Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____		
Kennwort: _____	**Herbst 2020**	**66115**
Arbeitsplatz-Nr.: _____		

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach:	**Informatik (vertieft studiert)**
Einzelprüfung:	**Theoretische Informatik, Algorithmen**
Anzahl der gestellten Themen (Aufgaben):	2
Anzahl der Druckseiten dieser Vorlage:	15

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Theoretische Informatik

Aufgabe 1 [40 PUNKTE]

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.

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.

c) Das Komplement einer kontextfreien Sprache ist wieder kontextfrei.

d) Wenn ein Problem unentscheidbar ist, dann ist es nicht semientscheidbar.

e) Sei f eine totale Funktion. Dann gibt es ein WHILE-Programm, das diese berechnet.

f) Das Halteproblem für LOOP-Programme ist entscheidbar.

g) Die Komplexitätsklasse NP enthält genau die Entscheidungsprobleme, die in nichtpolynomieller Zeit entscheidbar sind.

h) Falls P $\neq$ NP, dann gibt es keine NP-vollständigen Probleme, die in P liegen.

Aufgabe 2 [20 PUNKTE]

Sei $\Sigma = \{x, y, z\}$. Sei $L = (x^*yzx^*)^* \subseteq \Sigma^*$.

a) Geben Sie einen endlichen (deterministischen oder nichtdeterministischen) Automaten A an, der L erkennt bzw. akzeptiert.

b) Geben Sie eine reguläre und eindeutige Grammatik G an, die L erzeugt.

Aufgabe 3 [25 PUNKTE]

Seien $\Sigma = \{a, b, c\}$ und $L = \{wc\widehat{w} \mid w \in \{a, b\}^*\}$. Dabei ist $\widehat{w}$ das zu w gespiegelte Wort.

a) Zeigen Sie, dass L nicht regulär ist.

b) Zeigen Sie, dass L kontextfrei ist, indem Sie eine geeignete Grammatik angeben und anschließend begründen, dass diese die Sprache L erzeugt.

Fortsetzung nächste Seite!

Aufgabe 4 [15 PUNKTE]

Geben Sie für jede der folgenden Mengen an, ob sie entscheidbar ist oder nicht. Dabei ist φ_w die Funktion, die von der Turingmaschine berechnet wird, die durch das Wort w kodiert wird. Beweisen Sie Ihre Behauptungen.

a) $L_1 = \{w \in \Sigma^* \mid \varphi_w(0) = 0\}$

b) $L_2 = \{w \in \Sigma^* \mid \varphi_w(w) = w\}$

c) $L_3 = \{w \in \Sigma^* \mid \varphi_0(0) = w\}$

Aufgabe 5 [20 PUNKTE]

Sei 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 $\Longrightarrow$ als Operationszeichen aufgebaut sind, wobei natürlich Klammern zugelassen sind. Beachten Sie, dass $x_i \Longrightarrow x_j$ die gleiche Wahrheitstabelle wie $\neg x_i \vee x_j$ hat.

Wir betrachten das Problem ISAT. Eine Formel $F \in$ IF ist genau dann in 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: ISAT ist NP-vollständig. Sie dürfen benutzen, dass das SAT-Problem NP-vollständig ist.

Fortsetzung nächste Seite!

Teilaufgabe II: Algorithmen

Aufgabe 1 (Algorithmenanalyse) [32 PUNKTE]

Betrachten Sie die folgende Prozedur **countup**, die aus zwei ganzzahligen Eingabewerten n und m einen ganzzahligen Ausgabewert berechnet:

```
1  procedure countup(n, m : integer) : integer
2  var x, y : integer;
3  begin
4    x := n;
5    y := 0;
6    while (y < m) do
7      x := x − 1;
8      y := y + 1;
9    end while
10   return x;
11 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.

b) Gibt es Eingabewerte von n und m, für die die Prozedur **countup** nicht terminiert? Begründen Sie Ihre Antwort.

c) Geