Namen identifiziert.

(a) Erstellen Sie für das oben gegebene Szenario ein geeignetes ER-Diagramm.



Verwenden Sie dabei - wenn angebracht - das Prinzip der Spezialisierung. Kennzeichnen Sie die Primärschlüssel der Entity-Typen, totale Teilnahmen (existenzabhängige Beziehungen) und schwache Entity-Typen.

Zeichnen Sie die Funktionalitäten der Relationship-Typen in das Diagramm ein.

- (b) Überführen Sie das in Teilaufgabe a) erstellte ER-Modell in ein verfeinertes relationales Schema. Kennzeichnen Sie die Schlüssel durch Unterstreichen. Datentypen müssen nicht angegeben werden. Die einzelnen Schritte müssen angegeben werden.

#### Lösungsvorschlag

1. Schritt: Starke Entity-Typen einfügen Wetterdienst Name, Adresse Meteorologe Personalnummer, Name Wettermoderator Name, Größe Fernsehsender Name 2. Schritt: Schwache Entity-Typen einfügen Wetterstation Name [Wetterdienst], Nummer Messgeraet Name [Wetterdienst], Nummer [Wetterstation], Code, Betriebsdauer 3. Schritt: Is-A-Beziehung einfügen ManuellesMessgeraet Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] DigitalesMessgeraet Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] 4. Schritt: Beziehungen einfügen Betreibt Name [Wetterdienst], Nummer Hat Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] ArbeitetIn Personalnummer [Meteorologe], Name [Wetterdienst], Nummer [Wetterstation], Zeitpunkt IstMitarbeiterVon Personalnummer [Meteorologe], Name [Wetterdienst] IstVerantwortlichFür Personalnummer [Meteorologe], Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] LiestAb Personalnummer [Meteorologe], Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] PraesentiertWetter ModeratorenName [Wettermoderator], FernsehsenderName [Fernsehsender], WetterdienstName [Wetterdienst] (Bei PraesentiertWetter sind nur zwei Felder Teil des Primärschlüssels, da es sich um eine 1 : 1 : n- Beziehung handelt.) 5. Schritt: Verfeinerung Wetterdienst Name, Adresse Wetterstation Name [Wetterdienst], Nummer [Wetterstation] Messgeraet Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet], Betriebsdauer, VerantwortlicherMitarbeiterPersonalnummer [Meteorologe] ManuellesMessgeraet Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] DigitalesMessgeraet Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] Meteorologe Personalnummer, Name, WetterdienstName [Wetterdienst] ArbeitetIn Personalnummer [Meteorologe], Name [Wetterdienst], Nummer [Wetterstation], Zeitpunkt LiestAb Personalnummer [Meteorologe], Name [Wetterdienst], Nummer [Wetterstation], Code [Messgeraet] Wettermoderator Name, Größe Fernsehsender Name PraesentiertWetter ModeratorenName [Wettermoderator], FernsehsenderName [Fernsehsender], WetterdienstName [Wetterdienst]

**66116 / 2020 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 2**

Gegeben sei der folgende Ausschnitt eines Schemas für die Verwaltung von Kollektionen:

Die Tabelle *Promi* beschreibt Promis über ihren eindeutigen Namen, ihr Alter und ihren Wohnort. Kollektion enthält Informationen über *Kollektionen*, nämlich deren eindeutigen Namen, das Jahr und die Saison. Die Tabelle *promotet* verwaltet über Referenzen, welcher Promi welche Kollektion promotet. *Kleidungsstück* speichert die IDs von Kleidungsstücken zusammen mit dem Hauptbestandteil und einer Referenz auf die zugehörige Kollektion. Die Tabelle *hat\_getragen* verwaltet über Referenzen, welcher Promi welches Kleidungsstück an welchem Datum getragen hat.

Beachten Sie bei der Formulierung der SQL-Anweisungen, dass die Ergebnisrelationen keine Duplikate enthalten dürfen. Sie dürfen geeignete Views definieren.

```
CREATE TABLE Promi (
 Name VARCHAR(255) PRIMARY KEY,
 Alter INTEGER,
 Wohnort VARCHAR(255)
);

CREATE TABLE Kollektion (
 Name VARCHAR(255) PRIMARY KEY,
 Jahr INTEGER,
 Saison VARCHAR(255)
);

CREATE TABLE Kleidungsstueck (
 ID INTEGER PRIMARY KEY,
 Hauptbestandteil VARCHAR(255),
 gehoert_zu VARCHAR(255) REFERENCES Kollektion(Name)
);

CREATE TABLE hat_getragen (
 PromiName VARCHAR(255) REFERENCES Promi(Name),
 KleidungsstueckID INTEGER REFERENCES Kleidungsstueck(ID),
 Datum DATE,
 PRIMARY KEY(PromiName, KleidungsstueckID, Datum)
);

CREATE TABLE promotet (
 PromiName VARCHAR(255),
 KollektionName VARCHAR(255),
 PRIMARY KEY(PromiName, KollektionName)
);

INSERT INTO Promi
 (Name, Alter, Wohnort)
VALUES
 ('Till Schweiger', 52, 'Dortmund'),
 ('Lena Meyer-Landrut', 30, 'Hannover');

INSERT INTO Kollektion VALUES
 ('Gerry Weber', 2020, 'Sommer'),
```

```
('H & M', 2020, 'Sommer');
```

```
INSERT INTO promotet
(PromiName, KollektionName)
VALUES
('Till Schweiger', 'Gerry Weber'),
('Lena Meyer-Landrut', 'H & M');

INSERT INTO Kleidungsstueck
(ID, Hauptbestandteil, gehoert_zu)
VALUES
(1, 'Hose', 'Gerry Weber');

INSERT INTO hat_getragen
(PromiName, KleidungsstueckID, Datum)
VALUES
('Till Schweiger', 1, '2021-08-03');
```

- (a) Schreiben Sie SQL-Anweisungen, welche die Tabelle `hat_getragen` inklusive aller benötigten Fremdschlüsselconstraints anlegt. Erläutern Sie kurz, warum die Spalte `Datum` Teil des Primärschlüssels ist.

Lösungsvorschlag

```
CREATE TABLE IF NOT EXISTS hat_getragen (
 PromiName VARCHAR(255) REFERENCES Promi(Name),
 KleidungsstueckID INTEGER REFERENCES Kleidungsstueck(ID),
 Datum DATE,
 PRIMARY KEY(PromiName, KleidungsstueckID, Datum)
);
```

Das Datenbanksystem achtet selbst darauf, dass Felder des Primärschlüssels nicht NULL sind. Deshalb muss man NOT NULL bei keinem der drei Felder angeben.

- (b) Schreiben Sie eine SQL-Anweisung, welche die Namen der Promis ausgibt, die eine Sommer-Kollektion promoten (Saison ist „Sommer“).

Lösungsvorschlag

```
SELECT p.PromiName
FROM promotet p, Kollektion k
WHERE
 k.Saison = 'Sommer' AND
 p.KollektionName = k.Name;

prominame

Till Schweiger
Lena Meyer-Landrut
(2 rows)
```

- (c) Schreiben Sie eine SQL-Anweisung, die die Namen aller Promis und der Kollektionen bestimmt, welche der Promi zwar promotet, aber daraus noch kein Kleidungsstück getragen hat.

Lösung mit NOT IN:

```
SELECT * FROM promotet p
WHERE p.KollektionName NOT IN (
 SELECT k.Name FROM Kollektion k
 INNER JOIN Kleidungsstueck s ON s.gehoert_zu = K.Name
 INNER JOIN hat_getragen t ON t.KleidungsstueckID = s.ID
 WHERE t.PromiName = p.PromiName
);
```

Mit EXCEPT:

```
(
 SELECT p.PromiName, p.KollektionName FROM promotet p
)
EXCEPT
(
 SELECT p.PromiName, p.KollektionName FROM promotet p
 INNER JOIN Kollektion k ON p.KollektionName = k.Name
 INNER JOIN Kleidungsstueck s ON s.gehoert_zu = k.Name
 INNER JOIN hat_getragen t ON t.KleidungsstueckID = s.ID
 WHERE t.PromiName = p.PromiName
);
```

- (d) Bestimmen Sie für die folgenden SQL-Anweisungen die minimale und maximale Anzahl an Tupeln im Ergebnis. Beziehen Sie sich dabei auf die Größe der einzelnen Tabellen.

Verwenden Sie für die Lösung folgende Notation: – Promi – beschreibt die Größe der Tabelle Promi.

(i)

```
SELECT k.Name
FROM Kollektion k, Kleidungsstueck kl
WHERE k.Name = kl.gehoert_zu and k.Jahr = 2018
GROUP BY k.Name
HAVING COUNT(kl.Hauptbestandteil) > 10;
```

– Kollektion – beschreibt die Anzahl der Tupel in der Tabelle Kollektion. Die minimale Anzahl an Tupeln im Ergebnis ist 0, da es sein kann, dass keine Kollektion den Anforderungen `k.Jahr = 2018` oder `HAVING COUNT(...)` genügt. Die maximale Anzahl an Tupeln im Ergebnis ist – Kollektion –, da nur Namen aus Kollektion ausgewählt werden.

(ii)

```
SELECT DISTINCT k.Jahr
FROM Kollektion k
WHERE k.Name IN (
```

```
SELECT pr.KollektionName
FROM Promi p, promotet pr
WHERE p.Alter < 30 AND pr.PromiName = p.Name
);
```

Lösungsvorschlag

Die minimale Anzahl an Tupeln im Ergebnis ist 0, da es sein kann, dass keine Kollektion promotet wird. Die maximale Anzahl ist das Minimum von – Kollektion – und 30, da die Promis, die Kollektionen beworben haben, die älter als 30 Jahre sind, selbst mindestens 30 Jahre alt sein müssen. Zugrundeliegende Annahmen: Kollektionen werden nur im Erscheinungsjahr von Promis beworben; Neugeborene, die Kollektionen bewerben, werden ggf. Promis.

- (e) Beschreiben Sie den Effekt der folgenden SQL-Anfrage in natürlicher Sprache

```
SELECT pr.KollektionName
FROM promotet pr, Promi p
WHERE pr.PromiName = p.Name
GROUP BY pr.KollektionName
HAVING COUNT (*) IN (
 SELECT MAX(anzahl)
 FROM (
 SELECT k.Name, COUNT(*) AS anzahl
 FROM Kollektion k, promotet pr
 WHERE k.Name = pr.KollektionName
 GROUP BY k.Name
) as tmp
);
```

Lösungsvorschlag

Die Abfrage gibt die Namen aller Kollektionen aus, bei denen die Anzahl der bewerbenden Promis am größten ist.

- (f) Formulieren Sie folgende SQL-Anfrage in relationaler Algebra. Die Lösung kann in Baum- oder in Term-Schreibweise angegeben werden, wobei eine Schreibweise genügt.

```
SELECT p.Wohnort
FROM Promi p, promotet pr, Kollektion k
WHERE
 p.Name = pr.PromiName AND
 k.Name = pr.KollektionName AND
 k.Jahr = 2018;
```

- (i) Konvertieren Sie zunächst die gegebene SQL-Anfrage in die zugehörige Anfrage in relationaler Algebra nach Standard-Algorithmus.

$$\pi_{\text{Promi.Wohnort}}(\sigma_{\text{Promi.Name=promotet.PromiName} \wedge \text{Kollektion.Name=promotet.KollektionName} \wedge \text{Kollektion.Jahr=2018}}(\text{Promi} \times \text{promotet} \times \text{Kollektion}))$$

- (ii) Führen Sie anschließend eine relationale Optimierung durch. Beschreiben und begründen Sie dabei kurz jeden durchgeführten Schritt.

1. Schritt: Um die kartesischen Produkte in Joins zu verwandeln, muss die Selektion in drei Selektionen zerlegt werden.

$$\pi_{\text{Promi.Wohnort}}(\sigma_{\text{Promi.Name=promotet.PromiName}}(\sigma_{\text{Kollektion.Name=promotet.KollektionName}}(\sigma_{\text{Kollektion.Jahr=2018}}(\text{Promi} \times (\text{promotet} \times \text{Kollektion}))))$$

2. Schritt: Man kann die Selektion nach dem Jahr ausführen, bevor die kartesischen Produkte ausgeführt werden. Dadurch werden von vornherein nur die Kollektionen betrachtet, die in dem entsprechenden Jahr aufgetreten sind.

$$\pi_{\text{Promi.Wohnort}}(\sigma_{\text{Promi.Name = promotet.PromiName}}(\sigma_{\text{Kollektion.Name = promotet.KollektionName}}(\sigma_{\text{Kollektion.Jahr=2018}}(\text{Promi} \times (\text{promotet} \times \text{Kollektion}))))$$

3. Schritt: Die zweitinnerste Selektion kann direkt nach dem innersten kartesischen Produkt ausgeführt werden; dadurch wird das Gesamtprodukt nicht so groß. Man benötigt also weniger Rechenzeit und Speicher.

$$\pi_{\text{Promi.Wohnort}}(\sigma_{\text{Promi.Name = promotet.PromiName}}(\text{Promi} \times \sigma_{\text{Kollektion.Name = promotet.KollektionName}}(\text{promotet} \times \text{Kollektion})))$$

4. Schritt: Beide Selektionen in Verbindung mit dem jeweiligen kartesischen Produkt können in einen Join verwandelt werden. So wird

nicht zuerst das gesamte Produkt berechnet und danach die passenden Einträge ausgewählt, sondern gleich nur die zusammengehörigen Einträge kombiniert. Auch dies spart Rechenzeit und Speicher.

$$\pi_{\text{Promi.Wohnort}}(\text{Promi} \bowtie \text{Promi.Name} = \text{promotet.PromiName}(\text{promotet} \bowtie \text{Kollektion.Na})$$

## 66116 / 2020 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 3

Gegeben sei folgendes relationales Schema  $R$  in erster Normalform:

$R: [A, B, C, D, E, F]$

Für  $R$  gelte folgende Menge  $FD$  funktionaler Abhängigkeiten:

$$FA = \left\{ \begin{array}{l} \{ A \} \rightarrow \{ F \}, \\ \{ C, E, F \} \rightarrow \{ A, B \}, \\ \{ A, E \} \rightarrow \{ B \}, \\ \{ B, C \} \rightarrow \{ D \}, \\ \{ A, F \} \rightarrow \{ C \}, \end{array} \right\}$$

- (a) Bestimmen Sie alle Kandidatenschlüssel/Schlüsselkandidaten von  $R$  mit  $FD$ . Begründen Sie Ihre Antwort. Begründen Sie zudem, warum es keine weiteren Kandidatenschlüssel/Schlüsselkandidaten gibt.

*Hinweis: Die Angabe von Attributmengen, die keine Kandidatenschlüssel sind, führt zu Abzügen.*

Lösungsvorschlag

$E$  muss in allen Superschlüsseln enthalten sein, denn es steht nicht auf der rechten Seite von  $FD$  (\*).

$D$  kann in keinem Schlüsselkandidaten vorkommen, denn es steht nur auf der rechten Seite von  $FD$  (\*\*).

$E$  allein ist kein Schlüsselkandidat (\*\*\*).

$AE$  führt über  $FD$  zu  $B$ ,  $A$  zu  $F$ ,  $AF$  zu  $C$  und  $BC$  zu  $D$ , also ist  $AE$  ein Superschlüssel und damit wegen (\*) und (\*\*\*) ein Schlüsselkandidat. Wegen (\*) enthält jeder Superschlüssel, der  $A$  enthält,  $AE$ . Also ist kein weiterer Superschlüssel, der  $A$  enthält, ein Schlüsselkandidat (\*\*\*\*).

$BE$ ,  $CE$  und  $EF$  sind keine Superschlüssel, also auch keine Schlüsselkandidaten.

$BCE$  ist kein Superschlüssel, da  $A$  und  $F$  nicht erreicht werden können.

$BEF$  ist kein Superschlüssel, da  $A$ ,  $D$  und  $F$  nicht erreicht werden können.

$CEF$  führt über  $FD$  zu  $AB$ ,  $BC$  führt dann zu  $D$ , also ist  $CEF$  ein Superschlüssel.

Wegen (\*), (\*\*) und weil CE und EF keine Superschlüssel sind, ist CEF ein Schlüsselkandidat.

Das waren alle dreielementigen Buchstabenkombinationen, die (\*), (\*\*) und (\*\*\*\*) genügen. Vierelementig ist nur BCEF und das enthält CEF, ist also kein Schlüsselkandidat.

Die einzigen Schlüsselkandidaten sind folglich AE und CEF.

- (b) Prüfen Sie, ob R mit FD in 2NF bzw. 3NF ist.

Lösungsvorschlag

R mit FD ist nicht in 2NF, denn bei Wahl des Schlüsselkandidaten AE hängt F von A, also nur einem Teil des Schlüssels, ab. Also ist  $AE \rightarrow F$  nicht voll funktional. Damit ist R mit FD auch nicht in 3NF, denn  $3NF \subseteq 2NF$ .

- (c) Bestimmen Sie mit folgenden Schritten eine kanonische Überdeckung  $FD_c$  von FD. Begründen Sie jede Ihrer Entscheidungen:

- (i) Führen Sie eine Linksreduktion von FD durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Linksreduktion an ( $FD;$ ).

Lösungsvorschlag

$$FA = \left\{ \begin{array}{l} \{A\} \rightarrow \{F\}, \\ \{C, E, F\} \rightarrow \{A, B\}, \\ \{A, E\} \rightarrow \{B\}, \\ \{B, C\} \rightarrow \{D\}, \\ \{A\} \rightarrow \{C\}, \end{array} \right\}$$

- (ii) Führen Sie eine Rechtsreduktion des Ergebnisses der Linksreduktion ( $FD;$ ) durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Rechtsreduktion an ( $FD$ ).

Lösungsvorschlag

$$FA = \left\{ \begin{array}{l} \{A\} \rightarrow \{F\}, \\ \{C, E, F\} \rightarrow \{A\}, \\ \{A, E\} \rightarrow \{B\}, \\ \{B, C\} \rightarrow \{D\}, \\ \{A\} \rightarrow \{C\}, \end{array} \right\}$$



- (iii) Bestimmen Sie eine kanonische Überdeckung FD. von FD auf Basis des Ergebnisses der Rechtsreduktion (FD).

Lösungsvorschlag

$$FA = \left\{ \begin{array}{l} \{ A \} \rightarrow \{ F, C \}, \\ \{ C, E, F \} \rightarrow \{ A \}, \\ \{ A, E \} \rightarrow \{ B \}, \\ \{ B, C \} \rightarrow \{ D \}, \end{array} \right\}$$

- (d) Zerlegen Sie R mit FDc mithilfe des Synthesalgorithmus in 3NF. Geben Sie zudem alle funktionalen Abhängigkeiten der erzeugten Relationenschemata an.
- (e) Prüfen Sie für alle Relationen der Zerlegung aus 4., ob sie jeweils in BCNF sind.

## 66116 / 2020 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 4

- (a) Betrachten Sie den folgenden Schedule S:

| $T_1$    | $T_2$    | $T_3$    |
|----------|----------|----------|
|          | $r_2(z)$ |          |
|          |          | $w_3(y)$ |
|          | $r_2(x)$ |          |
| $w_1(x)$ | $w_2(x)$ |          |
|          |          | $r_3(z)$ |
|          |          | $c_3$    |
|          | $w_2(z)$ |          |
| $w_1(y)$ |          |          |
| $c_1$    |          |          |
|          | $c_2$    |          |

Geben Sie den Ausgabeschedule (einschließlich der Operationen zur Sperranforderung und -freigabe) im rigorosen Zweiphasen-Sperrprotokoll für den obigen Eingabeschedule S an.

| $T_1$                   | $T_2$                   | $T_3$                   |
|-------------------------|-------------------------|-------------------------|
|                         | $\text{rlock}_2(z)$     |                         |
|                         | $r_2(z)$                |                         |
|                         |                         | $\text{xlock}_3(y)$     |
|                         |                         | $w_3(y)$                |
|                         | $\text{rlock}_2(x)$     |                         |
|                         | $r_2(x)$                |                         |
|                         | $\text{xlock}_2(x)$     |                         |
|                         | $w_2(x)$                |                         |
|                         |                         | $\text{rlock}_3(z)$     |
|                         |                         | $r_3(z)$                |
|                         |                         | $c_3$                   |
|                         |                         | $\text{unlock}_2(y, z)$ |
|                         | $\text{xlock}_2(z)$     |                         |
|                         | $w_2(z)$                |                         |
|                         | $c_2$                   |                         |
|                         | $\text{unlock}_2(x, z)$ |                         |
| $\text{xlock}_1(x)$     |                         |                         |
| $w_1(x)$                |                         |                         |
| $\text{xlock}_1(y)$     |                         |                         |
| $w_1(y)$                |                         |                         |
| $c_1$                   |                         |                         |
| $\text{unlock}_2(x, z)$ |                         |                         |

- (b) Beschreiben Sie den Unterschied zwischen dem herkömmlichen Zweiphasen-Sperrprotokoll (2PL) und dem rigorosen Zweiphasen-Sperrprotokoll. Warum wird in der Praxis häufiger das rigorose Zweiphasen-Sperrprotokoll verwendet?

Bei beiden Protokollen fordert die Transaktion erst alle Sperren an (Anforderungsphase) und gibt sie später frei (Freigabephase).

- Beim strengen oder rigorosen 2PL werden die Sperren dann angefordert, wenn sie benötigt werden, beim konservativen 2PL werden alle Sperren zu Beginn gemeinsam angefordert.

- Beim strengen oder rigorosen 2PL werden die Lesesperren bis zum Commit, die Schreibsperren sogar bis nach dem Commit gehalten. Beim konservativen 2PL werden dagegen die Sperren freigegeben, wenn sie nicht mehr benötigt werden.

Das rigorose 2PL wird der Praxis häufiger verwendet, weil dabei nicht zu Beginn der Transaktion bekannt sein muss, welche Sperren benötigt werden, und durch die schrittweise Anforderung der Sperren unter Umständen ein höheres Maß an Parallelität erreicht werden kann.

## 66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 1

- (a) Definieren Sie die Begriffe „*partielle Korrektheit*“ und „*totale Korrektheit*“ und grenzen Sie sie voneinander ab.

Lösungsvorschlag

**partielle Korrektheit** Ein Programmcode wird bezüglich einer Vorbedingung  $P$  und einer Nachbedingung  $Q$  partiell korrekt genannt, wenn bei einer Eingabe, die die Vorbedingung  $P$  erfüllt, jedes Ergebnis die Nachbedingung  $Q$  erfüllt. Dabei ist es noch möglich, dass das Programm nicht für jede Eingabe ein Ergebnis liefert, also nicht für jede Eingabe terminiert.

**totale Korrektheit** Ein Code wird total korrekt genannt, wenn er partiell korrekt ist und zusätzlich für jede Eingabe, die die Vorbedingung  $P$  erfüllt, terminiert. Aus der Definition folgt sofort, dass total korrekte Programme auch immer partiell korrekt sind.

- (b) Geben Sie die Verifikationsregel für die abweisende Schleife `while(b) { A }` an.

Lösungsvorschlag

Eine abweisende Schleife ist eine while-Schleife, da die Schleifenbedingung schon bei der ersten Prüfung falsch sein kann und somit die Schleife abgewiesen wird.

Um die schwächste Vorbedingung eines Ausdrucks der Form „`while(b) { A }`“ zu finden, verwendet man eine *Schleifeninvariante*. Sie ist ein Prädikat für das

$$\{I \wedge b\}A\{I\}$$

gilt. Die Schleifeninvariante gilt also sowohl vor, während und nach der Schleife.

- (c) Erläutern Sie kurz und prägnant die Schritte zur Verifikation einer abweisenden Schleife mit Vorbedingung  $P$  und Nachbedingung  $Q$ .

**Schritt 0:** Schleifeninvariante  $I$  finden

**Schritt 1:**  $I$  gilt vor Schleifenbeginn,

d.h.  $P \Rightarrow \text{wp}(\text{"Code vor Schleife"}, I)$

**Schritt 2:**  $I$  gilt nach jedem Schleifendurchlauf

d.h.  $I \wedge b \Rightarrow \text{wp}(\text{"Code in der Schleife"}, I)$

**Schritt 3:** Bei Terminierung der Schleife liefert Methode das gewünschte Ergebnis,

d.h.  $I \wedge \neg b \Rightarrow \text{wp}(\text{"Code nach der Schleife"}, Q)$

(d) Wie kann man die Terminierung einer Schleife beweisen?

Zum Beweis der Terminierung einer Schleife muss eine Terminierungsfunktion  $T$  angegeben werden:

$$T: V \rightarrow \mathbb{N}$$

$V$  ist eine Teilmenge der Ausdrücke über die Variablenwerte der Schleife

Die Terminierungsfunktion muss folgende Eigenschaften besitzen:

- Ihre Werte sind natürliche Zahlen (einschließlich 0).
- Jede Ausführung des Schleifenrumpfs verringert ihren Wert (streng monoton fallend).
- Die Schleifenbedingung ist false, wenn  $T = 0$ .

$T$  ist die obere Schranke für die noch ausstehende Anzahl von Schleifendurchläufen.

Beweise für Terminierung sind nicht immer möglich!

- (e) Geben Sie für das folgende Suchprogramm die nummerierten Zusicherungen an. Lassen Sie dabei jeweils die invariante Vorbedingung  $P$  des Suchprogramms weg. Schreiben Sie nicht auf dem Aufgabenblatt!

$$P \equiv n > 0 \wedge a_0 \dots a_{n-1} \in \mathbb{Z}^n \wedge m \in \mathbb{Z}$$

```

int i = -1;
// (1)
int j = 0;
// (2)
while (i == -1 && j < n) // (3)
{ // (4)
 if (a[j] == m) {
 // (5)
 i = j;
 // (6)
 } else {
 // (7)
 j = j + 1;
 // (8)
 }
 // (9)
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/Verifikation.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/Verifikation.java)

$$Q \equiv P \wedge (i = -1 \wedge \forall 0 \leq k < n: a_k \neq m) \vee (i \geq 0 \wedge a_i = m)$$

Lösungsvorschlag

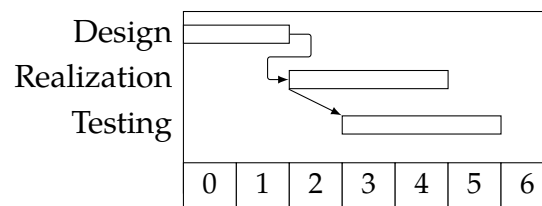
In dieser Aufgabenstellung fällt die Vorbedingung mit der Invariante zusammen. Es muss kein wp-Kalkül berechnet werden, sondern „nur“ die Zuweisungen nachverfolgt werden, um zum Schluss die Nachbedingung zu erhalten.

1.  $(i = -1) \wedge P$
2.  $(i = -1) \wedge (j = 0) \wedge P$
3.  $(i = -1) \wedge (0 \leq j < n) \wedge P$
4.  $(i = -1) \wedge (0 \leq j < n) \wedge P$
5.  $(i = -1) \wedge (a_j = m) \wedge (0 \leq j < n) \wedge P$
6.  $(i \geq 0) \wedge i = j \wedge (a_j = m) \wedge (0 \leq j < n) \wedge P$
7.  $(i = -1) \wedge (\forall 0 \leq j < n: a_j \neq m) \wedge P$
8.  $(i = -1) \wedge (\forall 0 < j \leq n: a_j \neq m) \wedge P$
9.  $((i = -1) \wedge (\forall 0 \leq k < j: a_k \neq m)) \vee ((i \geq 0) \wedge (a_i = m)) \wedge P$

## 66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Die Planung eines Softwareprojekts kann z. B. in Form von Gantt-Diagrammen oder CPM-Netzwerken (kritischer Pfad Methode) festgehalten werden.

Folgendes Gantt-Diagramm zeigt einen Teil der Projektplanung in einem klassischen Softwareentwicklungsprozess:



- (a) Im Diagramm werden 3 Phasen aus dem klassischen Softwareentwicklungsprozess genannt. Welche Phase sollte dem Design (Entwurf) immer vorangehen?

Lösungsvorschlag

Die Anforderungsdefinition

- (b) Wandeln Sie das Gantt-Diagramm in ein CPM-Netzwerk um. Fügen Sie dazu einen zusätzlichen Start- und Endknoten hinzu. Das Ende des Projekts ist durch das Ende aller Aktivitäten bedingt.

Lösungsvorschlag

$D_A$  Design Anfang

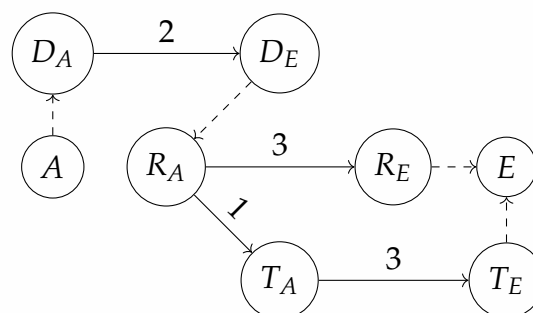
$R_A$  Realization Anfang

$T_A$  Testing Anfang

$D_E$  Design Ende

$R_E$  Realization Ende

$T_E$  Testing Ende



- (c) Welche im obigen Gantt-Diagramm nicht enthaltenen Beziehungsarten zwischen Aktivitäten können in einem Gantt-Diagramm noch auftreten? Nennen Sie auch deren Bedeutung.

Lösungsvorschlag

Diese Beziehungsarten sind im obigen Gantt-Diagramm vorhanden:

**Normalfolge EA:** *end-to-start relationship* Anordnungsbeziehung vom Ende eines Vorgangs zum Anfang seines Nachfolgers.

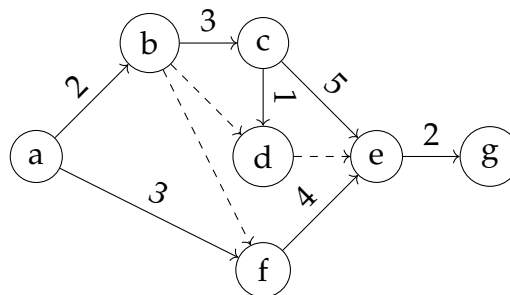
**Anfangsfolge AA:** *start-to-start relationship* Anordnungsbeziehung vom Anfang eines Vorgangs zum Anfang seines Nachfolgers.

Diese Beziehungsarten sind im obigen Gantt-Diagramm *nicht* vorhanden:

**Endefolge EE:** *finish-to-finish relationship* Anordnungsbeziehung vom Ende eines Vorgangs zum Ende seines Nachfolgers.

**Sprungfolge AE:** *start-to-finish relationship* Anordnungsbeziehung vom Anfang eines Vorgangs zum Ende seines Nachfolgers

Gegeben sei nun das folgende CPM-Netzwerk:



(d) Geben Sie für jedes Ereignis die früheste Zeit an.

Lösungsvorschlag

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $FZ_i$ : Frühester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann.

| $i$ | Nebenrechnung          | $FZ_i$ |
|-----|------------------------|--------|
| a   |                        | 0      |
| b   |                        | 2      |
| c   |                        | 5      |
| d   | $\max(2_b, 6_c)$       | 6      |
| e   | $\max(6_d, 10_e, 7_f)$ | 10     |
| f   | $\max(3_f, 2_b)$       | 3      |
| g   |                        | 12     |

(e) Geben Sie für jedes Ereignis die späteste Zeit an.

Lösungsvorschlag

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;

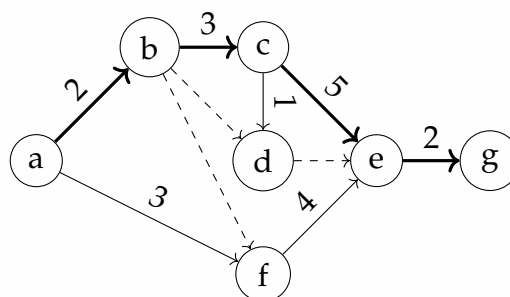
$SZ_i$ : Spätester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann. \_\_\_\_\_

| $i$ | Nebenrechnung          | $SZ_i$ |
|-----|------------------------|--------|
| g   |                        | 12     |
| f   |                        | 6      |
| e   |                        | 10     |
| d   |                        | 10     |
| c   | $\min(9_d, 5_e)$       | 5      |
| b   | $\min(2_c, 10_d, 6_f)$ | 2      |
| a   |                        | 0      |

- (f) Geben Sie einen kritischen Pfad durch das Netz an! Wie wirkt sich eine Verzögerung von 5 Zeiteinheiten auf dem kritischen Pfad auf das Projektende aus?

Lösungsvorschlag

| $i$    | a | b | c | d  | e  | f | g  |
|--------|---|---|---|----|----|---|----|
| $FZ_i$ | 0 | 2 | 5 | 6  | 10 | 3 | 12 |
| $SZ_i$ | 0 | 2 | 5 | 10 | 10 | 6 | 12 |
| GP     | 0 | 0 | 0 | 3  | 0  | 3 | 0  |

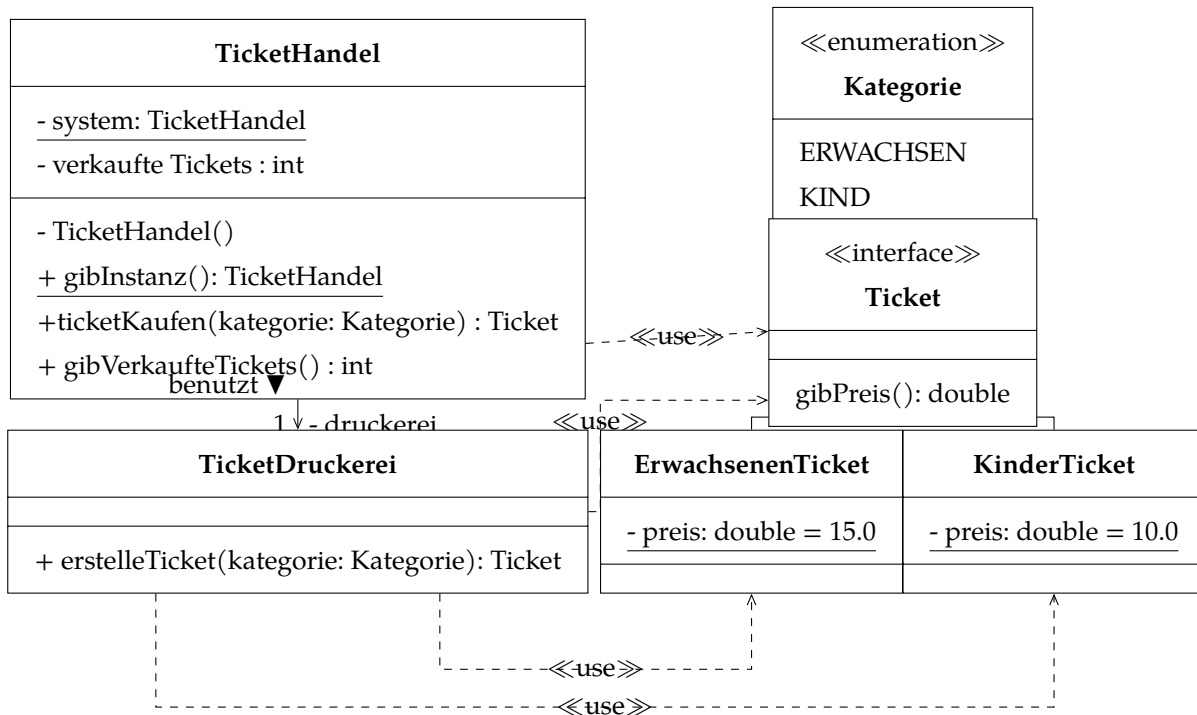


Kritischer Pfad:  $a \rightarrow b \rightarrow c \rightarrow e \rightarrow g$

Das Projekt verlängert sich um 5 Zeiteinheiten.



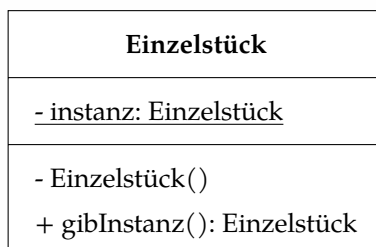
## 66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3



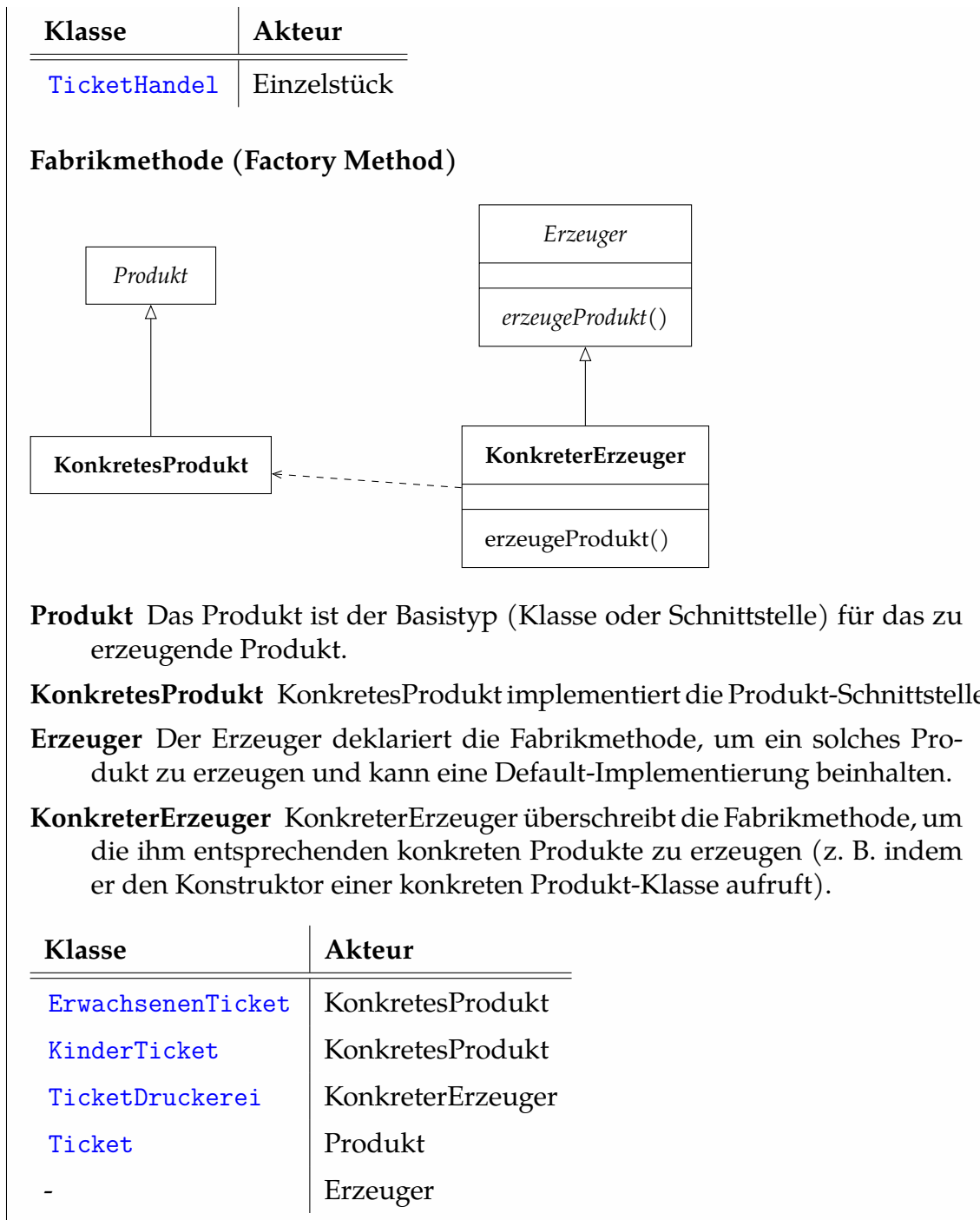
Ihnen sei ein UML-Klassendiagramm zu folgendem Szenario gegeben. Ein Benutzer (nicht im Diagramm enthalten) kann über einen **TicketHandel** Tickets erwerben. Dabei muss der Benutzer eine der zwei Ticketkategorien angeben. Das Handelsystem benutzt eine **TicketDruckerei**, um ein passendes **Ticket** für den Benutzer zu erzeugen.

- (a) Im angegebenen Klassendiagramm wurden zwei unterschiedliche Entwurfsmuster verwendet. Um welche Muster handelt es sich? Geben Sie jeweils den Namen des Musters sowie die Elemente des Klassendiagramms an, mit denen diese Muster im Zusammenhang stehen. ACHTUNG: Es handelt sich dabei *nicht* um das *Interface*- oder das *Vererbungsmuster*.

Lösungsvorschlag

**Einzelstück (Singleton)**

**Einzelstück (Singleton)** stellt eine statische Methode bereit, mit deren Hilfe die Klienten nur auf eine einzige Instanz der Klasse zugreifen können.



(b) Nennen Sie zwei generelle Vorteile von Entwurfsmustern.

Lösungsvorschlag

- Wiederverwendung einer bewährten Lösung für eine bestimmte Problemstellungen
- Verbesserung der Kommunikation unter EntwicklerInnen

(c) Geben Sie eine Implementierung der Klasse [TicketHandel](#) an. Bei der Methode [ticketKaufen\(\)](#) wird die Anzahl der verkauften Tickets um 1 erhöht und ein

entsprechendes Ticket erstellt und zurückgegeben. Beachten Sie den Hinweis auf der nächsten Seite.

Lösungsvorschlag

```
public class TicketHandel {
 private static TicketHandel system;
 private int verkaufteTickets;
 private TicketDruckerei druckerei;

 private TicketHandel() {
 druckerei = new TicketDruckerei();
 verkaufteTickets = 0;
 }

 public static TicketHandel gibInstanz() {
 if (system == null) {
 system = new TicketHandel();
 }
 return system;
 }

 public Ticket ticketKaufen(Kategorie kategorie) {
 verkaufteTickets++;
 return druckerei.erstelleTicket(kategorie);
 }

 public int gibVerkaufteTickets() {
 return verkaufteTickets;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/ticket/TicketHandel.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/ticket/TicketHandel.java)

(d) Geben Sie eine Implementierung der Klasse `TicketDruckerei` an.

Lösungsvorschlag

```
public class TicketDruckerei {
 public Ticket erstelleTicket(Kategorie kategorie) {
 if (kategorie == Kategorie.ERWACHSENEN) {
 return new ErwachsenenTicket();
 } else {
 return new KinderTicket();
 }
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/ticket/TicketDruckerei.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/ticket/TicketDruckerei.java)

(e) Geben Sie eine Implementierung der Klasse `KinderTicket` an.

Lösungsvorschlag

```
public class KinderTicket implements Ticket {
 private static double preis = 10.0;
```

```

public double gibPreis() {
 return preis;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/ticket/KinderTicket.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/ticket/KinderTicket.java)

*Hinweis:* Die Implementierungen *müssen* sowohl dem Klassendiagramm, als auch den Konzepten der verwendeten Muster entsprechen. Verwenden Sie eine objekt-orientierte Programmiersprache, vorzugsweise *Java*. Sie müssen sich an der nachfolgenden Testmethode und ihrer Ausgabe orientieren. Die Testmethode muss mit Ihrer Implementierung ausführbar sein und sich semantisch korrekt verhalten.

Quelltext der Testmethode:

```

public static void main(String[] args) {
 TicketHandel.gibInstanz().ticketKaufen(Kategorie.ERWACHSEN);
 TicketHandel.gibInstanz().ticketKaufen(Kategorie.KIND);
 System.out.println("Anzahl verkaufter Tickets: " +
 ↳ TicketHandel.gibInstanz().gibVerkaufteTickets());
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/ticket/Test.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/ticket/Test.java)

Konsolenausgabe:

Anzahl verkaufter Tickets: 2

## 66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4

Diese Aufgabe behandelt *Wortpalindrome*, also Wörter, die vorwärts und rückwärts gelesen jeweils dieselbe Zeichenkette bilden, z. B. Otto oder Rentner. Leere Wortpalindrome (also Wortpalindrome der Wortlänge 0) sind dabei nicht zulässig.

Folgende *Java-Methode* prüft, ob das übergebene Zeichen-Array ein Wortpalindrom darstellt:

```

public static boolean istWortpalindrom(char[] wort) { // 1
 boolean resultat = false; // 2
 if (wort != null) { // 3
 int laenge = wort.length; // 4
 if (laenge >= 2) { // 5
 resultat = true; // 6
 for (int i = 0; i < laenge / 2; ++i) { // 7
 char c1 = wort[i]; // 8
 char c2 = wort[laenge - 1 - i]; // 9
 if (Character.toLowerCase(c1) != Character.toLowerCase(c2)) // 10
 { // 11
 resultat = false; // 12
 break; // 13
 } // 14
 } // 15
 }
 }
}

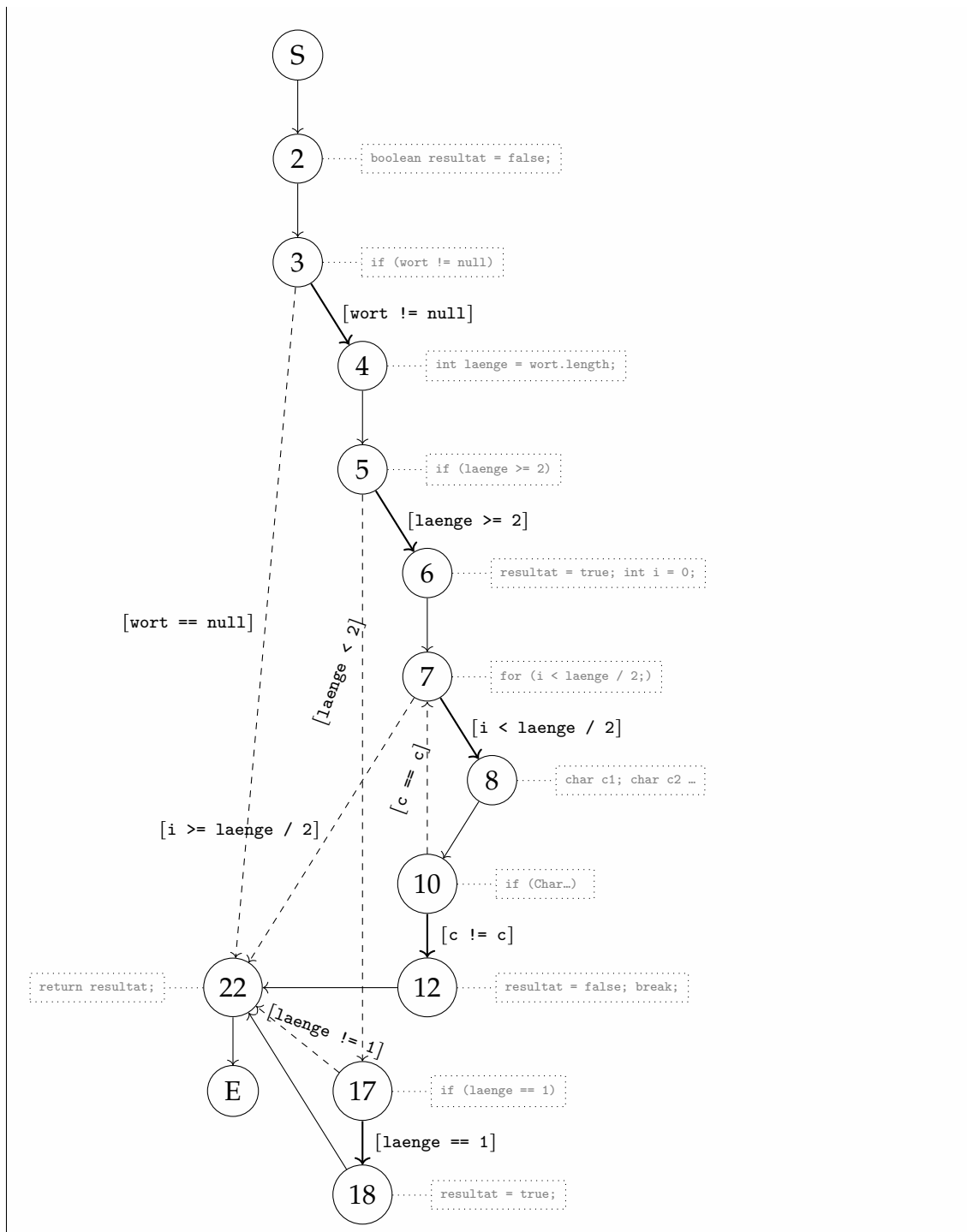
```

```
 } else { // 16
 if (laenge == 1) { // 17
 resultat = true; // 18
 } // 19
 } // 20
 } // 21
 return resultat; // 22
} // 23
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/Palindrom.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/Palindrom.java)

- (a) Geben Sie für die Methode einen *Kontrollflussgraphen* an, wobei Sie die Knoten mit den jeweiligen Zeilennummern im Quelltext beschriften.

Lösungsvorschlag



- (b) Geben Sie eine *minimale Testmenge* an, die das Kriterium der Anweisungsüberdeckung erfüllt.

Hinweis: Eine *Testmenge* ist *minimal*, wenn es keine Testmenge mit einer kleineren Zahl von Testfällen gibt. Die Minimalität muss *nicht* bewiesen werden.

```
isWortpalindrom(new char[] { 'a' }):
```

Ⓢ - ② - ③ - ④ - ⑤ - ⑰ - ⑱ - ㉒ - Ⓔ

```
isWortpalindrom(new char[] { 'a', 'b' }):
```

(S) - (2) - (3) - (4) - (5) - (6) - (7) - (8) - (10) - (12) - (22) - (E)

- (c) Geben Sie eine *minimale Testmenge* an, die das Kriterium der *Boundary-Interior-Pfadüberdeckung* erfüllt.

Hinweis: Das Kriterium *Boundary-Interior-Pfadüberdeckung* beschreibt einen Spezialfall der Pfadüberdeckung, wobei nur Pfade berücksichtigt werden, bei denen jede Schleife nicht mehr als zweimal durchlaufen wird.

Lösungsvorschlag

Es gibt noch ganz viele infeasible Pfade, die hier nicht aufgeführt werden.

### Äußere Pfade

```
- isWortpalindrom(null):
```

(S) - (2) - (3) - (22) - (E)

```
- isWortpalindrom(new char[] { }):
```

(S) - (2) - (3) - (4) - (5) - (17) - (22) - (E)

```
- isWortpalindrom(new char[] { 'a' }):
```

(S) - (2) - (3) - (4) - (5) - (17) - (18) - (22) - (E)

**Grenzpfade (boundary paths, boundary test)** Für Schleifen fester Lauflänge ist diese Testfallgruppe leer.

```
isWortpalindrom(new char[] { 'a', 'a' }):
```

(S) - (2) - (3) - (4) - (5) - (6) - (7) - (8) - (10) - (7) - (22) - (E)

```
isWortpalindrom(new char[] { 'a', 'b' }):
```

(S) - (2) - (3) - (4) - (5) - (6) - (7) - (8) - (10) - (12) - (22) - (E)

### Innere Pfade (interior test)

```
- isWortpalindrom(new char[] { 'a', 'a', 'a', 'a' }):
```

(S) - (2) - (3) - (4) - (5) - (6) - (7) - (8) - (10) - (7) - (8) - (10) - (7) - (22) - (E)

```
- isWortpalindrom(new char[] { 'a', 'b', 'a', 'a' }):
```

(S) - (2) - (3) - (4) - (5) - (6) - (7) - (8) - (10) - (7) - (8) - (10) - (12) - (22) - (E)

- (d) Im Falle des Kriteriums Pfadüberdeckung können minimale Testmengen sehr groß werden, da die Anzahl der Pfade sehr schnell zunimmt. Wie viele *mögliche Pfade* ergeben sich maximal für eine Schleife, die drei einseitig bedingte Anweisungen hintereinander enthält und bis zu zweimal durchlaufen wird? Geben Sie Ihren Rechenweg an (das Ergebnis alleine gibt keine Punkte).

Lösungsvorschlag

Pro Schleifendurchlauf:  $2 \cdot 2 \cdot 2 = 2^3 = 8$

Maximal 2 Schleifendurchläufe:  $2 \cdot 8 = 16$



- (e) Könnte für das hier abgebildete Quelltext-Beispiel auch das Verfahren der *unbegrenzten Pfadüberdeckung* (also Abdeckung aller möglicher Pfade ohne Beschränkung) als Test-Kriterium gewählt werden? Begründen Sie.

Lösungsvorschlag

Kante 7 nach 22

## 66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 5

- (a) Nennen Sie fünf kritische Faktoren, die bei der Auswahl eines Vorgehensmodells helfen können und ordnen Sie plangetriebene und agile Prozesse entsprechend ein.

Lösungsvorschlag

- (i) Vollständigkeit der Anforderungen
  - bei vollständiger Kenntnis der Anforderungen: *plangetrieben*
  - bei teilweiser Kenntnis der Anforderungen: *agil*
- (ii) Möglichkeit der Rücksprache mit dem Kunden
  - keine Möglichkeit: *plangetrieben*
  - Kunde ist partiell immer wieder involviert: *agil*
- (iii) Teamgröße
  - kleine Teams (max. 10 Personen): *agil*
  - größere Teams: *plangetrieben*
- (iv) Bisherige Arbeitsweise des Teams
  - bisher feste Vorgehensmodelle: *plangetrieben*
  - flexible Arbeitsweisen: *agil*
- (v) Verfügbare Zeit
  - kurze Zeitvorgabe: *plangetrieben*
  - möglichst schnell funktionierender Prototyp verlangt: *agil*
  - beide Vorgehensmodelle sind allerdings zeitlich festgelegt

Mögliche weitere Faktoren: Projektkomplexität, Dokumentation

- (b) Nennen und beschreiben Sie kurz die Rollen im Scrum.

Lösungsvorschlag

**Product Owner** Der Product Owner ist für die Eigenschaften und den wirtschaftlichen Erfolg des Produkts verantwortlich.

**Entwickler** Die Entwickler sind für die Lieferung der Produktfunktionalitäten in der vom Product Owner gewünschten Reihenfolge verantwortlich.

**Scrum Master** Der Scrum Master ist dafür verantwortlich, dass Scrum als Rahmenwerk gelingt. Dazu arbeitet er mit dem Entwicklungsteam zusammen, gehört aber selbst nicht dazu.

- (c) Nennen und beschreiben Sie drei Scrum Artefakte. Nennen Sie die verantwortliche Rolle für jedes Artefakt.

Lösungsvorschlag

**Product Backlog** Das Product Backlog ist eine geordnete Auflistung der Anforderungen an das Produkt.

**Sprint Backlog** Das Sprint Backlog ist der aktuelle Plan der für einen Sprint zu erledigenden Aufgaben.

**Product Increment** Das Inkrement ist die Summe aller Product-Backlog-Einträge, die während des aktuellen und allen vorangegangenen Sprints fertiggestellt wurden.

- (d) Beschreiben Sie kurz, was ein Sprint ist. Wie lange sollte ein Sprint maximal dauern?

Lösungsvorschlag

Ein Sprint ist ein Arbeitsabschnitt, in dem ein Inkrement einer Produktfunktionalität implementiert wird. Ein Sprint umfasst ein Zeitfenster von ein bis vier Wochen.

## 66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1

- (a) Nennen Sie fünf Phasen, die im Wasserfallmodell durchlaufen werden sowie deren jeweiliges Ziel bzw. Ergebnis(-dokument).

Lösungsvorschlag

**Anforderung** Lasten- und Pflichtenheft

**Entwurf** Softwarearchitektur in Form von Struktogrammen, UML-Diagrammen etc.

**Implementierung** Software

**Überprüfung** überarbeitete Software

**Einsatz und Wartung** erneut überarbeitete Software, verbesserte Wartung

- (b) Nennen Sie drei Arten der Softwarewartung und geben Sie jeweils eine kurze Beschreibung an.

Lösungsvorschlag

**korrektive Wartung** Korrektur von Fehlern, die schon beim Kunden in Erscheinung getreten sind

**präventive Wartung** Korrektur von Fehlern, die beim Kunden noch nicht in Erscheinung getreten sind

**adaptive Wartung** Anpassung der Software an neue Anforderungen

**perfektionierende Wartung** Verbesserung von Performance und Wartbarkeit

und Behebung von technical debts<sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/Softwarewartung>

Kontinuierliche Integration  
(Continuous Integration)  
EXtreme Programming  
Softwaremaße  
SCRUM

- (c) Eine grundlegende Komponente des *Extreme Programming* ist „*Continuous Integration*“. Erklären Sie diesen Begriff und warum man davon einen Vorteil erwartet.

Lösungsvorschlag

Die Software wird nach jeder Änderung (push) automatisch kompiliert und auch getestet. Man erwartet sich davon einen Vorteil, weil Fehler schneller erkannt werden und die aktuelle Version des Systems ständig verfügbar ist.

- (d) Nennen Sie zwei Softwaremetriken und geben Sie jeweils einen Vor- und Nachteil an, der im Projektmanagement von Bedeutung ist.

Lösungsvorschlag

#### Anzahl der Code-Zeilen

**Vorteil:** Ist sehr einfach zu bestimmen.

**Nachteil:** Entwickler können die Zeilenzahl manipulieren (zusätzliche Leerzeilen, Zeilen aufteilen), ohne dass sich die Qualität des Codes verändert.

#### Zyklomatische Komplexität

**Vorteil:** Trifft eine Aussage über die Qualität des Algorithmus.

**Nachteil:** Ist für Menschen nicht intuitiv zu erfassen. Zwei Codes, die dasselbe Problem lösen können die gleiche Zyklomatische Komplexität haben, obwohl der eine wesentlich schlechter zu verstehen ist (Spaghetticode!).

- (e) Nennen und beschreiben Sie kurz drei wichtige Aktivitäten, welche innerhalb einer Sprint-Iteration von Scrum durchlaufen werden.

Lösungsvorschlag

**Sprint Planning** Im Sprint Planning werden zwei Fragen beantwortet:

- Was kann im kommenden Sprint entwickelt werden?
- Wie wird die Arbeit im kommenden Sprint erledigt?

Die Sprint-Planung wird daher häufig in zwei Teile geteilt. Sie dauert in Summe maximal 2 Stunden je Sprint-Woche, beispielsweise maximal acht Stunden für einen 4-Wochen-Sprint.

**Daily Scrum** Zu Beginn eines jeden Arbeitstages trifft sich das Entwicklerteam zu einem max. 15-minütigen Daily Scrum. Zweck des Daily Scrum ist der Informationsaustausch.

**Sprint Review** Das Sprint Review steht am Ende des Sprints. Hier überprüft das Scrum-Team das Inkrement, um das Product Backlog bei Bedarf anzupassen. Das Entwicklungsteam präsentiert seine Ergebnisse und es wird überprüft, ob das zu Beginn gesteckte Ziel erreicht wurde.

- (f) Erläutern Sie den Unterschied zwischen dem Product-Backlog und dem Sprint-Backlog.

Lösungsvorschlag

**Product Backlog** Das Product Backlog ist eine geordnete Auflistung der Anforderungen an das Produkt.

**Sprint Backlog** Das Sprint Backlog ist der aktuelle Plan der für einen Sprint zu erledigenden Aufgaben.

- (g) Erläutern Sie, warum eine agile Entwicklungsmethode nur für kleinere Teams (max. 10 Personen) gut geeignet ist, und zeigen Sie einen Lösungsansatz, wie auch größere Firmen agile Methoden sinnvoll einsetzen können.

Lösungsvorschlag

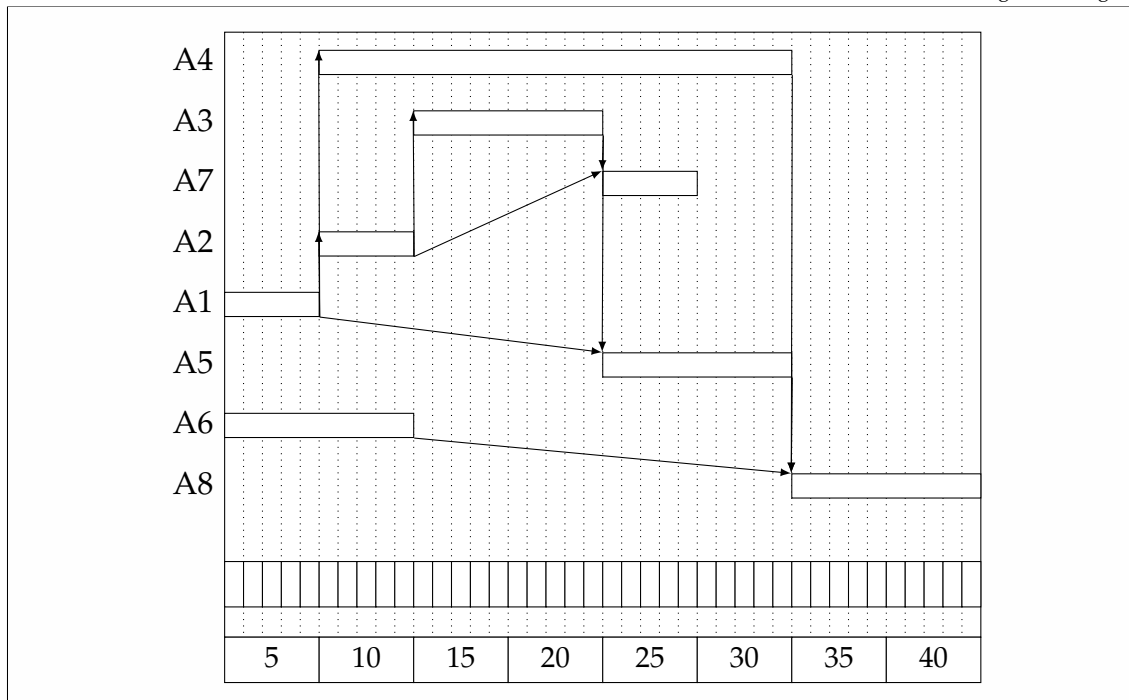
Bei agilen Entwicklungsmethoden finden daily scrums statt, die in der intendierten Kürze nur in kleinen Teams umsetzbar sind. Außerdem ist in größeren Teams oft eine Hierarchie vorhanden, die eine schnelle Entscheidungsfindung verhindert. Man könnte die große Firma in viele kleine Teams einteilen, die jeweils an kleinen (Teil-)Projekten arbeiten. Eventuell können die product owner der einzelnen Teams nach agilen Methoden zusammenarbeiten.

## 66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2

Betrachten Sie die folgende Tabelle zum Projektmanagement:

| Arbeitspaket | Dauer (Tage) | abhängig von |
|--------------|--------------|--------------|
| A1           | 5            |              |
| A2           | 5            | A1           |
| A3           | 10           | A2           |
| A4           | 25           | A1           |
| A5           | 10           | A1, A3       |
| A6           | 10           |              |
| A7           | 5            | A2, A3       |
| A8           | 10           | A4, A5, A6   |

- (a) Erstellen Sie ein Gantt-Diagramm, das die in der Tabelle angegebenen Abhängigkeiten berücksichtigt. Das Diagramm muss nicht maßstabsgetreu sein, jedoch jede Information aus der gegebenen Tabelle enthalten.



(b) Wie lange dauert das Projekt mindestens?

Lösungsvorschlag

Es dauert mindestens 40 Tage.

(c) Geben Sie alle kritischen Pfade an.

Lösungsvorschlag

A1 → A4 → A8

A1 → A2 → A3 → A5 → A8

(d) Bewerten Sie folgende Aussage eines Projektmanagers: „Falls unser Projekt in Verzug gerät, bringen uns neue Programmierer auch nicht weiter.“

Lösungsvorschlag

Ist der Verzug in einem Arbeitspaket, in dem genügend Pufferzeit vorhanden ist (hier A6), so hilft ein neuer Programmierer nicht unbedingt weiter. Geht allerdings die Pufferzeit zu Ende oder ist erst gar nicht vorhanden (z. B. im kritischen Pfad), so kann ein/e neue/r ProgrammierIn helfen, falls deren/dessen Einarbeitungszeit gering ist. Muss sich der/die Neue erste komplett einarbeiten, so wird er/sie wohl auch keine große Hilfe sein.

## 66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3

Im Folgenden ist eine Systembeschreibung für einen „Fahrkartenautomat“ angegeben.

Es gibt zwei Arten von Bahnfahrern: Gelegenheitsfahrer und Vielfahrer. Bei der Benutzung des Kartenautomaten kann sich grundsätzlich jeder die Hilfe anzeigen lassen und ein Beschwerdeformular ausfüllen.

Gelegenheitsfahrer haben keine Benutzer-Id und können Einzelfahrkarten auswählen und anschließend kaufen. Damit ein Kaufvorgang erfolgreich abgeschlossen wird, muss ein entsprechender Geldbetrag eingezahlt werden. Dies geschieht entweder mit Bargeld oder per EC-Karte. Nach dem Kauf eines Tickets kann man sich dafür optional eine separate Quittung drucken lassen.

Vielfahrer haben eine eindeutige Benutzer-Id mit Passwort, um sich am Automaten zu authentifizieren. Ein Vielfahrer kann sowohl eine Einzelfahrkarte als auch eine personalisierte Monatskarte erwerben. Sofern er eine Monatskarte besitzt (Information im System hinterlegt), kann er sich kostenfrei eine Ersatzfahrkarte ausstellen lassen, falls er seine Monatskarte verloren hat. Wenn die Authentifizierung oder ein Kaufvorgang fehlschlägt, soll eine entsprechende Fehlermeldung erscheinen.

- (a) Geben Sie die im Text erwähnten Akteure für das beschriebene System an.

Lösungsvorschlag

Bahnfahrer

- Gelegenheitsfahrer
- Vielfahrer

- (b) Identifizieren Sie zwei weitere Stakeholder und nennen Sie dazu je zwei unterschiedliche Anwendungsfälle des Systems, in die diese involviert sind.

Lösungsvorschlag

EC-Karte Bargeld

- (c) Geben Sie mindestens sechs verschiedene Anwendungsfälle für das beschriebene System an.

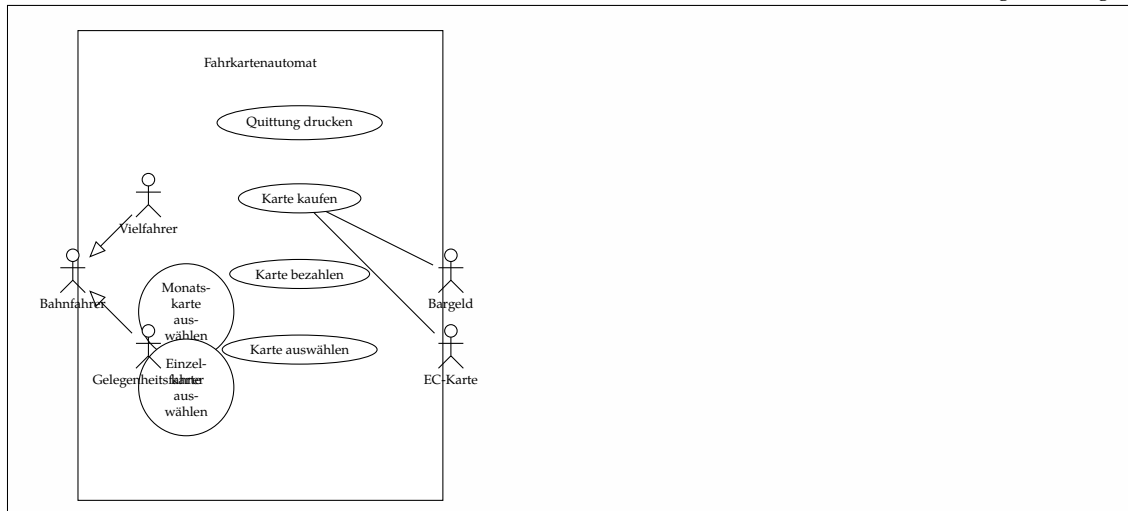
Lösungsvorschlag

- Hilfe anzeigen
- Beschwerdeformular ausfüllen
- Auswahl
  - Einzelfahrkarten
  - Monatskarte
- Einzelfahrkarten kaufen
- Zahlen
  - Bargeld
  - EC-Karte
- Quittung drucken
- Authentifizieren

- Bargeld
- EC-Karte

- (d) Erstellen Sie aus Ihren vorherigen Antworten ein Use-Case-Diagramm für das beschriebene System, in dem die Akteure und Anwendungsfälle inkl. möglicher Generalisierungen und Beziehungen eingetragen sind. Achten Sie insbesondere auf mögliche «include»- und «extends»-Beziehungen und Bedingungen für Anwendungsfälle.

Lösungsvorschlag



## 66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 4

Betrachten Sie das folgende Szenario:

Entwickeln Sie für einen Kunden eine einheitliche Online-Plattform, in welcher mehrere Restaurants angebunden sind. Die Nutzer des Systems sollen Essen wie Pizzen, Burger oder Pasta bestellen und dabei aus einer Liste von verschiedenen Gerichten auswählen können. Die Plattform soll zusätzliche Optionen (z. B. Lieferung durch einen Lieferdienst oder direkte Abholung, inkl. Salat oder einer Flasche Wein) ermöglichen. Die Bestellung soll dann von der Plattform an den jeweiligen Gaststättenbetreiber gesendet werden. Die Besteller sollen zudem eine Bestätigung als Nachricht erhalten. Die jeweiligen Gerichte und Optionen haben unterschiedliche Preise, die dem Internetnutzer angezeigt werden müssen, bevor er diese auswählt. Der Endpreis muss vor der endgültigen Zahlung des Auftrags angezeigt werden. Kunden können (optional) einen Benutzeraccount anlegen und erhalten bei häufigen Bestellungen einen Rabatt.

- (a) Beschreiben Sie kurz ein Verfahren, wie Sie aus der Szenariobeschreibung mögliche Kandidaten für Klassen erhalten können.

Lösungsvorschlag

Verfahren nach Abbott;<sup>a</sup>

Objektorientierte Analyse und Design (OOAD)

<sup>a</sup>[http://info.johpie.de/stufe\\_q1/info\\_01\\_verfahren\\_abbott.pdf](http://info.johpie.de/stufe_q1/info_01_verfahren_abbott.pdf)

- (b) Beschreiben Sie kurz ein Verfahren, um Vererbungshierarchien zu identifizieren.

Lösungsvorschlag

Eine Begriffshierarchie mit den Substantiven des Textes bilden.

- (c) Geben Sie fünf geeignete Klassen für das obige Szenario an. Nennen Sie dabei keine Klassen, welche durch Basisdatentypen wie Integer oder String abgedeckt werden können.

Lösungsvorschlag

- Benutzer (Kunde, Gaststättenbetreiber)
- Restaurant
- Gericht (Pizza, Burger, Pasta)
- ZusatzOption (Lieferung, Salat, Wein)
- Bestellung

- (d) Nennen Sie drei Klassen für das obige Szenario, die direkt durch Basisdatentypen wie Integer oder String abgedeckt werden können.

Lösungsvorschlag

- Nachricht
- Preis
- Rabatt

- (e) Erstellen Sie ein Sequenzdiagramm für das gegebene System mit folgendem Anwendungsfall: Ein Nutzer bestellt zwei Pizzen und eine Flasche Wein. Beginnen Sie mit der „Auswahl des Gerichts“ bis hin zur „Bestätigung der Bestellung“ sowie „Lieferung an die Haustür“. Das Diagramm soll mindestens je einen Akteur für Benutzer (Browser), Applikation (Webserver) und Restaurant (Koch und Liefersdienst) vorsehen. Die Bezahlung selbst muss nicht modelliert werden.

## 66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 5

Wir betrachten Terme über die Rechenarten  $op \in \{+, -, \cdot, \div\}$ , die rekursiv definiert sind:

- Jedes Literal ist ein Term, z. B. „4“.
- Jedes Symbol ist ein Term, z. B. „x“.
- Ist  $t$  ein Term, so ist „(t)“ ein (geklammerter) Term.
- Sind  $t_1, t_2$  Terme, so ist „ $t_1 \text{ op } t_2$ “ ebenso ein Term.

Beispiele für gültige Terme sind also „ $4 + 8$ “, „ $4 \cdot x$ “ oder „ $4 + (8 \cdot x)$ “.

- (a) Welches Design-Pattern eignet sich hier am besten zur Modellierung dieses Sachverhalts?



|            |
|------------|
| Kompositum |
|------------|

- (b) Nennen Sie drei wesentliche Vorteile von Design-Pattern im Allgemeinen.
- (c) Modellieren Sie eine Klassenstruktur in UML, die diese rekursive Struktur von *Termen* abbildet. Sehen Sie mindestens einzelne Klassen für die *Addition* und *Multiplikation* vor, sowie weitere Klassen für *geklammerte Terme* und *Literale*, welche ganze Zahlen repräsentieren. Gehen Sie bei der Modellierung der Klassenstruktur davon aus, dass eine objektorientierte Programmiersprache wie Java zu benutzen ist.
- (d) Erstellen Sie ein Objektdiagramm, welches den Term  $t := 4 + (3 \cdot 2) + (12 \cdot y / (8 \cdot x))$  entsprechend Ihres Klassendiagramms repräsentiert.
- (e) Überprüfen Sie, ob das Objektdiagramm für den in Teilaufgabe d) gegebenen Term eindeutig definiert ist. Begründen Sie Ihre Entscheidung.
- (f) Die gegebene Klassenstruktur soll mindestens folgende Operationen unterstützen:
- das Auswerten von Termen,
  - das Ausgeben in einer leserlichen Form,
  - das Auflisten aller verwendeten Symbole. Welches Design-Pattern ist hierfür am besten geeignet?
- (g) Erweitern Sie Ihre Klassenstruktur um die entsprechenden Methoden, Klassen und Assoziationen, um die in Teilaufgabe f) genannten zusätzlichen Operationen gemäß dem von Ihnen genannten Design Pattern zu unterstützen.

```
public class Addition extends Rechenart {
 public Addition(Term a, Term b) {
 super(a, b, "+");
 }

 public double auswerten() {
 return a.auswerten() + b.auswerten();
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Addition.java](https://github.com/org/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Addition.java)

```
public class Divison extends Rechenart {
 public Divison(Term a, Term b) {
 super(a, b, "/");
 }

 public double auswerten() {
 return a.auswerten() / b.auswerten();
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Divison.java](#)

```
public class GeklammerterTerm extends Term {
 Term term;

 public GeklammerterTerm(Term term) {
 this.term = term;
 }

 public double auswerten() {
 return term.auswerten();
 }

 public void ausgeben() {
 System.out.print("(");
 term.ausgeben();
 System.out.print(")");
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/GeklammerterTerm.java](#)

```
public class Literal extends Term {
 int wert;

 public Literal(int wert) {
 this.wert = wert;
 }

 public double auswerten() {
 return wert;
 }

 public void ausgeben() {
 System.out.print(wert);
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Literal.java](#)

```
public class Multiplikation extends Rechenart {
 public Multiplikation(Term a, Term b) {
 super(a, b, "*");
 }

 public double auswerten() {
 return a.auswerten() * b.auswerten();
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Multiplikation.java](#)

```
abstract class Rechenart extends Term {
 Term a;
 Term b;
```

```
String operatorZeichen;

public Rechenart (Term a, Term b, String operatorZeichen) {
 this.a = a;
 this.b = b;
 this.operatorZeichen = operatorZeichen;
}

public void ausgeben () {
 a.ausgeben();
 System.out.print(" " + operatorZeichen + " ");
 b.ausgeben();
}

abstract public double auswerten();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Rechenart.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Rechenart.java)

```
public class Subtraktion extends Rechenart {
 public Subtraktion(Term a, Term b) {
 super(a, b, "-");
 }

 public double auswerten() {
 return a.auswerten() - b.auswerten();
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Subtraktion.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Subtraktion.java)

```
public class Symbol extends Term {
 String name;
 public Symbol(String name) {
 this.name = name;
 }

 public double auswerten() {
 return Klient.symbol.get(name);
 }

 public void ausgeben() {
 System.out.print(name);
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Symbol.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Symbol.java)

```
abstract class Term {
 abstract public double auswerten();

 abstract public void ausgeben();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Term.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Term.java)

**66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 6**

Gegeben sei folgendes Java-Programm, das mit der Absicht geschrieben wurde, den größten gemeinsamen Teiler zweier Zahlen zu berechnen:

```
/** Return the Greatest Common Divisor of two integer values. */

public int gcd(int a, int b) {
 if (a < 0 || b < 0) {
 return gcd(-b, a);
 }
 while (a != b) {
 if (a > b) {
 a = a - b;
 } else {
 b = b % a;
 }
 }
 return 3;
}
```

- (a) Bestimmen Sie eine möglichst kleine Menge an Testfällen für das gegebene Programm, um (dennoch) vollständige Zweigüberdeckung zu haben.
- (b) Welche Testfälle sind notwendig, um die Testfälle der Zweigüberdeckung so zu erweitern, dass eine 100% Anweisungsüberdeckung erfüllt ist? Begründen Sie Ihre Entscheidung.
- (c) Beschreiben Sie zwei allgemeine Nachteile der Anweisungsüberdeckung.
- (d) Das gegebene Programm enthält (mindestens) zwei Fehler. Bestimmen Sie jeweils eine Eingabe, die fehlerhaftes Verhalten des Programms verursacht. Nennen Sie den Fault und den Failure, der bei der gewählten Eingabe vorliegt. Sie müssen das Programm (noch) nicht verbessern.
- (e) Geben Sie für die gefundenen zwei Fehler jeweils eine mögliche Verbesserung an. Es reicht eine textuelle Beschreibung, Code ist nicht notwendig.
- (f) Erläutern Sie, warum es im Allgemeinen hilfreich sein kann, bei der Fehlerbehebung in einem größeren Programm die Versionsgeschichte miteinzubeziehen.

**66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2**

Gegeben ist der folgende Ausschnitt eines Schemas für die Verwaltung eines Restaurants.

Hinweis: Unterstrichene Attribute sind Primärschlüsselattribute, kursiv geschriebene Attribute sind Fremdschlüsselattribute.

Restaurant : | RestaurantID : INTEGER, RestaurantName : VARCHAR (255), StadtName : VARCHAR(255),

PLZ: INTEGER, 1

Küche: |

RestaurantID : INTEGER, Art : VARCHAR(255), KochPersonID : INTEGER

Straße : VARCHAR (255), Hausnummer: INTEGER, Kategorie : VARCHAR (255)

1

Stadt : [ Person : |

StadtName : VARCHAR(255), PersonID : INTEGER,

Land : VARCHAR(255) Name : VARCHAR(255), 1 Vorname : VARCHAR (255),

StadtName : VARCHAR(255), PLZ: INTEGER, Straße : VARCHAR(255),

Hausnummer: INTEGER

bevorzugt : | PersonID : INTEGER, Art : VARCHAR(255)

I

Die Tabelle Restaurant beschreibt Restaurants eindeutig durch ihre ID. Zudem wird der Name, die Adresse des Restaurants und die (Sterne-)Kategorie gespeichert. Küche enthält u. a. Informationen zu der Art der Küche. Dabei kann ein Restaurant mehrere Arten anbieten, z. B. italienisch, deutsch, etc. In der Tabelle Stadt werden der Name der Stadt sowie das Land verwaltet, in dem die Stadt liegt. Wir gehen davon aus, dass eine Stadt eindeutig durch ihren Namen gekennzeichnet ist. Person beschreibt Personen mit Name, Vorname und Adresse. Personen werden eindeutig durch eine ID identifiziert. Die Tabelle bevorzugt gibt an, welche Person welche Art der Küche präferiert.

Bearbeiten Sie die folgenden Teilaufgaben:

- (a) Erläutern Sie kurz, warum das Attribut Art in Küche Teil des Primärschlüssels ist.

Lösungsvorschlag

Es kann mehr als eine Küche pro Restaurant geben.

- (b) Schreiben Sie eine SQL-Anweisung, welche alle Städte findet, in denen man “deutsch” (Art der Küche) essen kann.

Lösungsvorschlag

```
SELECT DISTINCT s.StadtName
FROM Stadt s, Restaurant r, Küche k
WHERE s.Stadtname = r.StadtName AND r.RestaurantID = k.RestaurantID AND Art =
→ 'deutsch';
```

- (c) Schreiben Sie eine SQL-Abfrage, die alle Personen (Name und Vorname) liefert, die kein deutsches Essen bevorzugen. Verwenden Sie keinen Join.

Lösungsvorschlag

```
SELECT Name, Vorname
FROM Person
WHERE PersonID IN (SELECT DISTINCT PersonID
FROM bevorzugt
EXCEPT
SELECT DISTINCT PersonID
FROM bevorzugt
WHERE Art = 'deutsch');
```

- (d) Schreiben Sie eine SQL-Abfrage, die für jede Stadt (StadtName) und Person (PersonID) die Anzahl der Restaurants ermittelt, in denen diese Person bevorzugt essen gehen würde. Es sollen nur Städte ausgegeben werden, in denen es mindestens drei solche Restaurants gibt.

Lösungsvorschlag

```
SELECT r.StadtName, b.PersonID, count(DISTINCT r.RestaurantID) as Anzahl
FROM Restaurant r, bevorzugt b, Kueche k
WHERE r.RestaurantID = k.RestaurantID AND k.Art = b.Art
GROUP BY r.StadtName, b.PersonID
HAVING count(r.RestaurantID) >= 3;
```

- (e) Schreiben Sie eine SQL-Abfrage, die die Namen aller Restaurants liefert, in denen sich die Personen mit den IDs 1 und 2 gemeinsam zum Essen verabreden können, und beide etwas zum Essen finden, das sie bevorzugen. Es sollen keine Duplikate ausgegeben werden.

Lösungsvorschlag

```
CREATE VIEW Person1 AS
SELECT DISTINCT r.RestaurantID, r.RestaurantName
FROM Person p, bevorzugt b, Restaurant r, Küche k
WHERE r.StadtName = p.StadtName AND p.PersonID = b.PersonID
AND r.RestaurantID = k.RestaurantID AND k.Art = b.Art
AND p.PersonID = 1;
CREATE VIEW Person2 AS
SELECT DISTINCT r.RestaurantID, r.RestaurantName
FROM Person p, bevorzugt b, Restaurant r, Küche k
WHERE r.StadtName = p.StadtName AND p.PersonID = b.PersonID
AND r.RestaurantID = k.RestaurantID AND k.Art = b.Art
AND p.PersonID = 2;
SELECT * FROM Person1
INTERSECT
SELECT * FROM Person2;
```

## 66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 3

```
CREATE TABLE V (
 Name VARCHAR(1),
 Jahr integer
);

CREATE TABLE S (
 Jahr integer
);

INSERT INTO V VALUES
('A', 2019),
('A', 2020),
('B', 2018),
('B', 2019),
```

```

('B', 2020),
('C', 2017),
('C', 2018),
('C', 2020);

INSERT INTO S VALUES
(2018),
(2019),
(2020);

```

- (a) Betrachten Sie die Relation  $V$ . Sie enthält eine Spalte *Name* sowie ein dazugehöriges Jahr.

| Name | Jahr |
|------|------|
| A    | 2019 |
| A    | 2020 |
| B    | 2018 |
| B    | 2019 |
| B    | 2020 |
| C    | 2017 |
| C    | 2018 |
| C    | 2020 |

- (i) Gesucht ist eine Relation  $S$ , die das folgende Ergebnis von  $V \div S$  berechnet ( $\div$  ist die Division der relationalen Algebra):
- $V \div S$

| Name |
|------|
| B    |

Welche der nachstehenden Ausprägungen für die Relation liefert das gewünschte Ergebnis? Geben Sie eine Begründung an.

|    |      |
|----|------|
|    | Jahr |
|    | 2017 |
| i. | 2018 |
|    | 2019 |
|    | 2020 |

|          |      |
|----------|------|
| ii.      | Jahr |
|          | 2018 |
|          | 2019 |
| iii.     | 2020 |
|          | Jahr |
|          | 2017 |
| iv. ii., | 2019 |
|          | 2020 |
|          |      |

Lösungsvorschlag

|                                        |      |
|----------------------------------------|------|
| iv) also weder i., noch ii., noch iii. |      |
| i.                                     | Name |
| ii.                                    | Name |
|                                        | C    |
| iii.                                   | Name |
|                                        | B    |
|                                        | C    |

(ii) Formulieren Sie die Divisions-Query aus Teilaufgabe i. in SQL.

```

SELECT DISTINCT v1.Name FROM V as v1
WHERE NOT EXISTS (
 (SELECT s.Jahr FROM S as s)
 EXCEPT
 (SELECT v2.Jahr FROM V as v2 WHERE v2.Name = v1.Name)
);
4

```

(b) Gegeben sind die Tabellen R(A, B) und S(C, D) sowie die folgende View:

```

1 CREATE VIEW mv (A,C,D) AS
, SELECT DISTINCT A,C,D
» FROM R,S
« WHERE B=D AND A <> 10;

```

Auf dieser View wird die folgende Query ausgeführt:

```

, SELECT DISTINCT A , FROM mv ;» WHERE C>D:

```

<sup>4</sup><https://www.geeksforgeeks.org/sql-division/>



Konvertieren Sie die Query und die zugrundeliegenden View in einen Ausdruck der relationalen Algebra in Form eines Operatorbaums. Führen Sie anschließend eine relationale Optimierung durch. Beschreiben und begründen Sie dabei kurz jeden durchgeführten Schritt.

- (c) Gegeben sind die Relationen R, S und U sowie deren Kardinalitäten Tr, Ts und Tr:

$$R(a_1, a_2, a_3) \quad Tr = 200 \quad S(a_1, a_2, a_3) \quad Ts = 100 \quad U(u_1, u_2) \quad Iv = 50$$

Bei der Ausführung des folgenden Query-Plans wurden die Kardinalitäten der Zwischenergebnisse mitgezählt und an den Kanten notiert.

Leiten Sie aus den Angaben im Ausführungsplan den Anteil der qualifizierten Tupel aller Prädikate her und geben Sie diese an.

$$Tx \text{ s0} | N \text{ Ral} > Vu$$

$$N \text{ R.a3} = S.a3 \cup N \text{ OR.a1} > 100 \text{ OS.a1} < 10$$

$$R \ 5$$

## 66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 4

Gegeben ist das folgende Relationenschema R in erster Normalform.

R:[A,B, C, D, E, F]

Für R gelte folgende Menge FD funktionaler Abhängigkeiten:

$$FA = \left\{ \begin{array}{l} \{AC\} \rightarrow \{DE\}, \\ \{ACE\} \rightarrow \{B\}, \\ \{E\} \rightarrow \{B\}, \\ \{D\} \rightarrow \{F\}, \\ \{AC\} \rightarrow \{F\}, \\ \{AD\} \rightarrow \{F\}, \end{array} \right\}$$

- (a) R mit FD hat genau einen Kandidatenschlüssel X. Bestimmen Sie diesen und begründen Sie Ihre Antwort.

Lösungsvorschlag

AC ist der Kandidatenschlüssel. AC kommt in keiner rechten Seite der funktionalen Abhängigkeiten vor.

- (b) Berechnen Sie Schritt für Schritt die Hülle  $X^+$  von  $X := \{K\}$ .

Lösungsvorschlag

- (i)  $AC \cup DE$
- (ii)  $ACDE \cup B \ (ACE \rightarrow B)$
- (iii)  $ACDEB \ (E \rightarrow B)$
- (iv)  $ACDEB \cup F \ (D \rightarrow F)$

- (v)  $ACDEBF$  ( $AC \rightarrow F$ )  
 (vi)  $ACDEBF$  ( $AD \rightarrow F$ )

Sichtbarkeit

- (c) Nennen Sie alle primen und nicht-primen Attribute.

Lösungsvorschlag

prim: AC  
 nicht-prim: BDEF

- (d) Geben Sie die höchste Normalform an, in der sich die Relation befindet. Begründen Sie.

Lösungsvorschlag

2NF  
 $D \rightarrow F$  hängt transitiv von AC ab:  $AC \rightarrow D, D \rightarrow F$

- (e) Gegeben ist die folgende Zerlegung von R:

R1 (A, C, D, E) R2 (B, E) R3 (D, F)

Weisen Sie nach, dass es sich um eine verlustfreie Zerlegung handelt.

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1

- (a) Definieren Sie die Bedeutung der Sichtbarkeiten *private*, *package-private* und *protected* von Feldern in Java-Klassen. Erklären Sie diese kurz (je ein Satz).

Lösungsvorschlag

**private** Das Feld ist nur innerhalb der eigenen Klasse zugreifbar.

**package-private** Bei *package private* (kein *modifier*) wird die Sichtbarkeit auf das Paket, dem die Klasse angehört, ausgedehnt (<https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html>).

**protected** Die Sichtbarkeit wird gegenüber *package-private* auf alle Subklassen (auch Subklassen, die nicht dem Paket angehören) ausgedehnt.

| Modifier           | Class | Package | Sub (same pkg) | Sub (diff pkg) | World |
|--------------------|-------|---------|----------------|----------------|-------|
| public             | ja    | ja      | ja             | ja             | ja    |
| protected          | ja    | ja      | ja             | ja             | nein  |
| <i>no modifier</i> | ja    | ja      | ja             | nein           | nein  |
| private            | ja    | nein    | nein           | nein           | nein  |

<https://stackoverflow.com/a/215505>

- (b) Benennen Sie jeweils einen Grund für den Einsatz der Sichtbarkeiten *private*, *package-private* und *protected* von Feldern in Java-Klassen.

**private** Datenkapselung, Verbergen der internen Implementation. So kann die interne Implementation geändert werden, ohne dass die öffentlichen Schnittstellen sich ändern.

**package-private** Um die Felder zwar innerhalb deines Pakets für alle Klassen zugänglich zu machen aber nicht außerhalb, z. B. in einem größeren Projekt.

**protected** Wenn eine Klasse vererbt werden soll, z. B. abstrakte Klassen.

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Lesen Sie die folgenden Beispielscodes gründlich. Identifizieren Sie für jeden Beispielscode den jeweiligen wesentlichen Verstoß gegen die Prinzipien guter objektorientierter Programmierung. Benennen und erklären Sie jeweils den Verstoß (Fehler) in einem Satz und erläutern Sie für jeden Beispielscode, welche Probleme aus dem jeweiligen Fehler resultieren können, ebenfalls in einem Satz.

```
(a) class Rectangle {
 private int width;
 private int length;

 Rectangle(int w, int l) {
 width = w;
 length = l;
 }

 public int getWidth() {
 return this.width;
 }

 public int getLength() {
 return this.length;
 }
}

class RectangleDemo {
 public static void main(String args[]) {
 Rectangle rectangle1 = new Rectangle(10, 20);
 Rectangle rectangle2 = new Rectangle(3, 90);
 Rectangle example = new Rectangle(1, 2);
 int area;

 // Compute area of first box
 area = rectangle1.getWidth() * rectangle1.getLength();
 System.out.println("Area is " + area);

 // Compute area for second box
 area = rectangle2.getWidth() * rectangle2.getLength();
 System.out.println("Area is " + area);

 // Compute area for third box
```

```
 area = example.getWidth() * example.getLength();
 System.out.println("Area is " + area);
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/Rectangle.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Rectangle.java)

#### Lösungsvorschlag

Es sollte die Methode computeArea in der Klasse Rectangel implementiert werden.

```
(b) public class CalculateSpeed {
 private double kilometers;
 private double minutes;

 public CalculateSpeed(double k, double m) {
 this.kilometers = k;
 this.minutes = m;
 }

 // Display the speed
 void speed() {
 double speed;
 speed = kilometers / (minutes / 60);
 System.out.println("A car traveling " + kilometers + " kilometers in " +
 ↳ minutes + " minutes travels at " + speed
 + " kilometers per hour");
 }

 public static void main(String args[]) {
 CalculateSpeed car = new CalculateSpeed(110.0, 120.0);

 // Display car speed
 car.speed();

 // Display bicycle speed
 double speed;

 speed = 20.0 / (80.0 / 60);
 // So steht es in der Angabe
 // System.out.println("A bicycle traveling " + kilometers + " kilometers in
 ↳ " + minutes + " minutes travels at "
 // + speed + " kilometers per hour");
 // Ohne Fehler:
 System.out.println("A bicycle traveling " + car.kilometers +
 ↳ " kilometers in " + car.minutes + " minutes travels at "
 + speed + " kilometers per hour");
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/CalculateSpeed.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/CalculateSpeed.java)

Klassen sollten nach Objekten modelliert werden und nicht nach Tätigkeiten (berechne Geschwindigkeit) Besser wäre der Name SpeedCalculator gewesen. Außerdem sind die beiden Attribute kilometer und miutes in der Main Methode so nicht ansprechbar, weil sie nicht statisch sind.

```
(c) class Stack {

 int stck[] = new int[3];
 public int top;

 // Initialize top of stack
 Stack() {
 top = -1;
 }

 // Push an item on the stack
 void push(int item) {
 if (top == 2) {
 System.out.println("Stack is full.");
 } else {
 stck[++top] = item;
 }
 }

 // Pop an item from the stack
 int pop() {
 if (top < 0) {
 throw new IllegalStateException("Stack is empty.");
 } else {
 return stck[top--];
 }
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/Stack.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Stack.java)

Das Feld stck sollte private sein. So wird die interne Implementetation verborgen.

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Wählen Sie bis zu fünf unterschiedliche Elementtypen aus folgendem Diagramm aus und benennen Sie diese Elemente und ihre syntaktische (nicht semantische) Bedeutung.

**Klasse** ConcreteCreator

**Interface** Produkt

**Abstrakte Klasse** Creator

**Kommentar** return new ConcreteProdukt()**66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 4**

(a) Schreiben Sie ein Programm in einer objektorientierten Programmiersprache Ihrer Wahl, das den folgenden Anweisungen entspricht.

- (i) Es gibt eine Klasse mit dem Namen `jBox`.
- (ii) Alle Zahlen sind Fließkommazahlen.
- (iii) `jBox` wird mit einem Argument instanziiert, dessen Wert einer Variable namens `jlength` zugewiesen wird.

Lösungsvorschlag

```
public Box(double length) {
 this.length = length;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/Box.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Box.java)

(iv) `jBox` hat eine Methode ohne Argumente namens `jsize`, welche den Wert von `jlength` zurückgibt.

Lösungsvorschlag

```
public double size() {
 return length;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/Box.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Box.java)

(v) Eine weitere Methode namens `jsize` hat genau ein Argument namens `jwidth`. Diese zweite Methode namens `jsize` gibt das Produkt aus `jwidth` und `jlength` zurück. Eine weitere Methode namens `jsize` hat genau zwei Argumente, nämlich eine Zahl `jnum` und einen Faktor `jf`. Es wird `jlength` minus das Produkt aus `jnum` und `jf` zurückgegeben.

Lösungsvorschlag

```
public double size(double width) {
 return this.length * width;
}

public double size(double num, double f) {
 return this.length - num * f;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/Box.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Box.java)

- (vi) Schreiben Sie eine `jmain`-Methode, die eine `Box` namens `jexample` mit einer Länge von 15 anlegt.
- (vii) Führen Sie die Methode `jsize` in der `jmain`-Methode wie unten angegeben drei Mal aus.

(viii) Speichern Sie hierbei das Ergebnis jeweils in einer Variable `mysize`. Geben Sie das Ergebnis jeweils in einer eigenen Zeile des Ausgabemediums `System.out` aus.

- Mit keinen Argumenten
- Mit dem Argument `j10`
- Mit den beiden Argumenten `j5` und `j2`

Lösungsvorschlag

```
public static void main(String[] args) {
 Box example = new Box(15);

 double mysize = example.size();
 System.out.println(mysize);

 mysize = example.size(10);
 System.out.println(mysize);

 mysize = example.size(5, 2);
 System.out.println(mysize);
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/Box.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Box.java)

```
public class Box {
 double length;

 public Box(double length) {
 this.length = length;
 }

 public double size() {
 return length;
 }

 public double size(double width) {
 return this.length * width;
 }

 public double size(double num, double f) {
 return this.length - num * f;
 }

 public static void main(String[] args) {
 Box example = new Box(15);

 double mysize = example.size();
 System.out.println(mysize);

 mysize = example.size(10);
 System.out.println(mysize);

 mysize = example.size(5, 2);
 }
}
```



```

 System.out.println(mysize);
 }

}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/Box.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/Box.java)

Einfach-verkettete Liste

(b) Notieren Sie die Ausgabe der jmain-Methode.

Lösungsvorschlag

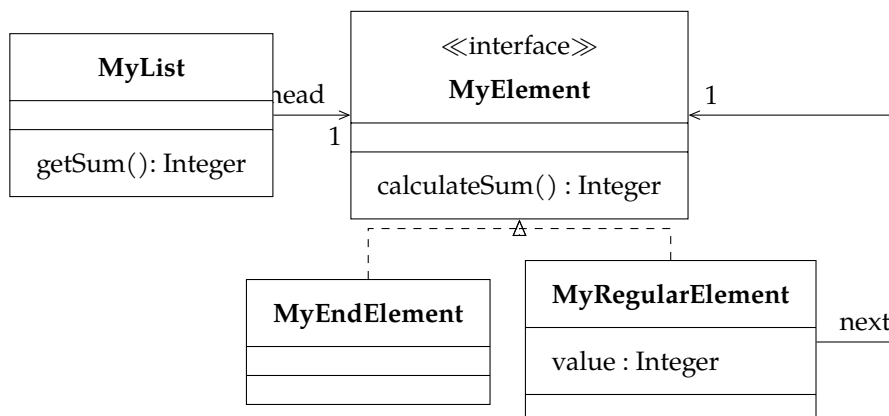
```

- 15
- 150
- 5

```

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 5

Die folgende Abbildung stellt den Entwurf der Implementierung einer verketteten Liste dar, welche Integer-Werte als Elemente enthalten kann.



Die Klasse `MyList` stellt die Methode `getSum()` zur Verfügung, welche die Summe über alle in einer Liste befindlichen Elemente berechnet. Ein Ausschnitt der Implementierung sieht folgendermaßen aus:

```

public class MyList {
 private MyElement head;

 public MyList() {
 this.head = new MyEndElement();
 }

 public int getSum() {
 // ..
 }
}

```

Gehen Sie im Folgenden davon aus, dass bereits Methoden existieren, welche Elemente in die Liste einfügen können.

- (a) Implementieren Sie in einer objektorientierten Programmiersprache Ihrer Wahl, z. B. Java, die Methode `calculateSum()` der Klassen `MyEndElement` und `MyRegularElement`, so dass rekursiv die Summe der Elemente der Liste berechnet wird.

Als Abbruchbedingung darf hierbei nicht das Feld `MyRegularElement.next` auf den Wert `null` überprüft werden.

Hinweis: Gehen Sie davon aus, die Implementierung von `MyList` garantiert, dass `MyRegularElement.next` niemals den Wert `null` annimmt, sondern das letzte hinzugefügte `MyRegularElement` auf eine Instanz der Klasse `MyEndElement` verweist. Es gibt immer nur eine Instanz der Klasse `MyEndElement` in einer Liste.

Hinweis: Achten Sie auf die Angabe einer korrekten Methodensignatur.

Lösungsvorschlag

```
int calculateSum() {
 return value + next.calculateSum();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/my\\_list/MyElement.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/my_list/MyElement.java)

```
int calculateSum() {
 return 0;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/my\\_list/MyEndElement.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/my_list/MyEndElement.java)

- (b) Nennen Sie den Namen des Entwurfsmusters, auf welchem das oben gegebene Klassendiagramm basiert, und ordnen Sie dieses in eine der Kategorien von Entwurfsmustern ein.

Hinweis: Es genügt die Angabe eines Musters, falls Sie mehrere Muster identifizieren sollten.

Lösungsvorschlag

Kompositum (Strukturmuster)

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 6

- (a) Erläutern Sie den Zweck (Intent) des Erzeugungsmusters Erbauer in max. drei Sätzen, ohne dabei auf die technische Umsetzung einzugehen.

Lösungsvorschlag

Die Erzeugung komplexer Objekte wird vereinfacht, indem der Konstruktionsprozess in eine spezielle Klasse verlagert wird. Er wird so von der Repräsentation getrennt und kann sehr unterschiedliche Repräsentationen zurückliefern.

- (b) Erklären Sie, wie das Erzeugungsmuster Erbauer umgesetzt werden kann (Implementierung). Die Angabe von Code ist hierbei NICHT notwendig!

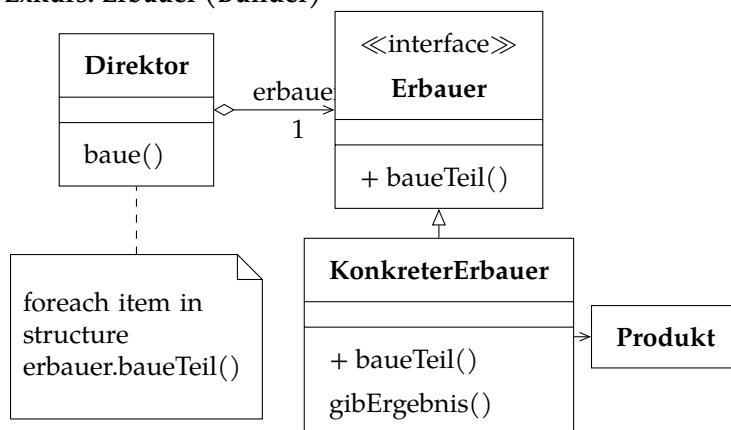
- (c) Nennen Sie jeweils einen Vor- und einen Nachteil des Erzeugungsmusters Erbauer im Vergleich zu einer Implementierung ohne dieses Muster.

Lösungsvorschlag

**Vorteil** Die Implementierungen der Konstruktion und der Repräsentationen werden isoliert. Die Erbauer verstecken ihre interne Repräsentation vor dem Direktor.

**Nachteil** Es besteht eine enge Kopplung zwischen Produkt, konkretem Erbauer und den am Konstruktionsprozess beteiligten Klassen.

### Exkurs: Erbauer (Builder)



**Erbauer** Der Erbauer spezifiziert eine abstrakte Schnittstelle zur Erzeugung der Teile eines komplexen Objektes.

**KonkreterErbauer** Der konkrete Erbauer erzeugt die Teile des komplexen Objekts durch Implementierung der Schnittstelle. Außerdem definiert und verwaltet er die von ihm erzeugte Repräsentation des Produkts. Er bietet auch eine Schnittstelle zum Auslesen des Produkts.

**Direktor** Der Direktor konstruiert ein komplexes Objekt unter Verwendung der Schnittstelle des Erbauers. Der Direktor arbeitet eng mit dem Erbauer zusammen: Er weiß, welche Baureihenfolge der Erbauer verträgt oder benötigt. Der Direktor entkoppelt somit den Konstruktionsablauf vom Klienten.

**Produkt** Das Produkt repräsentiert das zu konstruierende komplexe Objekt.

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 7

Lesen Sie die folgenden alternativen Codestücke.

(a)

```

public class MyParser {
 private InputStream input;

 public MyParser(String filePath) {

 }
}

```

(b)

```
public class MyParser {
 private InputStream input;

 public MyParser(InputStream stream) {
 }
}
```

Beide Codestücke zeigen die Initialisierung einer Klasse namens `MyParser`. Das zweite Codestück nutzt jedoch hierfür eine Technik namens Abhängigkeits-Injektion (Dependency Injection).

- (a) Erklären Sie den Unterschied zwischen beiden Initialisierungen. Hinweis: Sie können diese Aufgabe auch lösen, falls Sie die Technik nicht kennen.

Lösungsvorschlag

Die Abhängigkeit von einer Instanz der Klasse `InputStream` wird erst bei der Initialisierung des Objekt übergeben.

- (b) Benennen Sie einen Vorteil dieser Technik.

Lösungsvorschlag

Die Kopplung zwischen einer Klasse und ihrer Abhängigkeit wird verringert.

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 8

Das Client-Server-Modell ist ein Architekturmuster. Nennen Sie zwei Vorteile einer nach diesem Muster gestalteten Architektur.

Lösungsvorschlag

- (a) Einfache Integration weiterer Clients
- (b) Prinzipiell uneingeschränkte Anzahl der Clients <sup>a</sup>
- (c) Es muss nur ein Server gewartet werden. Dies gilt z. B. für Updates, die einmalig und zentral auf dem Server durchgeführt werden und danach für alle Clients verfügbar sind. <sup>b</sup>

<sup>a</sup><https://www.karteikarte.com/card/164928/vorteile-und-nachteile-des-client-server-modells>

<sup>b</sup><https://www.eoda.de/wissen/blog/client-server-architekturen-performance-und-agilitaet-fuer-data-s>

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 9

Betrachten Sie die folgende Liste von Technologien:

- Nodejs
- PHP
- CSS

- AJAX
- Python
- Java

Welche dieser Technologien laufen in einem Client-Server-System üblicherweise auf der Seite des Klienten und welche auf der Seite des Servers? Nehmen Sie hierzu an, dass der Client ein Browser ist.

Übertragen Sie die im Folgenden gegebene Tabelle in Ihren Bearbeitungsbogen und ordnen Sie die aufgelisteten Technologien anhand der Buchstaben in die Tabelle ein. Fortsetzung nächste Seite!

Hinweis: Mehrfachzuordnungen sind möglich.

Lösungsvorschlag

| Client-seitige Technologien | Server-seitige Technologien |
|-----------------------------|-----------------------------|
| CSS                         | Nodejs                      |
| AJAX                        | PHP                         |
| Python                      | Python                      |
| Java                        | Java                        |

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 10

- (a) Was bedeutet die Abkürzung AJAX?

Lösungsvorschlag

Asynchronous JavaScript and XML

- (b) Erklären Sie in max. drei Sätzen die grundlegende Funktion von AJAX.

Lösungsvorschlag

Konzept der asynchronen Datenübertragung zwischen einem Browser und dem Server. Dieses ermöglicht es, HTTP-Anfragen durchzuführen, während eine HTML-Seite angezeigt wird, und die Seite zu verändern, ohne sie komplett neu zu laden.<sup>a</sup>

<sup>a</sup>[https://de.wikipedia.org/wiki/Ajax\\_\(Programmierung\)](https://de.wikipedia.org/wiki/Ajax_(Programmierung))

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 11

- (a) Was ist das Hypertext Transfer Protocol (HTTP) und wozu dient es?

Lösungsvorschlag

Das Hypertext Transfer Protocol ist ein zustandsloses Protokoll zur Übertragung von Daten auf der Anwendungsschicht über ein Rechnernetz. Es wird hauptsächlich eingesetzt, um Webseiten (Hypertext-Dokumente) aus dem

World Wide Web (WWW) in einen Webbrowser zu laden. Es ist jedoch nicht prinzipiell darauf beschränkt und auch als allgemeines Dateiübertragungsprotokoll sehr verbreitet.<sup>a</sup>

<sup>a</sup>[https://de.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](https://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol)

- (b) Betrachten Sie die folgende Zeile Text. Um welche Art von Text handelt es sich?

`https://developer.mozilla.org/en-US/search?q=client+servertooverview`

Lösungsvorschlag

Es handelt sich um eine HTTP-URL (Uniform Resource Locator). Die URL lokalisiert eine Ressource, beispielsweise eine Webseite, über die zu verwendende Zugriffsmethode (zum Beispiel das verwendete Netzwerkprotokoll wie HTTP oder FTP) und den Ort (engl. location) der Ressource in Computernetzwerken.<sup>a</sup>

<sup>a</sup>[https://de.wikipedia.org/wiki/Uniform\\_Resource\\_Locator](https://de.wikipedia.org/wiki/Uniform_Resource_Locator)

- (c) Was sind die vier wesentlichen Bestandteile des Texts aus der vorigen Teilaufgabe?

Lösungsvorschlag

**Schema** `https://`  
**Host** `developer.mozilla.org`  
**Pfad** `/en-US/search`  
**Query** `?q=client+servertooverview`

<sup>a</sup>

<sup>a</sup>[https://de.wikipedia.org/wiki/Uniform\\_Resource\\_Locator](https://de.wikipedia.org/wiki/Uniform_Resource_Locator)

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 12

Es gibt Softwaresysteme, welche auf peer-to-peer (P2P) Kommunikation basieren und eine entsprechende Architektur aufweisen.

- (a) Bewerten Sie die folgenden Aussagen als entweder richtig oder falsch.

- (i) Mithilfe des Befehls "lookup" können Peers sich gegenseitig identifizieren.

Lösungsvorschlag

richtig

- (ii) In einem P2P-System, wie auch bei Client-Server, sind alle Netzwerkteilnehmer gleichberechtigt.

Lösungsvorschlag

falsch. Im Client-Servermodell sind nicht alle Netzwerkteilnehmer gleichberechtigt. Der Server hat mehr Privilegien wie der Client.

- (iii) Alle P2P-Systeme funktionieren grundsätzlich ohne einen zentralen Verwaltungs-Peer.

Lösungsvorschlag

falsch. Es gibt zentralisierte P2P-Systeme (Beispiel: Napster), welche einen zentralen Server zur Verwaltung benötigen, um zu funktionieren.

<sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/Peer-to-Peer>

- (iv) P2P kann auch für eine Rechner-Rechner-Verbindung stehen.

Lösungsvorschlag

richtig

- (v) Es gibt strukturierte und unstrukturierte P2P-Systeme. In unstrukturierten P2P-Systemen wird zum Auffinden von Peers eine verteilte Hashtabelle verwendet (DHT).

Lösungsvorschlag

richtig

- (vi) In einem P2P-System sind theoretisch alle Peers gleichberechtigt, praktisch gibt es jedoch leistungsabhängige Gruppierungen.

Lösungsvorschlag

richtig

- (vii) Ein Peer kann sowohl ein Client wie auch ein Server für einen anderen Peer sein.

Lösungsvorschlag

richtig

- (b) Wählen Sie zwei falsche Aussagen aus der vorherigen Tabelle aus und berichtigen Sie diese in jeweils einem Satz.

Lösungsvorschlag

Sie oben.

**66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 1**

Beantworten Sie die folgenden Fragen und begründen oder erläutern Sie Ihre Antwort.

- (a) Erläutern Sie die Begriffe Kardinalität und Partizipität. Welche Arten von Partizipität gibt es in der ER-Modellierung? Nennen und erklären Sie diese kurz.

Lösungsvorschlag

**Kardinalitäten** Für die noch genauere Darstellung der Beziehungen im ER-Modell verwendet man Kardinalitäten (auch Grad der Beziehungen genannt). Diese geben an wie viele Entitätsinstanzen mit wie vielen Entitätsinstanzen einer anderen Entitätsinstanz in Beziehung stehen. <sup>a</sup>

**Partizipation** Die Partizipation eines Beziehungstyps (in einem Entity-Relationship-Modell) bestimmt, ob alle Entities eines beteiligten Entitätstyps an einer bestimmten Beziehung teilnehmen müssen. <sup>b</sup>

**totale Partizipation:** Wenn eine Beziehung Entität A und Entität B in Beziehung setzt, dann muss ein Eintrag in Entität A existieren, damit ein Eintrag in Entität B existiert und umgekehrt. Beide Entitäten müssen also an der Relation teilnehmen. Eine Entitätsinstanz aus A kann also nicht ohne eine in-Beziehung-stehende Entitätsinstanz aus B existieren und umgekehrt.

**partielle Partizipation:** Wenn eine Beziehung Entität A mit Entität B in Beziehung setzt, dann muss kein Eintrag in Entität A existieren, damit ein Eintrag in Entität B existieren kann und umgekehrt. Die beiden Entitäten müssen also nicht an der Relation teilnehmen (enthalten sein). <sup>c</sup>

Die Kardinalität definiert, wie viele Entities eines Typs mit wie vielen Entities eines anderen Typs in Beziehung stehen können (siehe Schneider et al., S. 446)

Partizipität – ein anderer Begriff dafür ist Totalität – beschreibt den Grad der Teilnahmeverpflichtung zweier Entitäten an einer Beziehung. Sie kann partiell, total oder einseitig total sein.

a. Totale P.: Jede Entity A und Entity B besteht nur dann, wenn sie an dieser Beziehung teilnehmen.

b. Einseitige totale P.: Eine Entity A besteht nur dann, wenn sie an der Beziehung zu Entity B teilnimmt. Entitäten von B dagegen können, müssen aber nicht teilnehmen.

c. Partielle P.: Die Existenz der Entitäten ist unabhängig von der Teilnahme an dieser Beziehung. Die Teilnahme ist nicht verpflichtend.

<sup>a</sup><https://usehardware.de/datenbanksysteme-iv-entity-relationship-modell-er-modell-datenbankda>

<sup>b</sup><https://lehrbuch-wirtschaftsinformatik.org/glossar/kapitel03/>

Partizipation

<sup>c</sup><https://usehardware.de/datenbanksysteme-iv-entity-relationship-modell-er-modell-datenbankda>



- (b) Mit welchen beiden Befehlen kann eine Transaktion beendet werden? Nennen Sie diese und erklären Sie den Unterschied.

Lösungsvorschlag

Für den Abschluss einer Transaktion gibt es 2 Möglichkeiten:

- Den erfolgreichen Abschluss mit commit.
- Den erfolglosen Abschluss mit abort

- (c) Erläutern Sie den Unterschied zwischen einer kurzen und einer langen Sperre.

Lösungsvorschlag

**lange Sperren:** LOCKs werden erst nach dem commit zurückgegeben (→ striktes 2PL)

**kurze Sperren:** LOCKs werden direkt nach dem schreiben/lesen zurückgegeben

<sup>a</sup>

<sup>a</sup><https://www.dbs.ifi.lmu.de/Lehre/DBSII/SS2015/vorlesung/DBS2-03-Synchronisation.pdf>

- (d) Stellen Sie außerdem die Kompatibilitätsmatrix zur Umsetzung des ACID-Prinzips mit den richtigen Werten dar. S stehe dabei für eine Lese- und X für eine Schreibsperre.

Lösungsvorschlag

Kompatibilitätsmatrix zur Umsetzung des ACID-Prinzips (Atomicity, Consistency, Isolation, Durability)

|                   | NL (no lock) | S (Lesesperre) | X (Schreibsperre) |
|-------------------|--------------|----------------|-------------------|
| S (Lesesperre)    | ja           | ja             | nein              |
| X (Schreibsperre) | ja           | nein           | nein              |

- (e) Nennen und erklären Sie kurz die Armstrong-Axiome. Sind diese vollständig und korrekt?

Lösungsvorschlag

**Reflexivität:** Eine Menge von Attributen bestimmt eindeutig die Werte einer Teilmenge dieser Attribute (triviale Abhängigkeit), das heißt,  $\beta \subseteq \alpha \Rightarrow \alpha \rightarrow \beta$ .

**Verstärkung:** Gilt  $\alpha \rightarrow \beta$ , so gilt auch  $\alpha\gamma \rightarrow \beta\gamma$  für jede Menge von Attributen  $\gamma$  der Relation.

**Transitivität:** Gilt  $\alpha \rightarrow \beta$  und  $\beta \rightarrow \gamma$ , so gilt auch  $\alpha \rightarrow \gamma$ .

Die Armstrong-Axiome sind korrekt und vollständig: Diese Regeln sind gültig (korrekt) und alle anderen gültigen Regeln können von diesen Regeln ab-

geleitet werden (vollständig).<sup>a</sup>

<sup>a</sup>[https://dbresearch.uni-salzburg.at/teaching/2019ss/db1/db1\\_06-handout-1x1.pdf](https://dbresearch.uni-salzburg.at/teaching/2019ss/db1/db1_06-handout-1x1.pdf)

- (f) Was versteht man unter einem (Daten-)Katalog (Data Dictionary) und was enthält dieser (es genügt eine Auswahl zu nennen)?

Lösungsvorschlag

Bei einer relationalen Datenbank ist ein Datenkatalog eine Menge von Tabellen und Ansichten, die bei Abfragen nur gelesen werden. Das Data-Dictionary ist wie eine Datenbank aufgebaut, enthält aber nicht Anwendungsdaten, sondern Metadaten, das heißt Daten, welche die Struktur der Anwendungsdaten beschreiben (und nicht den Inhalt selbst).

Zu einem Data-Dictionary zur physischen Datenmodellierung gehören genaue Angaben zu:

- Tabellen und Datenfeldern
- Primär- und Fremdschlüsselbeziehungen
- Integritätsbedingungen, z. B. Prüfinformationen

<sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/Data-Dictionary>

- (g) Erklären Sie das konservative und das strikte Zwei-Phasen-Sperrprotokoll.

Lösungsvorschlag

**Konservatives 2-Phasen-Sperrprotokoll** Das konservative 2-Phasen-Sperrprotokoll (Preclaiming), bei welchem zu Beginn der Transaktion alle benötigten Sperren auf einmal gesetzt werden. Dies verhindert in jedem Fall Deadlocks, führt aber auch zu einem hohen Verlust an Parallelität, da eine Transaktion ihre erste Operation erst dann ausführen kann, wenn sie alle Sperren erhalten hat.

**Striktes 2-Phasen-Sperrprotokoll** Das strikte 2-Phasen-Sperrprotokoll, bei welchem alle gesetzten Write-Locks erst am Ende der Transaktion (nach der letzten Operation) freigegeben werden. Dieses Vorgehen verhindert den Schneeballeffekt, also das kaskadierende Zurücksetzen von sich gegenseitig beeinflussenden Transaktionen. Der Nachteil ist, dass Sperren häufig viel länger gehalten werden als nötig und sich somit die Wartezeit von blockierten Transaktionen verlängert. Die Read-Locks werden entsprechend dem Standard-2PL-Verfahren entfernt.

- (h) Erklären Sie die Begriffe „Steal/NoSteal“ und „Force/NoForce“ im Kontext der Systempufferverwaltung eines DBS.

**No-Steal** Schmutzige Seiten dürfen nicht aus dem Puffer entfernt und in die Datenbank übertragen werden, solange die Transaktion noch aktiv ist. Die Datenbank enthält keine Änderungen nicht-erfolgreicher Transaktionen. Eine UNDO-Recovery ist nicht erforderlich. langen Änderungs-Transaktionen können zu Problemen führen, da große Teile des Puffers blockiert werden

**Steal** Schmutzige Seiten dürfen jederzeit ersetzt und in die Datenbank eingebracht werden. Die Datenbank kann unbestätigte Änderungen enthalten. Eine UNDO-Recovery ist erforderlich. Es handelt sich um eine effektivere Puffernutzung bei langen Transaktionen mit vielen Änderungen.

**Force** Alle geänderten Seiten werden spätestens bei EOT (vor COMMIT) in die Datenbank geschrieben. Bei einem Systemfehler ist keine REDO-Recovery erforderlich. Die Force-Strategie benötigt einen hohen I/O-Aufwand, da Änderungen jeder Transaktion einzeln geschrieben werden. Die Vielzahl an Schreibvorgängen führt zu schlechteren Antwortzeiten, länger gehaltenen Sperren und damit zu mehr Sperrkonflikten. Große Datenbank-Puffer werden schlecht genutzt.

**No-Force** Änderungen können auch erst nach dem COMMIT in die Datenbank geschrieben werden. Die Änderungen durch mehrere Transaktionen werden „gesammelt“. Beim COMMIT werden lediglich REDO-Informationen in die Log-Datei geschrieben. Bei einem Systemfehler ist eine REDO-Recovery erforderlich. Die Änderungen auf einer Seite über mehrere Transaktionen hinweg können gesammelt werden.

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2

Erstellen Sie ein möglichst einfaches ER-Schema, das alle gegebenen Informationen enthält. Attribute von Entitäten und Beziehungen sind anzugeben, Schlüsselattribute durch Unterstreichen zu kennzeichnen. Verwenden Sie für die Angabe der Kardinalitäten von Beziehungen die Min-Max-Notation. Führen Sie Surrogatschlüssel (künstlich definierte Schlüssel) nur dann ein, wenn es nötig ist, und modellieren Sie nur die im Text vorkommenden Elemente.

Zunächst gibt es **Autos**, die einen eindeutigen *Namen* (AName), einen *Typ* sowie eine Liste an *Ausstattungen* besitzen.

Autos werden aus **Bauteilen** zusammengesetzt. Diese besitzen eine *ID* sowie eine *Beschreibung*.

Jedes Bauteil wird von genau einem **Zulieferer** geliefert. Zu jedem Zulieferer werden sein *Name* (ZName) sowie seine *E-Mail-Adresse* (EMailAdresse) gespeichert.

Weiter gibt es **Werke**, die einen eindeutigen *Namen* sowie einen *Standort* besitzen.

Jedes Werk besteht aus **Hallen**, welche werksintern eindeutig *nummeriert* sind. Zu-

- ☐ E: Autos
- ☐ A: Namen
- ☐ A: Typ
- ☐ A: Ausstattungen
- ☐ E: Bauteilen
- ☒ R: zusammengesetzt
- ☐ A: ID
- ☐ A: Beschreibung
- ☐ E: Zulieferer
- ☐ A: Name
- ☐ A: E-Mail-Adresse
- ☐ E: Werke
- ☐ A: Namen
- ☐ A: Standort
- ☒ R: besteht
- ☐ E: Hallen
- ☐ A: nummeriert

dem besitzt eine Halle noch eine gewisse *Größe* (in  $m^2$ ).

○ A: *Größe*

Es gibt genau zwei Typen von Hallen: **Produktionshallen** und **Ersatzteillager**.

□ E: **Produktionshallen**

□ E: **Ersatzteillager**

In jeder Produktionshalle wird mindestens ein Auto hergestellt.

↔ R: hergestellt

Zu den Ersatzteillagern wird festgehalten, welche Bauteile und wie viele davon sich dort befinden.

↔ R: festgehalten

Zu jedem **Mitarbeiter** werden eine eindeutige *ID*, sein *Vor-* und *Nachname*, die *Adresse* (*Straße*, *PLZ*, *Ort*), das *Gehalt* sowie das *Geschlecht* gespeichert.

□ E: **Mitarbeiter**

○ A: *ID*

○ A: *Vor-*

○ A: *Nachname*

○ A: *Adresse (Straße, PLZ, Ort)*

○ A: *Gehalt*

○ A: *Geschlecht*

Mitarbeiter werden unter anderem in **Reinigungskräfte**, **Werksarbeiter** und **Ingenieure** unterteilt.

□ E: **Reinigungskräfte**

□ E: **Werksarbeiter**

□ E: **Ingenieure**

Zu den Ingenieuren wird zusätzlich der *Hochschulabschluss* gespeichert. Ingenieure sind genau einem Werk zugeordnet, Werksarbeiter einer Halle.

○ A: *Hochschulabschluss*

↔ R: zugeordnet

↔ R: Halle

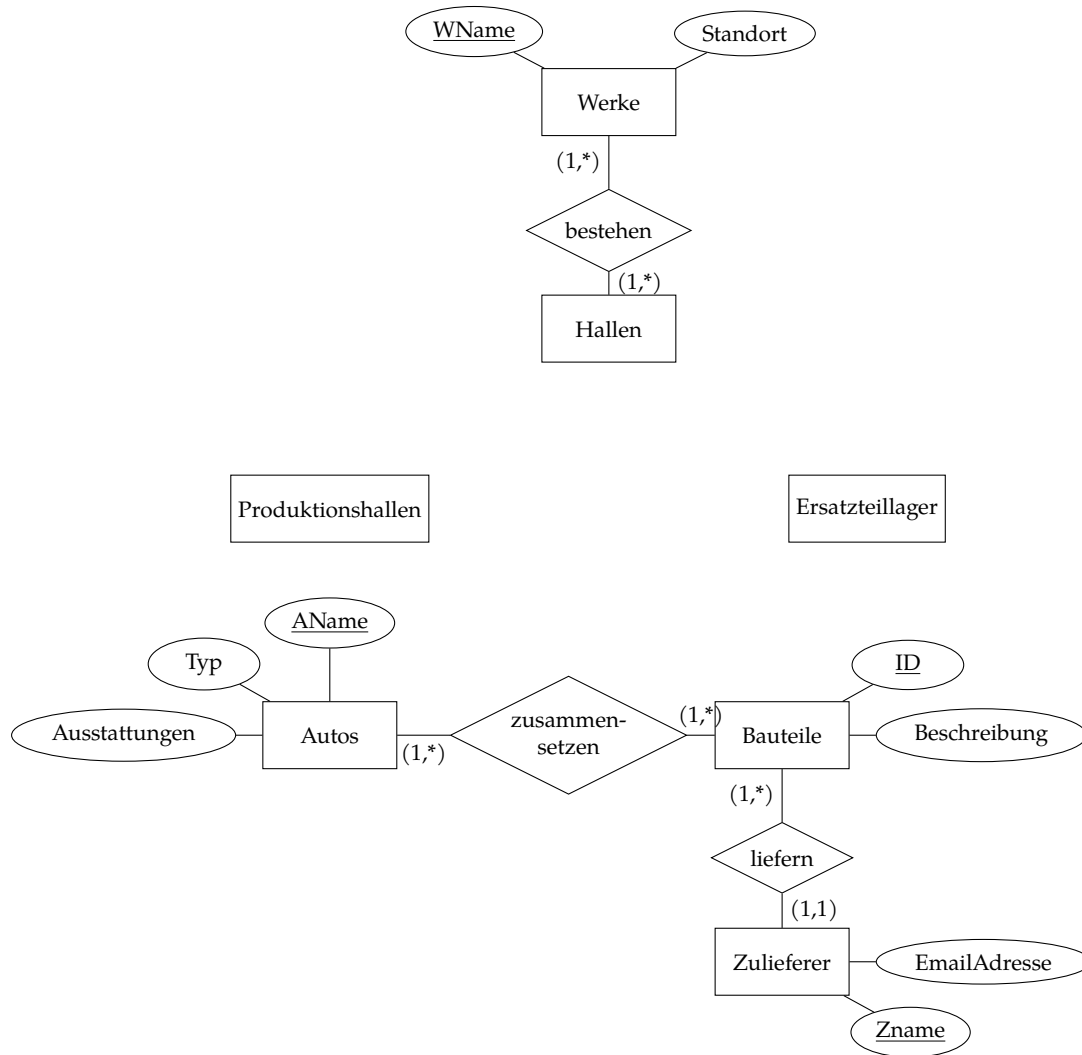
Eine Reinigungskraft reinigt mindestens eine Halle. Jede Halle muss regelmäßig gereinigt werden.

↔ R: reinigt

Weiter sind Ingenieure Projekten zugeteilt. Zudem wird zu jedem Projekt genau ein Ingenieur als *Projektleiter* festgehalten.

↔ R: zugeteilt

○ A: *Projektleiter*



### 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 3

(a) Gegeben seien die folgenden beiden Relationen R1 und R2:

**R1**

| P  | Q       | S |
|----|---------|---|
| 10 | einfach | 5 |
| 15 | b       | 8 |
| 13 | einfach | 6 |

**R2**

| A  | B | C |
|----|---|---|
| 10 | b | 6 |
| 13 | c | 3 |
| 10 | b | 5 |

Geben Sie das Ergebnis des folgenden relationalen Ausdrucks an:

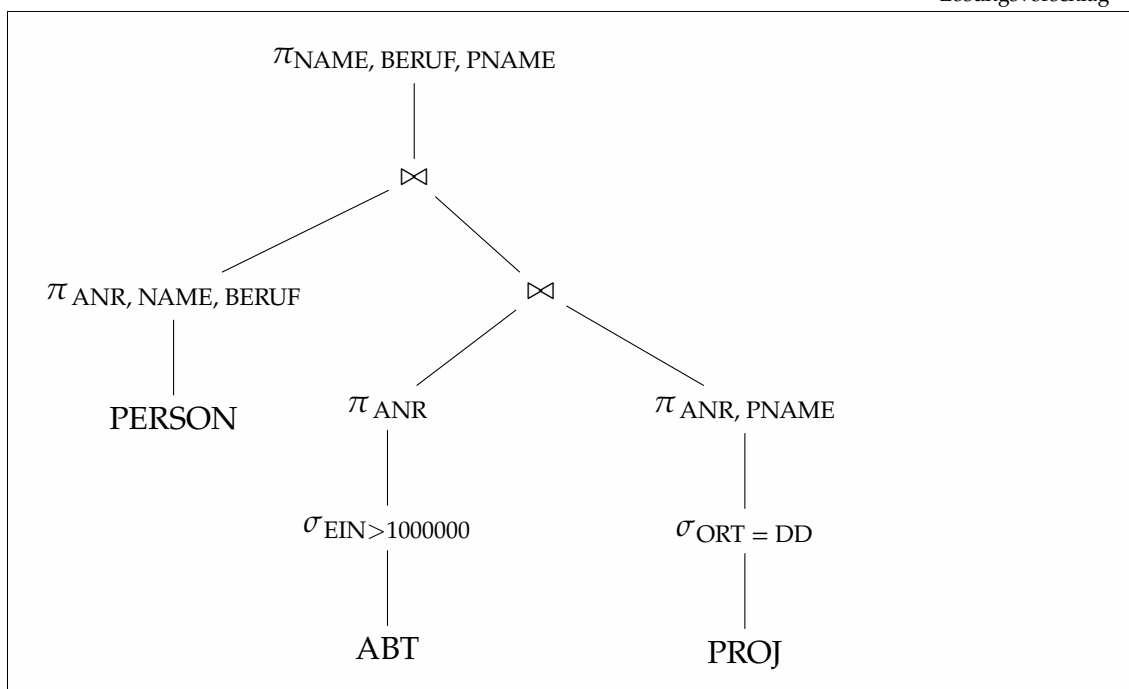
$$R_1 \bowtie_{R_1.P=R_2.A} R_2 \text{ (Equi-Join)}$$

Lösungsvorschlag

| P  | Q       | S | A  | B | C |
|----|---------|---|----|---|---|
| 10 | einfach | 5 | 10 | b | 6 |
| 10 | einfach | 5 | 10 | b | 5 |
| 13 | einfach | 6 | 13 | c | 3 |

(b) Zeichnen Sie den Operatorbaum zu folgender Abfrage in relationaler Algebra:

Lösungsvorschlag



(c) Ist der linke (bzw. rechte) Verbundoperator (Left- bzw. Right-Outer Join) assoziativ? Falls ja, beweisen Sie die Aussage, falls nein, geben Sie ein Gegenbeispiel an.

Lösungsvorschlag

Nein. Beleg durch Gegenbeispiel:

**R1**

| A | B  |
|---|----|
| 1 | 2  |
| 2 | 15 |

**R2**

| A  | C  |
|----|----|
| 1  | 35 |
| 2  | 12 |
| 13 | 5  |

**R3**

| B   | C  |
|-----|----|
| 2   | 35 |
| 100 | 35 |

**(R1 LEFT OUTER JOIN R2) LEFT OUTER JOIN R3**

| R1.A | R1.B | R2.A | R2.C | R3.B | R3.C |
|------|------|------|------|------|------|
| 1    | 2    | 1    | 35   | 2    | 35   |
| 2    | 15   | 2    | 12   | NULL | NULL |

**R1 LEFT OUTER JOIN (R2 LEFT OUTER JOIN R3)**

| R1.A | R1.B | R2.A | R2.C | R3.B | R3.C |
|------|------|------|------|------|------|
| 1    | 2    | 1    | 35   | 2    | 35   |
| 2    | 15   | NULL | NULL | NULL | NULL |

(Nur wenn beide Tabellen leer sind, wären auch die Outer Joins assoziativ).

**66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 4**

Gegeben ist das folgende Relationenschema in erster Normalform, bestehend aus zwei Relationen:

Relation1(A, B, C, D, E)  
 Relation2(F, G, H, A, E)

In diesem Schema gelten die folgenden funktionalen Abhängigkeiten:

$$FA = \left\{ \begin{array}{l} \{A, B\} \rightarrow \{C\}, \\ \{A, B, C\} \rightarrow \{E\}, \\ \{A\} \rightarrow \{D\}, \\ \{F, G\} \rightarrow \{H, A\}, \\ \{G, H\} \rightarrow \{E\}, \end{array} \right\}$$

- (a) Nennen Sie die Bedingungen, damit ein Schema in erster Normalform ist.

Lösungsvorschlag

Ein Schema ist in erster Normalform, wenn es ausschließlich atomare Attributwerte aufweist.

- (b) Überprüfen Sie, ob das Schema in zweiter Normalform ist.

Lösungsvorschlag

Eine Relation ist in 2NF, wenn sie in 1NF ist und jedes Nichtschlüsselattribut von jedem Schlüsselkandidaten voll funktional abhängig ist. Der Schlüsselkandidat ist (A, B) in Relation 1 sowie (F, G) in Relation 2.

Das Nichtschlüsselattribut D in Relation 1 ist nicht voll funktional abhängig von (A, B), sondern nur von A. Somit ist das Schema nicht in 2NF. Alle anderen Nichtschlüsselattribute sind voll funktional abhängig.

- (c) Wenden Sie den Synthesealgorithmus an, um das Schema in ein Schema in dritter Normalform zu überführen.

Lösungsvorschlag

(i) **Kanonische Überdeckung**

— Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden. —

i. **Linksreduktion**

— Führe für jede funktionale Abhängigkeit  $\alpha \rightarrow \beta \in F$  die Linksreduktion durch, überprüfe also für alle  $A \in \alpha$ , ob A überflüssig ist, d. h. ob  $\beta \subseteq \text{AttrHülle}(F, \alpha - A)$ . —

$$FA = \left\{ \begin{array}{l} \{A, B\} \rightarrow \{C\}, \\ \{A, B\} \rightarrow \{E\}, \\ \{A\} \rightarrow \{D\}, \\ \{F, G\} \rightarrow \{H, A\}, \\ \{G, H\} \rightarrow \{E\}, \end{array} \right\}$$

ii. **Rechtsreduktion**

— Führe für jede (verbliebene) funktionale Abhängigkeit  $\alpha \rightarrow \beta$  die Rechtsreduktion durch, überprüfe also für alle  $B \in \beta$ , ob  $B \in \text{AttrHülle}(F - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta -$



$B)), \alpha)$  gilt. In diesem Fall ist  $B$  auf der rechten Seite überflüssig und kann eliminiert werden,  $\delta\alpha \rightarrow \beta$  wird durch  $\alpha \rightarrow (\beta - B)$  ersetzt. —————

nichts zu tun

iii. **Löschen leerer Klauseln**

— Entferne die funktionalen Abhängigkeiten der Form  $\alpha \rightarrow \emptyset$ , die im 2. Schritt möglicherweise entstanden sind. —————

nichts zu tun

iv. **Vereinigung**

— Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form  $\alpha \rightarrow \beta_1, \dots, \alpha \rightarrow \beta_n$ , so dass  $\alpha \rightarrow \beta_1 \cup \dots \cup \beta_n$  verbleibt. —————

$$FA = \left\{ \begin{array}{l} \{A, B\} \rightarrow \{C, E\}, \\ \{A\} \rightarrow \{D\}, \\ \{F, G\} \rightarrow \{H, A\}, \\ \{G, H\} \rightarrow \{E\}, \end{array} \right\}$$

(ii) **Relationsschemata formen**

— Erzeuge für jede funktionale Abhängigkeit  $\alpha \rightarrow \beta \in F_c$  ein Relationenschema  $\mathcal{R}_\alpha := \alpha \cup \beta$ .

R1 (A, B, C, E) R2 (A, D) R3 (F, G, H, A) R4 (G, H, E)

(iii) **Schlüssel hinzufügen**

— Falls eines der in Schritt 2. erzeugten Schemata  $\mathcal{R}_\alpha$  einen Schlüsselkandidaten von  $\mathcal{R}$  bezüglich  $F_c$  enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten  $\mathcal{K} \subseteq \mathcal{R}$  aus und definiere folgendes zusätzliche Schema:  $\mathcal{R}_\mathcal{K} := \mathcal{K}$  und  $\mathcal{F}_\mathcal{K} := \emptyset$  —————

R1 (A, B, C, E) R2 (A, D) R3 (F, G, H, A) R4 (G, H, E) R5 (B, F, G)

als Verbindung von R1 bis R4 (Attributhülle erhält alle Attribute, ist daher Schlüsselkandidat)

(iv) **Entfernung überflüssiger Teilschemata**

— Eliminiere diejenigen Schemata  $\mathcal{R}_\alpha$ , die in einem anderen Relationenschema  $\mathcal{R}_{\alpha'}$  enthalten sind, d. h.  $\mathcal{R}_\alpha \subseteq \mathcal{R}_{\alpha'}$ . —————

nichts zu tun

- (d) Sei nun das Relationenschema  $R(A,B,C,D)$  in erster Normalform gegeben. In  $R$  gelten die folgenden funktionalen Abhängigkeiten:

$$FA = \left\{ \begin{array}{l} \{A, B\} \rightarrow \{D\}, \\ \{B\} \rightarrow \{C\}, \\ \{C\} \rightarrow \{B\}, \end{array} \right\}$$

Welches ist die höchste Normalform, in der sich das Schema R befindet? Begründen Sie Ihre Entscheidung.

### 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 5

Gegeben sind die folgenden transaktionsähnlichen Abläufe. (Zunächst wird auf das Setzen von Sperren verzichtet.) Hierbei steht  $R(X)$  für ein Lesezugriff auf  $X$  und  $W(X)$  für einen Schreibzugriff auf  $X$ .

| T1          | T2          |
|-------------|-------------|
| $R(A)$      | $R(D)$      |
| $A := A-10$ | $D := D-20$ |
| $W(A)$      | $W(D)$      |
| $R(C)$      | $R(A)$      |
| $R(B)$      | $A := A+20$ |
| $B := B+10$ | $W(A)$      |
| $W(B)$      |             |

Betrachten Sie folgenden Schedule:

| T1          | T2          |
|-------------|-------------|
| $R(A)$      | $R(D)$      |
|             | $D := D-20$ |
|             | $W(D)$      |
|             | $R(A)$      |
|             | $A := A+20$ |
|             | $W(A)$      |
| $A := A-10$ |             |
| $W(A)$      |             |
| $R(C)$      |             |
| $R(B)$      |             |
| $B := B+10$ |             |
| $W(B)$      |             |

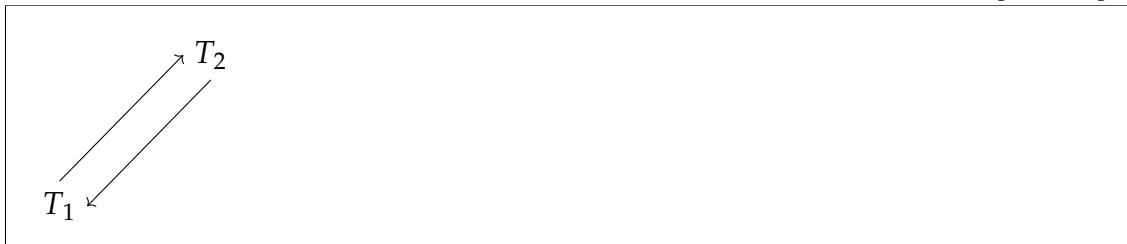
- (a) Geben Sie die Werte von  $A$ ,  $B$ ,  $C$  und  $D$  nach Ablauf des Schedules an, wenn mit  $A = 100$ ,  $B = 200$ ,  $C = \text{true}$  und  $D = 150$  begonnen wird.

Lösungsvorschlag

- A** 90 ( $A := A - 10 := 100 - 10$ ) T2 schreibt 120 in A, was aber von T1 wieder überschrieben wird.
- B** 210 (B wird nur in T1 gelesen, verändert und geschrieben)
- C** true (C wird nur in T1 gelesen)
- D** 130 (D wird nur in T2 gelesen, verändert und geschrieben)

(b) Geben Sie den Dependency-Graphen des Schedules an.

Lösungsvorschlag



(c) Geben Sie alle auftretenden Konflikte an.

Lösungsvorschlag

- $R_1(A) < W_2(A)$  resultierende Kante  $T_1 \rightarrow T_2$ ,
- $R_2(A) < W_1(A)$  resultierende Kante  $T_2 \rightarrow T_1$ ,
- $W_2(A) < W_1(A)$  resultierende Kante  $T_2 \rightarrow T_1$  (bereits vorhanden)

(d) Begründen Sie, ob der Schedule serialisierbar ist.

Lösungsvorschlag

Nicht ohne den Einsatz der Lese- und Schreibsperrern, denn der Dependency Graph enthält einen Zyklus, womit er nicht konfliktserialisierbar ist.

(e) Beschreiben Sie, wie die beiden Transaktionen mit LOCK Aktionen erweitert werden können, so dass nur noch serialisierbare Schedules ausgeführt werden können. Die Angabe eines konkreten Schedules ist nicht zwingend notwendig.

Lösungsvorschlag

Hier führt die Verwendung des Zwei-Phasen-Sperrpotokolls zur gewünschten Serialisierbarkeit. (Es muss dabei weder die konservative, noch die strenge Variante verwendet werden, damit es funktioniert). T1 würde zu Beginn die Lese- und Schreibsperre für A anfordern, den Wert verändern, zurückschreiben und anschließend die Sperren für A zurückgeben. Währenddessen könnte T2 „ungestört“ die Schreib- und Lesesperren für D anfordern, D lesen, verändern und schreiben, und die Sperren zurückgeben. T2 bemüht sich nun um die Lesesperre für A, muss aber nun so lange warten, bis T1 die Schreibsperre zurückgegeben hat. Dadurch kann man den Lost-Update-Fehler vermeiden und erhält allgemein einen serialisierbaren Schedule.

Beispiel für einen konkreten Schedule mit LOCKs (auch wenn nicht zwin-

gend gefordert in der Aufgabenstellung):

| T1        | T2             |
|-----------|----------------|
| rLock(A)  |                |
| xLock(A)  |                |
|           | rLock(D)       |
|           | xLock(D)       |
| R1(A)     |                |
|           | R2(D)          |
| A := A-10 |                |
|           | D := D-20      |
|           | W2(D)          |
|           | unLock(D)      |
|           | rLock(A) DELAY |
| W1(A)     |                |
| unLock(A) |                |
|           | R2(A)          |
|           | A := A+20      |
|           | W2(A)          |
|           | unLock(A)      |
| rLock(C)  |                |
| R1(D)     |                |
| unLock(C) |                |
|           | commit         |
| rLock(B)  |                |
| xLock(B)  |                |
| R1(B)     |                |
| B := B+10 |                |
| W1(B)     |                |
| unLock(B) |                |
| commit    |                |

**66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 6**

Gegeben ist folgendes Relationenschema zur Verwaltung von Daten aus der Fußballweltmeisterschaft:

Die Tabelle Match wurde in Spiel umgenannt, da es sonst zu Konflikten mit der SQL-Syntax kommt, da match ein SQL Schlüsselwort ist.

Nation (Land, Kapitän, Trainer) Kapitän ist Fremdschlüssel zu Spieler\_ID in Spieler.

Spiel (Spiel\_ID, Ort, Datum, Team1, Team2, ToreTeam1, ToreTeam2) Team1 ist Fremdschlüssel zu Land in Nation. Team2 ist Fremdschlüssel zu Land in Nation.

Spieler (Spieler\_ID, Name, Vorname, Wohnort, Land) Land ist Fremdschlüssel zu Land in Nation.

Platzverweise (Platzverweis\_ID, Spiel\_ID, Spieler\_ID, Spielminute) Spiel\_ID ist Fremdschlüssel zu Spiel\_ID in Spiel. Spieler\_ID ist Fremdschlüssel zu Spieler\_ID in Spieler.

Die Primärschlüssel der Relationen sind wie üblich durch Unterstreichen gekennzeichnet. Pro Ort und Datum findet jeweils nur ein Spiel statt.

Formulieren Sie folgende Abfragen in SQL. Vermeiden Sie nach Möglichkeit übermäßige Nutzung von Joins und Views.

- (a) Ermitteln Sie die Anzahl der Platzverweise pro Spieler und geben Sie jeweils Name und Vorname des Spielers mit aus. Die Ausgabe soll nach der Anzahl der Platzverweise absteigend erfolgen.

Lösungsvorschlag

```
SELECT COUNT(*) AS Anzahl, s.Name, s.Vorname
FROM Platzverweise p, Spieler s
WHERE p.Spieler_ID = s.Spieler_ID
GROUP BY s.Name, s.Vorname
ORDER BY Anzahl DESC;
```

| anzahl | name       | vorname |
|--------|------------|---------|
| 2      | Matthäus   | Lothar  |
| 1      | Bodden     | Olaf    |
| 1      | Beckham    | David   |
| 1      | Rizzitelli | Luca    |
| 1      | Babel      | Markus  |
| 1      | Häßler     | Thomas  |

(6 rows)

- (b) Welches ist die maximale Anzahl an Toren, die eine Mannschaft insgesamt im Turnier erzielt hat? (Sie dürfen der Einfachheit halber annehmen, dass jede Mannschaft jeweils mindestens einmal als Team1 und Team2 angetreten ist.)

Lösungsvorschlag

```
SELECT MAX(tmp2.Summe) FROM (
 SELECT Team, SUM(Summe) as Summe FROM (
 SELECT Team1 AS Team, SUM(ToreTeam1) AS Summe
 FROM Spiel
 GROUP BY Team1, ToreTeam1
 UNION
 SELECT Team2 AS Team, SUM(ToreTeam2) AS Summe
```

```

 FROM Spiel
 GROUP BY Team2, ToreTeam2
) AS tmp
 GROUP BY Team
) AS tmp2;

max

 9
(1 row)

```

(c) Wie viele Tore sind im Turnier insgesamt gefallen?

Lösungsvorschlag

```

SELECT SUM(ToreTeam1 + ToreTeam2) AS GesamtanzahlTore
FROM Spiel;

```

(d) Ermitteln Sie die Namen und Länder der fünf Spieler, die nach der kürzesten Spielzeit einen Platzverweis erhielten. Die Ausgabe soll nummeriert erfolgen (beginnend bei 1 für die kürzeste Spielzeit).

Lösungsvorschlag

```

SELECT s.Name, s.Land, COUNT(*) AS Rang
FROM Spieler s, Platzverweise p1, Platzverweise p2
WHERE
 s.Spieler_ID = p2.Spieler_ID AND
 p1.Spielminute <= p2.Spielminute
GROUP BY s.Name, s.Land, p2.Spieler_ID
HAVING COUNT(*) < 6
ORDER BY Rang;

```

Der Erstplatzierte kommt durch die WHERE-Bedingungen nur einmal in der Relation vor, weil sein Eintrag genau einmal mit sich selbst vorkommt. Alle anderen Einträge, bei denen die p2.Spieler\_ID, der Spieler mit der „geringsten Minute“ ist, werden ja eliminiert, da ja nur die Einträge behalten werden, die der Bedingung p1.Spielminute <= p2.Spielminute entsprechen.

## 66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1

Gegeben seien folgende Tätigkeiten mit ihren Abhängigkeiten und Dauern:

| Task | Dauer (in h) | Abhängigkeiten |
|------|--------------|----------------|
| T1   | 3            | /              |
| T2   | 6            | /              |
| T3   | 2            | T1             |
| T4   | 2            | T2             |
| T5   | 5            | T1             |
| T6   | 3            | T4, T5         |
| T7   | 6            | T3             |
| T8   | 7            | T4             |
| T9   | 4            | T6, T8         |
| T10  | 1            | T7, T9         |

- (a) Zeichnen Sie ein CPM-Diagramm basierend auf der gegebenen Aufgabenliste. Benutzen Sie explizite Start- und Endknoten.

Lösungsvorschlag

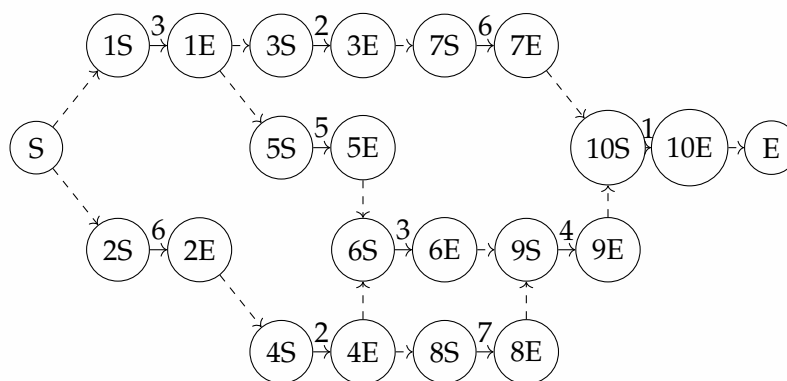
### Abkürzungen

**S** Start

**1S** Start von T1

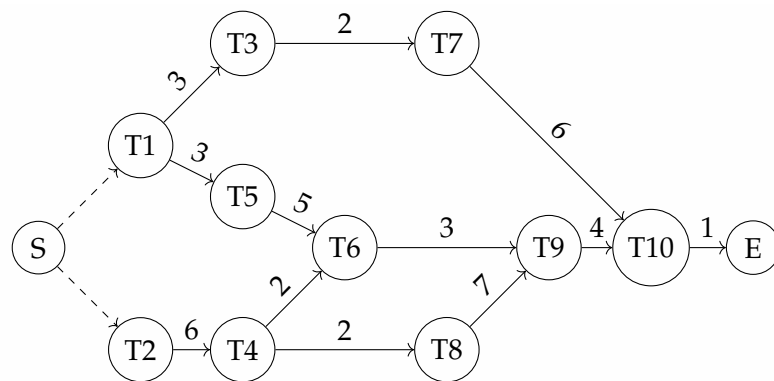
**1E** Ende von T1

**E** Ende

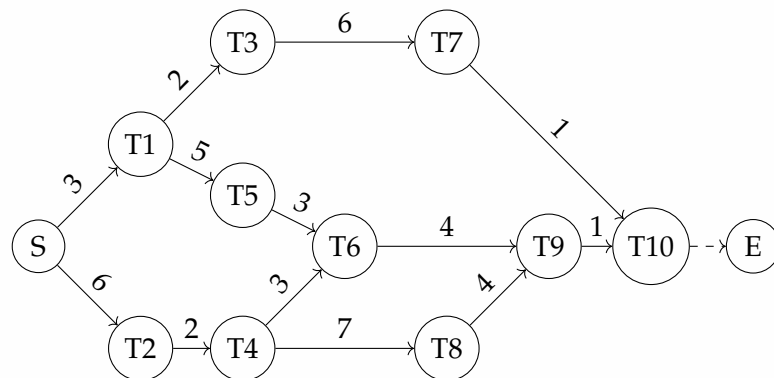


Teilen wir einen Task in zwei Knoten auf, so wird das Diagramm sehr unübersichtlich. Wir verwenden pro Task nur einen Knoten. Es gibt zwei Möglichkeiten:

**Knoten sind Anfang der Tasks**



### Knoten sind Ende der Tasks



- (b) Als *Slack* bezeichnet man die Zeit, um die eine Aufgabe bezüglich ihres frühesten Startzeitpunktes verzögert werden kann, ohne dass es Probleme bei der fristgerechten Fertigstellung des Projektes gibt. Berechnen Sie den Slack für alle Aktivitäten und ergänzen Sie ihn in Ihrem Diagramm.

Lösungsvorschlag

### Knoten sind Anfang der Tasks

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $FZ_i$ : Frühester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann. \_\_\_\_\_



| $i$ | Nebenrechnung            | $FZ_i$ |
|-----|--------------------------|--------|
| T1  |                          | 0      |
| T2  |                          | 0      |
| T3  |                          | 3      |
| T4  |                          | 6      |
| T5  |                          | 3      |
| T6  | $\max(8_{T4}, 8_{T5})$   | 8      |
| T7  |                          | 5      |
| T8  |                          | 8      |
| T9  | $\max(11_{T6}, 15_{T4})$ | 15     |
| T10 | $\max(19_{T9}, 11_{T7})$ | 19     |
| E   |                          | 20     |

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $SZ_i$ : Spätester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann. —

| $i$ | Nebenrechnung           | $SZ_i$ |
|-----|-------------------------|--------|
| E   |                         | 20     |
| T10 |                         | 19     |
| T9  |                         | 15     |
| T8  |                         | 8      |
| T7  |                         | 13     |
| T6  |                         | 12     |
| T5  |                         | 7      |
| T4  | $\min(12_{T6}, 6_{T8})$ | 6      |
| T3  |                         | 11     |
| T2  |                         | 0      |
| T1  | $\min(8_{T3}, 4_{T5})$  | 4      |

| $i$    | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | E  |
|--------|----|----|----|----|----|----|----|----|----|-----|----|
| $FZ_i$ | 0  | 0  | 3  | 6  | 3  | 8  | 5  | 8  | 15 | 19  | 20 |
| $SZ_i$ | 4  | 0  | 11 | 6  | 7  | 12 | 13 | 8  | 15 | 19  | 20 |
| GP     | 4  | 0  | 8  | 0  | 4  | 4  | 8  | 0  | 0  | 0   | 0  |

**Knoten sind Ende der Tasks**

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $FZ_i$ : Frühester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann.

| $i$ | Nebenrechnung            | $FZ_i$ |
|-----|--------------------------|--------|
| T1  |                          | 3      |
| T2  |                          | 6      |
| T3  |                          | 5      |
| T4  |                          | 8      |
| T5  |                          | 8      |
| T6  | $\max(11_{T4}, 11_{T5})$ | 11     |
| T7  |                          | 11     |
| T8  |                          | 15     |
| T9  | $\max(15_{T6}, 19_{T8})$ | 19     |
| T10 | $\max(20_{T9}, 12_{T7})$ | 20     |
| E   |                          | 20     |

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $SZ_i$ : Spätester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann.

| $i$ | Nebenrechnung           | $SZ_i$ |
|-----|-------------------------|--------|
| E   |                         | 20     |
| T10 |                         | 20     |
| T9  |                         | 19     |
| T8  |                         | 15     |
| T7  |                         | 19     |
| T6  |                         | 15     |
| T5  |                         | 12     |
| T4  | $\min(12_{T6}, 8_{T8})$ | 8      |
| T3  |                         | 13     |
| T2  |                         | 6      |
| T1  | $\min(11_{T3}, 7_{T5})$ | 7      |

| $i$    | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | E  |
|--------|----|----|----|----|----|----|----|----|----|-----|----|
| $FZ_i$ | 3  | 6  | 5  | 8  | 8  | 11 | 11 | 15 | 19 | 20  | 20 |
| $SZ_i$ | 7  | 6  | 13 | 8  | 12 | 15 | 19 | 15 | 19 | 20  | 20 |
| GP     | 4  | 0  | 8  | 0  | 4  | 4  | 8  | 0  | 0  | 0   | 0  |

- (c) Zeichnen Sie den kritischen Pfad in Ihr Diagramm ein oder geben Sie die Tasks des kritischen Pfades in der folgenden Form an: **Start ! ... ! Ende**. Sollte es mehrere kritische Pfade geben, geben Sie auch diese an. Wie lange ist die Dauer des kritischen Pfades bzw. der kritischen Pfade?

Lösungsvorschlag

Kritischer Pfad: **Start ! T2 ! T4 ! T8 ! T9 ! T10 ! Ende**  
Dauer: 20 h

## 66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2

- (a) Erklären Sie den Unterschied zwischen *iterativen* und *inkrementellen* Entwicklungsprozessen. Nennen Sie zudem je ein Prozessmodell als Beispiel.

Lösungsvorschlag

**Iterativ:** Ein Entwicklungszyklus wird immer wieder durchlaufen:

Planung → Implementierung → Testung → Evaluation.

Mit jeder Iteration wird das Produkt verfeinert. Das Wasserfallmodell in seiner normalen Form würde dies nicht erfüllen, da hier alle Phasen nur einmal durchlaufen werden.

**Beispiel:** Agile Programmierung, erweitertes Wasserfallmodell

**Inkrementell:** Das Projekt wird in einzelne Teile zerlegt. An jedem Teilprojekt kann bestenfalls separat gearbeitet werden. In den einzelnen Teilprojekten kann dann ebenfalls wieder iterativ gearbeitet werden, die beiden Methoden schließen sich gegenseitig also nicht aus.

**Beispiel:** Agile Programmierung, V-Modell XT, Aufteilung in Teilprojekte, die alle mit Wasserfallmodell bearbeitet werden

- (b) Nennen und erklären Sie kurz die vier Leitsätze der agilen Softwareentwicklung laut *Agilem Manifest*.

(i) **Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge**

Ein festgelegter Prozess, der nicht sinnvoll von den beteiligten Personen umgesetzt werden kann, ist nicht sinnvoll und sollte nicht durchgeführt

werden. Es geht darum, die Stärken der Mitarbeiter einzusetzen und sich nicht sklavisches an Abläufen zu orientieren. Es geht darum, Freiräume zu schaffen, um das Potential des Einzelnen voll entfalten zu lassen.

(ii) **Funktionierende Software ist wichtiger als umfassende Dokumentation**

Der Kunde ist in erster Linie an einem funktionierenden Produkt interessiert. Es soll möglichst früh ein Prototyp entstehen, der dann im weiteren Prozess an die Bedürfnisse des Kunden angepasst wird. Eine umfassende Dokumentation kann am Ende immer noch ergänzt werden.

(iii) **Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlung**

Anforderungen und Spezifikationen können sich flexibel ändern, dadurch kann der Kunde direkt Rückmeldung geben, was ist nicht wichtig im Vertrag kleinste Details zu regeln, sondern auf die Bedürfnisse des Kunden einzugehen. Daraus folgt direkt der letzte Punkt:

(iv) **Reagieren auf Veränderung ist wichtiger als das Befolgen eines Plans**

Softwareentwicklung ist ein dynamischer Prozess. Das starre Befolgen eines Plans widerspricht der Grundidee der agilen Programmierung.

- (c) Beschreiben Sie die wesentlichen Aktivitäten in *Scrum* (agiles Entwicklungsmodell) inklusive deren zeitlichen Ablaufs. Gehen Sie dabei auch auf die *Artefakte* ein, die im Verlauf der Entwicklung erstellt werden.

Lösungsvorschlag

(i) **Initiale Projekterstellung:**

Festlegen eines Product Backlog durch den Product Owner, Erstellen von ersten User Stories. Diese entstehen gegebenenfalls durch den Kontakt mit den Stakeholdern, falls der Product Owner nicht selbst der Kunde ist.

(ii) **Sprint planning:**

Der nächste Sprint (zwischen 1 und 4 Wochen) wird geplant. Dabei wird ein Teil des Product Backlog als *Sprint Backlog* definiert und auf die einzelnen Entwickler verteilt. Es wird festgelegt, *was* implementiert werden soll (Product Owner anwesend) und *wie* das geschehen soll (Entwicklerteam). Während des Sprints findet jeden Tag ein *Daily Scrum* statt, ein fünfzehnminütiger Austausch zum aktuellen Stand. Am Ende des Sprints gibt es ein *Sprint Review* und das *Product Inkrement* wird evaluiert und das *Product Backlog* gegebenenfalls angepasst in Absprache mit *Product Owner* und *Stakeholdern*.

**Artefakte:**

- Product Backlog
- Sprint Backlog

---

- Product Increment

---

- (d) Nennen und erklären Sie die *Rollen* in einem Scrum-Team.

Lösungsvorschlag

**Product Owner** Verantwortlich für das Projekt, für die Reihenfolge der Bearbeitung und Implementierung, ausgerichtet auf Maximierung und wirtschaftlichen Erfolg. Er führt und aktualisiert das *Product Backlog*.

**Scrum Master** Führungsperson, die das Umsetzen des Scrum an sich leitet und begleitet. Er mischt sich nicht mit konkreten Arbeitsanweisungen ein, sondern moderiert und versucht Hindernisse zu beseitigen, die einem Gelingen der Sprintziele im Weg sind.

**Entwickler** Sie entwickeln und implementieren die einzelnen Produktfunktionalitäten, die vom Product Owner festgelegt wurden, in eigenverantwortlicher Zeit. Sie müssen die Qualitätsstandards beachten.

**66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 4**

- (a) Nennen und erklären Sie kurz die drei Kategorien der Organisation von klassischen Entwurfsmustern. Geben Sie zu jeder Kategorie ein Beispiel-Pattern an.
- (b) Erstellen Sie ein Klassendiagramm zu den Komponenten einer Beobachtungsstation, welche aus einem einzigen Wasserstandsmesser besteht. Dabei sollen bei Änderungen des Wasserstandes der aktuelle Wasserstand sowohl auf der Konsole als auch in eine Log-Datei geschrieben werden. Der momentane Wasserstand soll dabei mittels einer Variablen des Datentyps `double` dargestellt werden. Die verschiedenen Anzeigearten (Konsolenanzeige, Logger) sollen als verschiedene Klassen modelliert werden und enthalten jeweils nur eine Methode zum Anzeigen bzw. Schreiben des aktuellen Wasserstandes. Verwenden Sie für die Realisierung dieses Klassendiagramms die passenden Entwurfsmuster.

Hinweise:

- Getter und Setter müssen nicht eingezeichnet werden.
- Fügen Sie auch Methoden ein, welche durch die einzelnen Klassen implementiert werden.

**66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 5**

- (a) Nennen Sie vier Programmierparadigmen.

Lösungsvorschlag

- Imperative Programmierung
- Prozedurale Programmierung
- Funktionale Programmierung
- Objektorientierte Programmierung

a

<sup>a</sup>[https://de.wikipedia.org/wiki/Programmierparadigma#Strukturierte\\_Programmierung](https://de.wikipedia.org/wiki/Programmierparadigma#Strukturierte_Programmierung)

Sequenzdiagramm

(b) Erläutern Sie die Begriffe *Overloading* und *Overriding*, sowie deren Unterschiede.

Lösungsvorschlag

- Beim Überladen muss die Methode eine andere Signatur haben, beim Überschreiben dieselbe Signatur.
- Die Intention beim Überladen ist, Methode zu erweitern, beim Überschreiben die Methode vollständig zu ersetzen.
- Überladen der Methode wird verwendet, um den Polymorphismus der Kompilierzeit zu erreichen. Das Überschreiben der Methode wird verwendet, um einen Laufzeit-Polymorphismus zu erreichen.  
Bei der Methodenüberladung weiß der Compiler, welches Objekt welcher Klasse zum Zeitpunkt der Kompilierung zugewiesen wurde. In der Methodenüberschreibung sind diese Informationen jedoch erst zur Laufzeit bekannt.
- Das Überladen von Methoden findet in derselben Klasse statt, während das Überschreiben in einer von einer Basisklasse abgeleiteten Klasse stattfindet.

a

<sup>a</sup><https://gadget-info.com/difference-between-method-overloading>

(c) Erläutern Sie, wie sich zentrale und dezentrale Versionsverwaltung unterscheiden.

Lösungsvorschlag

Beim der dezentralen Versionsverwaltung hat jede/r EntwicklerIn das komplette Repository mit seiner kompletten History lokal gespeichert und kann diese dann mit anderen Repositories abgleichen.

Bei der zentralen Versionsverwaltung gibt es einen zentralen Server, der die komplette History vorhält.

(d) Erstellen Sie ein Sequenzdiagramm zur Methode `main` der Klasse `Webshop`.

Hinweise:

- Arithmetische Operationen müssen nicht weiter aufgelöst werden.
- Listenoperationen müssen nicht explizit dargestellt werden.
- Auf das Zeichnen einer passiven Lebenslinie muss nicht geachtet werden.
- Übertragen Sie das untenstehende Diagramm als Ausgangspunkt in Ihren Bearbeitungsbogen.

```
public class Webshop {
 public static void main(String[] args) {
 Bestellung b1 = new Bestellung();

 // ab hier soll modelliert werden
 Artikel a1 = new Artikel();
 a1.setName("Taschenrechner");
 a1.setPrice(10);

 b1.addArticle(a1);

 Bestellung b2 = new Bestellung();
 Artikel a2 = new Artikel();
 a2.setName("Lineal");
 a2.setPrice(2.5);

 Artikel a3 = new Artikel();
 a3.setName("Bleistift");
 a3.setPrice(0.7);

 b2.addArticle(a3);
 b1.addArticle(a2);

 b1.getSize();

 b2.getPrice();
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/webshop/Webshop.java](src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/webshop/Webshop.java)

```
@SuppressWarnings({"unused"})
public class Artikel {
 private String name;

 private double price;

 public void setName(String name) {
 this.name = name;
 }

 public void setPrice(double price) {
 this.price = price;
 }

 public double getPrice() {
 return price;
 }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/webshop/Artikel.java](src/main/java/org/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/webshop/Artikel.java)

```
import java.util.ArrayList;
import java.util.List;

public class Bestellung {
```



```
private List<Artikel> articles;
// Anzahl an Artikeln
private int size = 0;
// Gesamtpreis der Bestellung
private double price = 0;

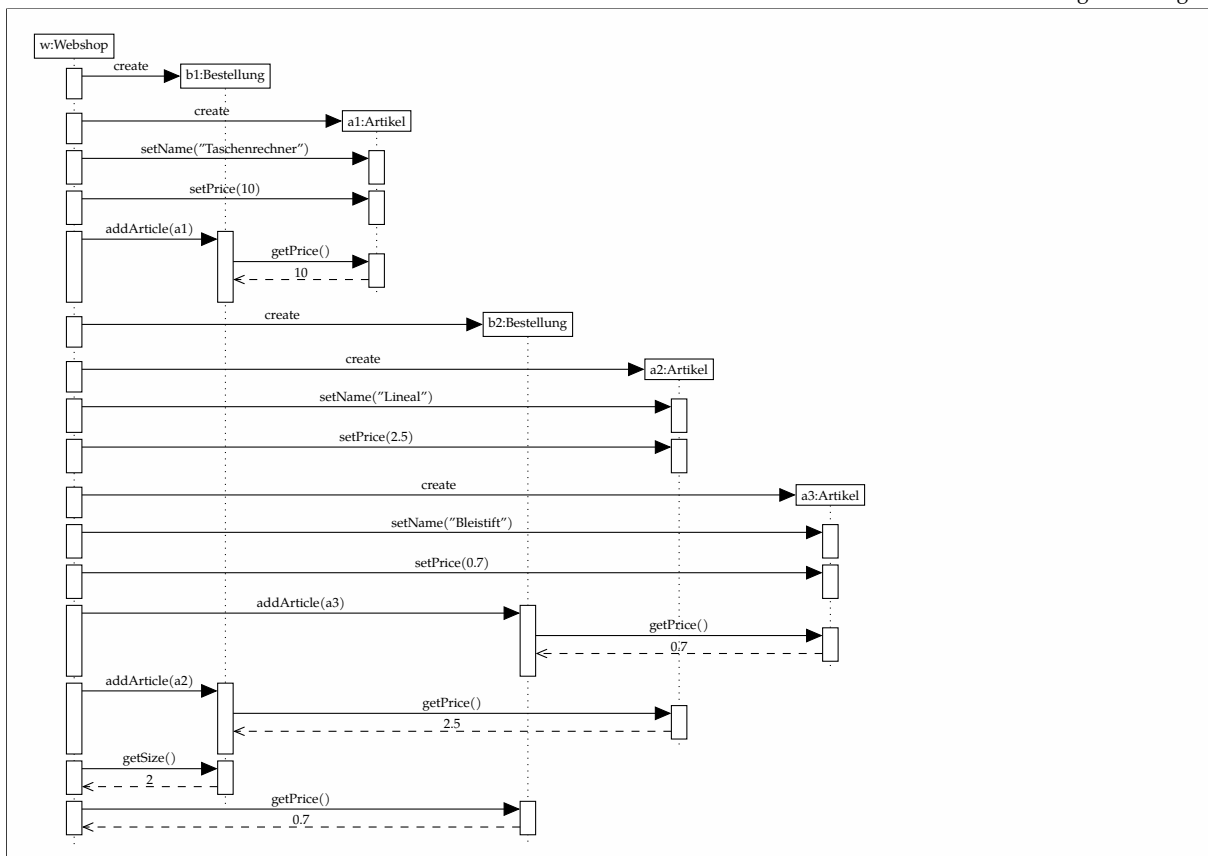
public Bestellung() {
 articles = new ArrayList<>();
}

public void addArticle(Artikel article) {
 // muss nicht weiter aufgelöst werden, siehe Hinweise
 articles.add(article);
 size++;
 // muss nicht weiter aufgelöst werden, siehe Hinweise
 price = article.getPrice() + price;
}

public int getSize() {
 return size;
}

public double getPrice() {
 return price;
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2021/fruehjahr/webshop/Bestellung.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2021/fruehjahr/webshop/Bestellung.java)



## 66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Beantworten Sie die folgenden Fragen und begründen oder erläutern Sie Ihre Antwort.

- (a) Kann ein Tupel mehrfach im Ergebnis einer SQL-Anfrage enthalten sein?

Ja. Geben wir nur eine Teilmenge an Attributen aus (z. B. ohne Primärschlüssel), so kann ein Tupel mehrfach in der Ausgabe erscheinen.

Außerdem ist es möglich eine Tabelle ohne PRIMARY KEY anzulegen. In so einer Tabelle kann dann eine Tupel mehrmals gespeichert werden und über `SELECT * FROM ...` mehrmals ausgegeben werden. (getestet in MySQL und in PostgreSQL).

```
CREATE TABLE tmp (
 tmp INTEGER
);

INSERT INTO tmp VALUES
 (1),
 (1),
 (1);

SELECT * FROM tmp;
```

```

+-----+
| tmp |
+-----+
| 1 |
| 1 |
| 1 |
+-----+

```

Um die mehrfache Ausgabe zu verhindern, gibt es in SQL das Schlüsselwort `DISTINCT`. In der Relationalen Algebra hingegen sind die Tupel einer Relation eindeutig.

- (b) Was ist der Unterschied zwischen einem `INNER JOIN` und einem `OUTER JOIN`?

Lösungsvorschlag

Ein `INNER JOIN` entspricht der Schnittmenge  $A \cap B$ .

Ein `OUTER JOIN` entspricht der Vereinigung  $A \cup B$ . Bei `OUTER JOINS` können auch `NULL`-Werte vorkommen.<sup>a</sup>

<sup>a</sup><https://stackoverflow.com/a/38578>

- (c) Welche Auswirkung hat die Verwendung von `ON DELETE CASCADE` bei einem Fremdschlüsselattribut?

Lösungsvorschlag

Ist `ON DELETE CASCADE` bei einem Fremdschlüsselattribut gesetzt, so wird der referenzierte Datensatz bei einem Löschvorgang mitgelöscht.

- (d) Kann eine abgebrochene (aborted) Transaktion wieder fortgesetzt werden?

Lösungsvorschlag

Eine Transaktion kann nicht fortgesetzt werden. Sie muss zurückgesetzt und wiederholt werden.

- (e) Was versteht man unter einer `stored procedure` im Kontext einer Programmierschnittstelle für relationale Datenbanken (z.B. `JDBC`)?

Lösungsvorschlag

Eine `stored procedure` bildet eine Gruppe von SQL-Befehlen, die eine logische Einheit bilden und einer bestimmten Aufgabe zugeordnet sind. `stored procedure` werden dazu benutzt, mehrere Anweisungen und Abfragen zu koppeln.

Beispielsweise können bei einer Angestellten-Datenbank die Aufgaben „einstellen“, „entlassen“, „befördern“ als `stored procedure` kompiliert werden und dann mit unterschiedlichen Parametern ausgeführt werden.<sup>a</sup>

<sup>a</sup><https://docs.oracle.com/javase/tutorial/jdbc/basics/storedprocedures.html>

- (f) Was sind `check constraints` und wie wirken sich diese aus?

Constraints definieren Bedingungen, die beim Einfügen, Ändern und Löschen von Datensätzen in der Datenbank erfüllt werden müssen. Wird beispielsweise eine Bedingung bei Einfügen eines Datensatzes nicht erfüllt, so kann dieser Datensatz nicht gespeichert werden.

## 66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 2

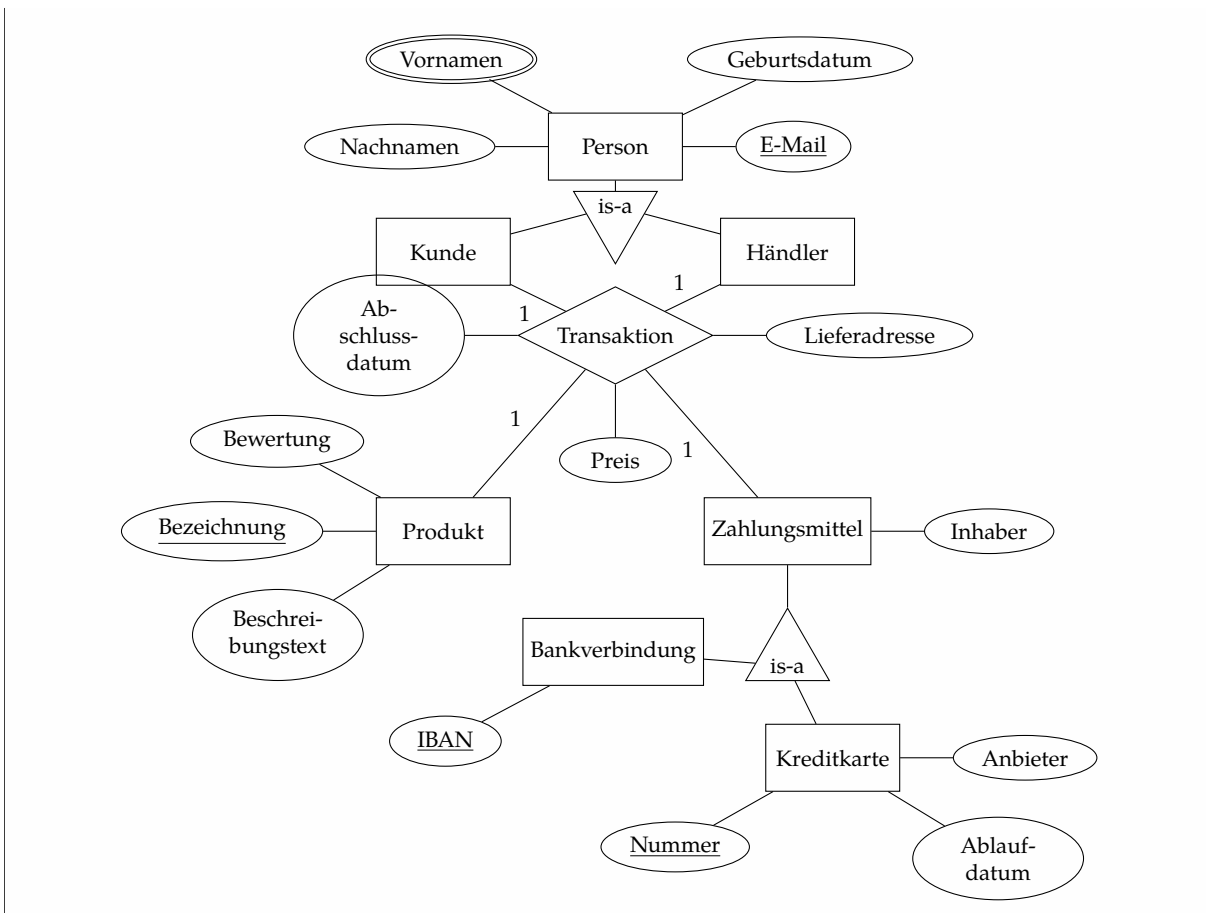
Im Folgenden finden Sie die Beschreibung eines Online-Marktplatzes. Erstellen Sie zu dieser Beschreibung ein erweitertes ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Kennzeichnen Sie auch die totale Teilnahme von Entity-Typen an Beziehungstypen.

Es gibt **Produkte**. Diese haben eine eindeutige *Bezeichnung*, einen *Beschreibungstext* und eine *Bewertung*. Außerdem gibt es **Personen**, die entweder **Kunde**, **Händler** oder beides sind. Jede Person hat einen *Nachnamen*, einen oder mehrere *Vornamen*, ein *Geburtsdatum* und eine *E-Mail-Adresse*, mit der diese eindeutig identifiziert werden kann.

Das System verwaltet außerdem **Zahlungsmittel**. Jedes Zahlungsmittel ist entweder eine **Kreditkarte** oder eine **Bankverbindung** für Lastschriften. Für das Lastschriftverfahren wird die international eindeutige *IBAN* und der Name des *Kontoinhabers* erfasst, bei Zahlung mit Kreditkarte der Name des *Karteninhabers*, die eindeutige *Kartenummer*, das *Ablaufdatum* sowie der *Kartenanbieter*. Es gibt Transaktionen. Jede Transaktion bezieht sich stets auf ein Produkt, einen Kunden, einen Händler und auf ein Zahlungsmittel, das für die Transaktion verwendet wird. Jede Transaktion enthält außerdem den *Preis*, auf den sich Kunde und Händler geeinigt haben, das *Abschlussdatum* sowie eine *Lieferadresse*, an die das Produkt versandt wird.

- ☐ E: **Produkte**
- ☐ A: *Bezeichnung*
- ☐ A: *Beschreibungstext*
- ☐ A: *Bewertung*
- ☐ E: **Personen**
- ☐ E: **Kunde**
- ☐ E: **Händler**
- ☐ A: *Nachnamen*
- ☐ A: *Vornamen*
- ☐ A: *Geburtsdatum*
- ☐ A: *E-Mail-Adresse*
- ☐ E: **Zahlungsmittel**
- ☐ E: **Kreditkarte**
- ☐ E: **Bankverbindung**
- ☐ A: *IBAN*
- ☐ A: *Kontoinhabers*
- ☐ A: *Karteninhabers*
- ☐ A: *Kartenummer*
- ☐ A: *Ablaufdatum*
- ☐ A: *Kartenanbieter*
- ☒ R: Transaktionen
- ☐ A: *Preis*
- ☐ A: *Abschlussdatum*
- ☐ A: *Lieferadresse*

Lösungsvorschlag



### 66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 3

Entwerfen Sie zum untenstehenden ER-Diagramm ein Relationenschema (in dritter Normalform, 3. NF) mit möglichst wenigen Relationen.

Verwenden Sie dabei folgende Notation: Primärschlüssel werden durch Unterstreichen gekennzeichnet, Fremdschlüssel durch die Nennung der Relation, auf die sie verweisen, in eckigen Klammern hinter dem Fremdschlüsselattribut. Attribute zusammengesetzter Fremdschlüssel werden durch runde Klammern als zusammengehörig markiert. Wenn ein Attribut zur korrekten Abbildung des ER-Diagramms als UNIQUE oder NOT NULL ausgezeichnet werden muss, geben Sie dies an.

Beispiel:

Relation1 (Primärschlüssel, Attribut1, Attribut2, Fremdschlüsselattribut1[Relation1], (Fremdschlüssel2 Attribut1, Fremdschlüssel2 Attribut2) [Relation2]); Attribut1 UNIQUE Attribut2 NOT NULL

#### Exkurs: Enhanced entity-relationship model

[https://en.wikipedia.org/wiki/Enhanced\\_entity-relationship\\_model](https://en.wikipedia.org/wiki/Enhanced_entity-relationship_model) <https://www.tutorialride.com/dbms/enhanced-entity-relationship-model-eer-model.htm>

```
Automarke : {[AName, Name[Firma]]}
Fahrzeug : {[Kennzeichen, Mieter-Nr[Mieter], Name[Firma], Modell, Farbe]}
Firma : {[Name, Umsatz, Mieter-Nr]}
Hersteller : {[Name[Firma]]}
Mieter : {[Mieter-Nr]}
Person : {[Handynummer, Vorname, Nachname, Mieter-Nr[Mieter]]}
```

## 66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 4

Gegeben sind folgende Relationen:

Mitarbeiter (MitarbeiterID, Vorname, Nachname, Adresse, Gehalt, Vorgesetzter [Mitarbeiter]  
NOT NULL, AbteilungsID[Abteilung])

Abteilung (AbteilungsID, Bezeichnung UNIQUE NOT NULL)

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz nicht mehrfach aus.

```
CREATE TABLE Abteilung(
 AbteilungsID INTEGER PRIMARY KEY,
 Bezeichnung VARCHAR(30) UNIQUE NOT NULL
);
```

```
CREATE TABLE Mitarbeiter(
 MitarbeiterID INTEGER PRIMARY KEY,
 Vorname VARCHAR(30),
 Nachname VARCHAR(30),
 Adresse VARCHAR(60),
 Gehalt DECIMAL(7, 2),
 Vorgesetzter INTEGER NOT NULL REFERENCES Mitarbeiter(MitarbeiterID),
 AbteilungsID INTEGER REFERENCES Abteilung(AbteilungsID)
);
```

```
INSERT INTO Abteilung VALUES
(1, 'Buchhaltung'),
(42, 'Vertrieb');
```

```
INSERT INTO Mitarbeiter VALUES
(1, 'Karl', 'Landsbach', 'Sigmaringstraße 4, 87153 Farnbach', 2467.23, 1, 42),
(2, 'Lisa', 'Grätzner', 'Scheidplatz 6, 18434 Tullach', 5382.2, 1, 42),
(3, 'Sarah', 'Riedel', 'Am Angera 3, 79527 Töll', 7382.2, 1, 42),
(4, 'Franz', 'Rudolf', 'Strewitzstraße 4, 45507 Strewith', 2382.2, 1, 42),
(5, 'Sergej', 'Puschkin', 'Radolf 4, 12507 Radstadt', 1382.2, 1, 1);
```

- (a) Schreiben Sie eine SQL-Anweisung, die die Tabelle Mitarbeiter anlegt. Gehen Sie davon aus, dass die Tabelle Abteilung bereits existiert.

Lösungsvorschlag

Siehe oben

- (b) Schreiben Sie eine SQL-Anweisung, die Vor- und Nachnamen der Mitarbeiter der Abteilung mit der Bezeichnung Vertrieb ausgibt, absteigend sortiert nach MitarbeiterID.

Lösungsvorschlag

```
SELECT m.Vorname, m.Nachname
FROM Mitarbeiter m, Abteilung a
WHERE
 a.AbteilungsID = m.AbteilungsID AND
 a.Bezeichnung = 'Vertrieb'
ORDER BY m.MitarbeiterID DESC;
```

- (c) Schreiben Sie eine SQL-Anweisung, die Vor- und Nachnamen sowie das Gehalt von Mitarbeitern ausgibt, die mehr verdienen als ihr Chef. Sortieren Sie die Ausgabe absteigend nach dem Gehalt.

Lösungsvorschlag

```
SELECT m.Vorname, m.Nachname, m.Gehalt
FROM Mitarbeiter m, Mitarbeiter n
WHERE
 m.Vorgesetzter = n.MitarbeiterID AND
 m.Gehalt > n.Gehalt
ORDER BY m.Gehalt DESC;
```

- (d) Schreiben Sie eine SQL-Anweisung, die das Gehalt von allen Mitarbeitern aus der Abteilung mit der ID 42 um 10% erhöht.

Lösungsvorschlag

```
UPDATE Mitarbeiter
SET Gehalt = Gehalt * 1.1
WHERE AbteilungsID = 42;
SELECT * FROM Mitarbeiter;
```

- (e) Schreiben Sie eine SQL-Anweisung, welche den Vornamen, die Nachnamen und das Gehalt der sieben bestbezahlten Mitarbeiter aus der Buchhaltung ausgibt. Standardkonforme Sprachkonstrukte, die eine Beschränkung der Ausgabe bewirken, sind erlaubt.

Lösungsvorschlag

```
SELECT m.Vorname, m.Nachname, m.Gehalt
FROM Mitarbeiter m, Mitarbeiter n, Abteilung a
WHERE
 m.Gehalt <= n.Gehalt AND
 a.AbteilungsID = m.AbteilungsID AND
 a.AbteilungsID = n.AbteilungsID AND
 a.Bezeichnung = 'Buchhaltung'
```



```
GROUP BY m.Vorname, m.Nachname, m.Gehalt
HAVING COUNT(*) <= 7
ORDER BY m.Gehalt DESC;
```

- (f) Schreiben Sie eine SQL-Anweisung, die für jede Abteilung die Mitarbeiter ermittelt, die am wenigsten verdienen. Dabei sollen Vorname, Nachname und die Abteilungsbezeichnung der Mitarbeiter ausgegeben werden.

Lösungsvorschlag

```
SELECT m.Vorname, m.Nachname, m.Gehalt, a.Bezeichnung
FROM Mitarbeiter m, Mitarbeiter n, Abteilung a
WHERE
 m.Gehalt >= n.Gehalt AND
 m.AnteilungsID = n.AnteilungsID AND
 m.AnteilungsID = a.AnteilungsID
GROUP BY m.Vorname, m.Nachname, m.Gehalt, m.AnteilungsID, a.Bezeichnung
HAVING COUNT(*) <= 1
ORDER BY m.Gehalt DESC;
```

## 66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 5

Formulieren Sie basierend auf den in der letzten Aufgabe gegebenen Relationen die geforderten Anfragen in der Relationenalgebra.

Mitarbeiter (MitarbeiterID, Vorname, Nachname, Adresse, Gehalt, Vorgesetzter [Mitarbeiter]  
NOT NULL, AbteilungsID[Abteilung])

Abteilung (AbteilungsID, Bezeichnung UNIQUE NOT NULL)

- (a) Formulieren Sie eine Anfrage, welche die Vornamen und Nachnamen der Mitarbeiter ausgibt, die in der Buchhaltung arbeiten.

Lösungsvorschlag

$$\pi_{\text{Vorname, Nachname}} \left( \sigma_{\text{Bezeichnung} = \text{'Buchhaltung'}} \left( \text{Mitarbeiter} \bowtie_{\text{Mitarbeiter.AnteilungsID} = \text{Abteilung.AnteilungsID}} \text{Abteilung} \right) \right)$$

- (b) Formulieren Sie eine Anfrage, welche die Vornamen und Nachnamen der Mitarbeiter ausgibt, die in keiner Abteilung arbeiten.

Lösungsvorschlag

$$\pi_{\text{Vorname, Nachname}}(\text{Mitarbeiter}) - \pi_{\text{Vorname, Nachname}}(\text{Mitarbeiter} \bowtie \text{Abteilung})$$

## 66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 6

Gegeben ist die Relation Prüfung (Prüfungsnummer, Fakultät, Prüfungsname, Dozent, Prüfungstyp, ECTS) mit den beiden Schlüsselkandidaten (Prüfungsnummer, Fakultät) und (Fakultät, Prüfungsname, Dozent).

Alle Attributwerte sind atomar. Es gelten nur die durch die Schlüsselkandidaten implizierten funktionalen Abhängigkeit.

Geben Sie die höchste Normalform an, die die Relation-Prüfung erfüllt. Zeigen Sie, dass alle Bedingungen für diese Normalform erfüllt sind und dass mindestens eine Bedingung der nächsthöheren Normalform verletzt ist. Beziehen Sie sich bei der Begründung auf die gegebene Relation und nennen Sie nicht nur die allgemeinen Definitionen der Normalformen.

Lösungsvorschlag

$$\text{FA} = \left\{ \begin{array}{l} \{ \text{Prüfungsnummer, Fakultät} \} \rightarrow \{ \text{Prüfungsname, Dozent, Prüfungstyp, ECTS} \}, \\ \{ \text{Fakultät, Prüfungsname, Dozent} \} \rightarrow \{ \text{Prüfungsnummer, Prüfungstyp, ECTS} \}, \end{array} \right\}$$

Höchste Normalform: 4NF  
Siehe Taschenbuch Seite 449

**1NF** Alle Werte sind atomar.

**2NF** Ist in 1NF und jedes Attribut ist Teil des Schlüsselkandidaten (Prüfungsnummer, Fakultät oder Fakultät, Prüfungsname, Dozent) oder das Attribut ist von einem Schlüsselkandidaten voll funktional abhängig (Prüfungsname, Dozent, Prüfungstyp, ECTS oder Prüfungsnummer, Prüfungstyp, ECTS).

**3NF** Ist in 2NF und ein Nichtschlüsselattribut darf nur vom Schlüsselkandidaten abhängen (Prüfungsname, Dozent, Prüfungstyp, ECTS hängt von Prüfungsnummer, Fakultät ab) und (Prüfungsnummer, Prüfungstyp, ECTS hängt von Fakultät, Prüfungsname, Dozent ab).

**BCNF** Jede Determinate ist Schlüsselkandidat (Prüfungsnummer, Fakultät und Fakultät, Prüfungsname, Dozent).

**4NF** keine paarweise Unabhängigkeiten mehrwertigen Abhängigkeiten zwischen ihren Attributen.

**66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 7**

- (a) Erläutern Sie kurz, was Indizes sind und warum diese in Datenbanksystemen verwendet werden.

Lösungsvorschlag

Ein Datenbankindex ist eine von der Datenstruktur getrennte Indexstruktur in einer Datenbank, die die Suche und das Sortieren nach bestimmten Feldern beschleunigt.

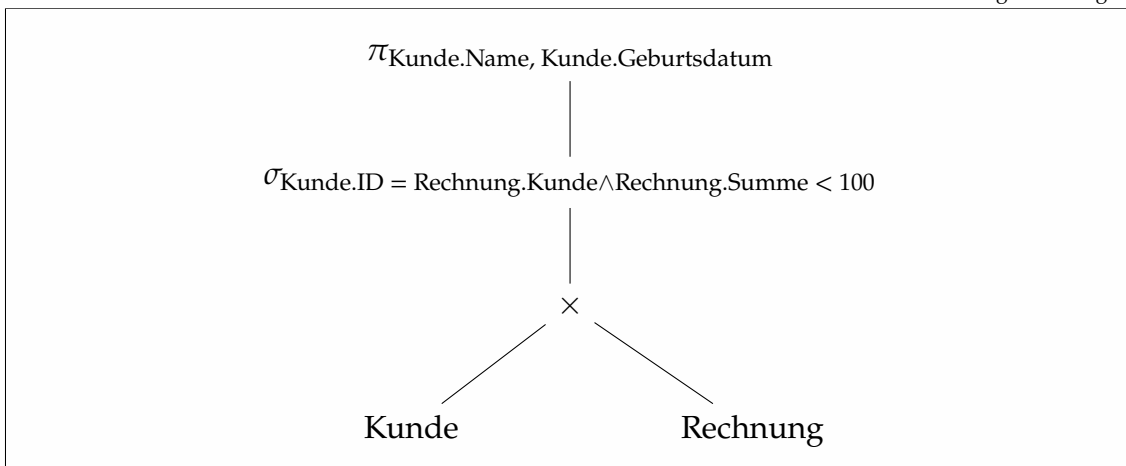
Ein Index besteht aus einer Ansammlung von Zeigern (Verweisen), die eine Ordnungsrelation auf eine oder mehrere Spalten in einer Tabelle definieren. Wird bei einer Abfrage eine indizierte Spalte als Suchkriterium herangezogen, sucht das Datenbankmanagementsystem (DBMS) die gewünschten Datensätze anhand dieser Zeiger. In der Regel finden hier B+-Bäume Anwendung. Ohne Index müsste die Spalte sequenziell durchsucht werden, während eine Suche mit Hilfe des Baums nur logarithmische Komplexität hat.<sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/Datenbankindex>

- (b) Übertragen Sie folgendes SQL-Statement in einen nicht optimierten algebraischen Term oder in einen Anfragegraphen.

```
SELECT Kunde.Name, Kunde.Geburtsdatum
FROM Kunde, Rechnung
WHERE Kunde.ID = Rechnung.Kunde
AND Rechnung.Summe < 100;
```

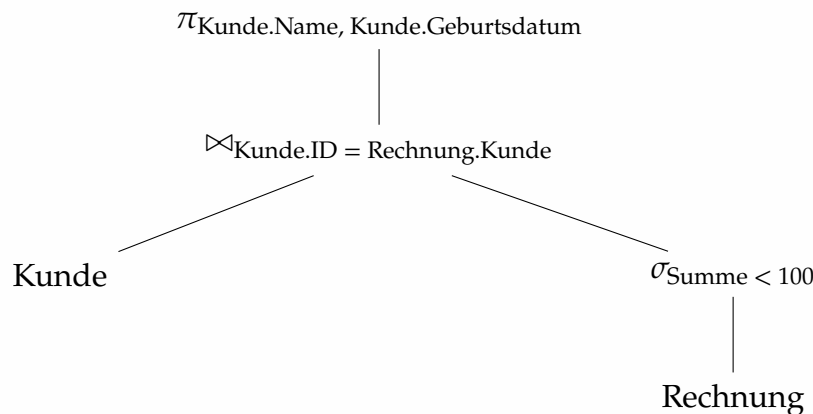
Lösungsvorschlag



- (c) Nennen Sie zwei Möglichkeiten, den algebraischen Term bzw. den Anfragegraphen aus der vorhergehenden Teilaufgabe logisch (d. h. algebraisch) zu optimieren. Beziehen Sie sich auf konkrete Stellen und Operatoren des von Ihnen aufgestellten algebraischen Ausdrucks.

Zuerst Selection und dann Join

Theta-Join



## 66118 (Fachdidaktik (Gymnasium))

### 66118 / 2021 / Frühjahr / Thema 2 / Aufgabe 2

Im LehrplanPLUS finden Sie für den „Lernbereich 3: Grundlagen der objektorientierten Modellierung und Programmierung (ca. 26 Stunden)“ für die 9. Jahrgangsstufe folgenden Text:

#### Kompetenzerwartungen:

Die Schülerinnen und Schüler...

- analysieren Objekte aus ihrer Erfahrungswelt (z. B. Fahrzeuge, Personen) hinsichtlich ihrer Eigenschaften (Attribute) und Fähigkeiten (Methoden) und abstrahieren sie zu Klassen. Sie stellen Objekte und Klassen als Grundlage einer möglichen Implementierung grafisch dar.
- deklarieren eine Klasse sowie die zugehörigen Attribute und Methoden in einer objektorientierten Programmiersprache.
- verwenden bei der Implementierung Wertzuweisungen, um Attributwerte zu ändern, und interpretieren diese als Zustandsänderung des zugehörigen Objekts.
- formulieren unter Verwendung der Kontrollstrukturen Algorithmen zu geeigneten Problemstellungen, u. a. durch grafische Darstellungen.
- implementieren Methoden auf der Grundlage gegebener Algorithmen objektorientiert, wobei sie sich des Unterschiedes zwischen Methodendefiniti-

on und Methodenaufruf bewusst sind. Dabei nutzen sie ggf. auch Methoden anderer Klassen.

- analysieren, interpretieren und modifizieren Algorithmen, wodurch sie die Fähigkeit erlangen, fremde Programme flexibel einzusetzen und kritisch zu bewerten.
- modellieren durch Klassendiagramme einfache Generalisierungshierarchien zu geeigneten Strukturen aus ihrer Erfahrungswelt.
- implementieren mithilfe einer objektorientierten Sprache einfache Generalisierungshierarchien; dabei nutzen sie das Konzept der Vererbung sowie die Möglichkeit, Methoden zu überschreiben

#### **Inhalte zu den Kompetenzen:**

- objektorientierte Konzepte, u. a. Objekt, Klasse, Attribut, Attributwert, Methode
- Variablenkonzept; Arten von Variablen: Parameter, lokale Variable und Attribut; Übergabewert
- Wertzuweisung zur Änderung von Variablenwerten
- Methoden: Methodenkopf, Methodenrumpf, Methodendefinition, Methodenaufruf, Übergabewert, Rückgabewert; Konstruktor als spezielle Methode; Standardmethoden zum Geben und Setzen von Attributwerten
- Algorithmus: Strukturelemente, grafische Darstellung, Pseudocode
- Datentypen: ganze Zahlen, Gleitkommazahlen, Wahrheitswerte, Zeichen, Zeichenketten
- Generalisierung und Spezialisierung: Ober- und Unterklasse, Vererbung von Attributen und Methoden an Unterklassen, Überschreiben von Methoden
- Fachbegriffe: Parameter, Übergabewert, Rückgabewert, lokale Variable, Wertzuweisung, Konstruktor, Methodenkopf, Methodenrumpf, Vererbung, Generalisierung, Spezialisierung, Oberklasse, Unterklasse

#### **Aufgabe**

- (a) In obigem Lehrplanabschnitt wird mehrfach „Modellierung“ und „Programmierung“ genannt.
- (i) Grenzen Sie die beiden Begriffe gegeneinander ab und begründen Sie, weshalb man beides im Informatikunterricht benötigt.
  - (ii) Erläutern Sie das Konzept der „Fundamentalen Ideen“ nach Schwill! Nehmen Sie dabei Bezug auf „Modellierung“ und „Programmierung“.

- (b) Im Informatikunterricht ist sowohl der Einsatz einer blockbasierten als auch einer textbasierten Sprache denkbar.
- (i) Gehen Sie jeweils auf Vor- und Nachteile der beiden Möglichkeiten anhand konkreter Beispiele ein.
  - (ii) Entscheiden Sie dann begründet, welche Wahl Sie in der 9. Jahrgangsstufe treffen würden.
- (c) Im zitierten Lehrplanabschnitt ist die Verwendung von Kontrollstrukturen zur Formulierung von Algorithmen vorgesehen. Es gibt aber keine konkrete Auflistung, welche Kontrollstrukturen besprochen werden sollen.
- (i) Geben Sie einen kurzen Überblick über die Kontrollstrukturen imperativer bzw. objektorientierter Programmiersprachen an, die hier fachlich in Frage kommen könnten.
  - (ii) Im Hinblick darauf, dass für den gesamten Lehrplanabschnitt 26 Unterrichtsstunden zur Verfügung stehen, kann es notwendig sein, sich auf wenige Kontrollstrukturen beschränken zu müssen. Entscheiden Sie, welche Kontrollstrukturen Sie wählen würden und erklären Sie, in welcher Reihenfolge Sie diese in der 9. Jahrgangsstufe einführen würden! Begründen Sie Ihre Ausführungen.
- (d) Erstellen Sie für den zitierten Lehrplanabschnitt einen Sequenzplan mit 13 Doppelstunden! Geben Sie für jede dieser Doppelstunden eine kurze, nachvollziehbare Beschreibung der jeweiligen Zielsetzung an.
- (e) Erstellen Sie eine Feinplanung für eine Doppelstunde zur bedingten Wiederholung unter Verwendung der von Ihnen in Aufgabe 2b) gewählten Art der Programmiersprache.
- (i) Legen Sie zunächst drei beobachtbare Lernziele fest.
  - (ii) Skizzieren Sie eine Einführungsaufgabe, die die Schülerinnen und Schüler zu diesem Thema bearbeiten sollen.
  - (iii) Erläutern Sie anschließend den Unterrichtsfortgang nachvollziehbar (textuelle Beschreibung). Begründen Sie dabei Ihre fachdidaktischen Entscheidungen und gruppieren Sie Ihren Text nach Unterrichtsphasen.

## 66118 / 2021 / Frühjahr / Thema 2 / Aufgabe 3

Der LehrplanPLUS der 7. Jahrgangsstufe (Natur und Technik) des neunjährigen Gymnasiums enthält den folgenden Lehrplanpunkt:

NT7 2.3 Beschreibung von Abläufen durch Algorithmen (ca. 11 Std.)

### Kompetenzerwartungen

Die Schülerinnen und Schüler ...

- analysieren und strukturieren geeignete Problemstellungen u. a. aus ihrer Erfahrungswelt (z. B. Bedienung eines Geräts), entwickeln Algorithmen zu deren Lösung und beschreiben diese unter effizienter Verwendung von Kontrollstrukturen.
- setzen unter sinnvoller Nutzung algorithmischer Bausteine einfache Algorithmen mithilfe geeigneter Programmierwerkzeuge um.

**Inhalte zu den Kompetenzen:**

- Algorithmus: Definition des Begriffs, Strukturelemente (Anweisung, Sequenz, ein- und zweiseitig bedingte Anweisung, Wiederholung mit fester Anzahl, Wiederholung mit Bedingung)
- Fachbegriffe: Algorithmus, Anweisung, Sequenz, ein- und zweiseitig bedingte Anweisung, Wiederholung mit fester Anzahl, Wiederholung mit Bedingung

## Aufgabe

Gehen Sie bei den folgenden Aufgaben von folgendem Szenario aus:

Ihrer Schule wurde eine Sachspende über einen Klassensatz programmierbare Roboter angeboten. Dabei handelt es sich um fertig montierte, fahrbare Kleinroboter mit je zwei Abstandssensoren (siehe Abbildung).

Der Roboter verfügt über zwei unabhängige Motoren, deren Geschwindigkeit gesteuert werden kann. Kurven können gefahren werden, indem die Motoren unterschiedlich schnell laufen. Zur Programmierung steht eine einfache Umgebung zur Verfügung.

- (a) Der oben zitierte Lehrplanpunkt wird oftmals anhand einer Software mit einem simulierten, steuerbaren Roboter — z.B. Robot Karol — unterrichtet. Diskutieren Sie Unterschiede zwischen dieser Simulation und dem Einsatz der beschriebenen realen Roboter aus fachdidaktischer Sicht. Beschreiben Sie, welchen Ansprüchen die Programmierungsumgebung genügen sollte, damit ein Unterrichtseinsatz in Natur und Technik zu obigem Lehrplanpunkt möglich und sinnvoll ist.

Der größte Unterschied ist sicherlich, dass der hier beschriebene Roboter real, also anfassbar ist. SuS können unmittelbar in der Realität erfahren, was ihre Programmierung auslöst. Dies ist vor allem für solche SuS vorteilhaft, die einen Computer und dessen Programme als eine Art Blackbox verstehen, d.h. die z.B. Robot Karol als Spielzeug ansehen, das keinen Bezug zur Lebenswelt hat, da er nur im Rechner existiert. Zusätzlich sehe ich den Roboter aus oben genanntem Grund als genderneutraler als Robot Karol, der wohl eher die computerbegeisterten Jungs in der Klasse ansprechen wird.

Darüber hinaus kann man anhand des Roboters Fragen aus den Bereich Technik aufnehmen, wie z.B. „Wie funktioniert eine Lenkung im Auto?“ im Ge-



gensatz zur Frage „Wie kann man mit einer starren Achse wie z.B. bei einem Rollstuhl lenken?“. Solche Querverweise sind bei Robot Karol nicht möglich.

Für Robot Karol spricht hingegen, dass die Möglichkeiten des Programmierens vielfältiger sind. So kann der reale Roboter z.B. keine Steine „hinlegen“. Somit erscheint mir auf den ersten Blick Robot Karol abwechslungsreicher zu sein.

Ein weiterer Unterschied könnte darin liegen, dass Robot Karol eine textbasierte Programmierung erfordert, wohingegen der reale Roboter blockbasiert funktionieren könnte (je nach Hersteller). Der große Vorteil blockbasierter Programmiersprachen wie z.B. Snap! liegt beim Einstieg in das Thema „Programmierung“ darin, dass sich die SuS auf die Art und Weise, wie programmiert wird, konzentrieren können und sich keine Gedanken darüber machen müssen, wie nun der richtige Befehl lautet. Sie können sich also auf die Strukturen des Programmierens konzentrieren.

### Ansprüche an die Programmierungsumgebung

Der Roboter sollte eine einfache, intuitiv strukturierte Benutzeroberfläche aufweisen. Vorteilhaft wäre hierfür eine App für Tablets, die sich über das WLAN der Schule mit den Robotern verbinden kann. Die App sollte blockbasiert sein, wobei es für die einzelnen Fähigkeiten des Roboters vorgefertigte Bricks geben muss. Darüber hinaus müssen die im Lehrplan geforderten Sequenzen, bedingte Anweisungen und Wiederholungen möglich sein. Die erstellten Programme sollten für jede SuS speicherbar sein, so dass in der Folgestunde weitergearbeitet werden kann.

- (b) Geben Sie eine Grobplanung der Unterrichtssequenz für oben zitierten Lehrplanpunkt unter Nutzung der realen Roboter an. Gehen Sie dabei von insgesamt sechs Doppelstunden aus! Nennen Sie für jede Stunde ein beobachtbares Feinziel.

Lösungsvorschlag

Anmerkung: Jede Doppelstunde besteht nach der in folgenden beschriebenen Wissensvermittlung per Lehrerinput aus ausreichend Aufgaben, die die SuS selbstständig lösen sollen. Die Schüleraktivierungsphasen werden nicht explizit in der folgenden Grobübersicht erwähnt, sind aber wesentlicher Bestandteil der Doppelstunde.

**1.DS: Einführung in die Bedienung des Roboters** SuS erhalten einen Grobübersicht über die Funktionsmöglichkeiten des Roboters und die einzelnen Bricks zur Programmierung in der zugehörigen App -> Klärung der Fachbegriffe Anweisung und Sequenz Feinziel: SuS sind am Ende der Stunde in der Lage, einfache Bewegungen des Roboters zu programmieren und auszuführen.

**2.DS: Wiederholung mit fester Anzahl** SuS lernen, mit Wiederholungen Programmsequenzen mehrfach hintereinander auszuführen. Auch eine Schach-

telung von Wiederholungen werden gelernt und deren Sinn erarbeitet  
-> Klärung des Fachbegriffs Wiederholung mit fester Anzahl Feinziel:  
SuS sind am Ende der Stunde in der Lage sein, den Roboter mit Hilfe  
von Wiederholungen z.B. Quadrate und Rechtecke automatisch fahren  
zu lassen.

**3.DS: Wiederholung mit Bedingung** SuS lernen mögliche Bedingungen kennen und die Art, wie man mit diesen programmieren kann. SuS erkennen den Vorteil gegenüber der Wiederholung mit fester Anzahl erarbeitet. SuS lernen den Begriff Algorithmus im Zusammenhang mit ihren Programmsequenzen kennen -> Klärung der Fachbegriffe Wiederholung mit Bedingung und Algorithmus Feinziel: SuS sind am Ende der Stunde in der Lage sein, den Roboter mit Hilfe von Wiederholungen mit Bedingungen zu programmieren.

**4.DS: Aufgabenstunde** SuS erhalten Aufgaben (Schwierigkeit variabel), die sie mit den bisherigen Unterrichtsinhalten lösen können. Jede SuS wählt Anfangsschwierigkeit selbst aus. Feinziel: SuS sind am Ende der Stunde in der Lage sein, vorgefertigte Probleme mit einem eigenen Programm lösen zu können.

**5.DS: bedingte Anweisung** SuS lernen die Möglichkeit kennen, dem Roboter mögliche Alternativen im Programmablauf zu programmieren erarbeitet -> Klärung des Fachbegriffs ein- und zweiseitig bedingte Anweisung Feinziel: SuS sind am Ende der Stunde in der Lage sein, zweiseitig bedingte Anweisungen programmieren zu können

**6.DS: Zusammenfassende Aufgabenstunde** Vgl. 4. DS

- (c) Entwerfen Sie einen schriftlichen Leistungsnachweis mit Lösungsskizze für eine Bearbeitungszeit von 20 Minuten. Geben Sie an, an welcher Stelle der Unterrichtssequenz Sie ihn einsetzen würden.

Lösungsvorschlag

Test zu Beginn der 6.DS zur Überprüfung der Programmierfähigkeiten der SuS

**Aufgabe 1:** Erkläre den Unterschied zwischen einer einseitig und einer zweiseitig bedingten Anweisung!

**Aufgabe 2:** Schreibe in Anlehnung an den Unterricht die Hilfsprogramme „Linksdrehen“ und „Umdrehen“ auf!

**Aufgabe 3:** Unser Roboter soll das Ende eines verwinkelten Ganges finden. Der Gang ist seitlich durch Mauern begrenzt und kann sich nur um 90° nach links bzw. nach rechts biegen (keine Verzweigungen!!!!). Das Ende ist durch eine Sackgasse gegeben. Mögliches Beispiel:  
Schreibe ein Programm, mit dem der Roboter für alle möglichen Gänge dieser Art den Endpunkt findet. Verwende hierbei sowohl Wiederholungen mit Bedingungen als auch zweiseitig bedingte Anweisungen!

## Lösungsskizze:

**Zu 2:** Linksdrehen soll den Roboter um  $90^\circ$  nach links drehen. Dabei bewegt sich das rechte Rad vorwärts, das linke rückwärts. Umdrehen ist ein zweimaliges Linksdrehen!

**Zu 3:** Die SuS können auch die aus dem Programm bekannten Bricks darstellen!

```

Wiederhole solange NichtIstWand
 vorwärts
endeWiederhole
linksdrehen
Wenn IstWand
 Dann Umdrehen
 Wenn IstWand
 Dann fertig
 Sonst vorwärts
 endeWenn
Sonst vorwärts
endeWenn

```

- (d) Geben Sie für den Einstieg in die Unterrichtssequenz (erste Doppelstunde) eine Feinplanung an. Beschreiben Sie dabei den geplanten Ablauf detailliert und skizzieren Sie Tafelbilder, Hefteinträge, Arbeitsblätter o. Ä. (ggf. mit einer Musterlösung).

Lösungsvorschlag

**Einstieg/Motivation (5Min)** Zeigen eines Videos zum Thema „Roboter in der Wirtschaft“ (z.B. in der Autoproduktion) Vorführen des schuleigenen Roboters

**Kennenlernphase (15Min)** Benutzeroberfläche der App wird per Beamer an die Wand projiziert und die einzelnen grundlegenden Bestandteile der App wird im Lehrervortrag erklärt. Arbeitsblatt mit Screenshot der App und einzelnen Bestandteile zum Ausfüllen durch die Schüler (hier nicht möglich, den Screenshot anzugeben) Die Begriffe Anweisung und Sequenz werden erklärt -> Tafelbild und Hefteintrag

**Ausprobierphase (25Min)** Die SuS erhalten einen Roboter, verbinden diesen mit der Bedienoberfläche wie in der Kennenlernphase beschrieben und führen erste Anweisungen aus (von SuS frei wählbar). Arbeitsauftrag: Wie kann sich der Roboter rechtsdrehen? -> Gruppen- bzw. Einzelarbeit (nach Anzahl der Roboter)

**Erarbeitungsphase (30Min)** SuS sollen Sequenzen zu folgenden Aufgaben schreiben: Mein Roboter fährt einen Kreis mit 70 cm Radius. Mein Roboter fährt ein Rechteck mit Kantenlängen 40 und 60 cm. Wettbewerb: Welche Gruppe setzt die Vorgaben am besten um?

**Wissenssicherungs- und Aufräumphase (10Min)** SuS geben folgende Inhalte wiederholend in eigenen Worten wieder: Anweisung – Sequenz- grobe Steuerelemente des Roboters Aufräumen der Roboter

- (e) Geben Sie weitere Stellen des bayerischen Lehrplans für das Fach Informatik am neunjährigen Gymnasium (NTG) an, an denen Roboter dieser oder ähnlicher Art sinnvoll eingesetzt werden könnten. Bewerten Sie in jedem Fall knapp, ob die eingangs beschriebenen Roboter dafür geeignet wären und nennen Sie ggf. Alternativen.

Lösungsvorschlag

Der Einsatz eines Roboters wäre vielleicht noch in der 9. Jahrgangsstufe bei der Einführung in die objektorientierte Programmierung möglich. Dabei kann anhand des Roboters folgende Lehrplaninhalte eingeführt werden:

analysieren Objekte aus ihrer Erfahrungswelt (z. B. Fahrzeuge, Personen) hinsichtlich ihrer Eigenschaften (Attribute) und Fähigkeiten (Methoden) und abstrahieren sie zu Klassen. Sie stellen Objekte und Klassen als Grundlage einer möglichen Implementierung grafisch dar.

deklarieren eine Klasse sowie die zugehörigen Attribute und Methoden in einer objektorientierten Programmiersprache.

verwenden bei der Implementierung Wertzuweisungen, um Attributwerte zu ändern, und interpretieren diese als Zustandsänderung des zugehörigen Objekts.

Um kompliziertere Vorgänge beschreiben zu können, erscheint der in der Aufgabe beschriebene Roboter zu limitiert zu sein. So haben z.B. Lego Mindstorm-Roboter aufgrund ihrer vielfältigen Sensoren viel mehr Möglichkeiten, weshalb diese wohl viel ansprechender für SuS der 9. Klasse sein werden.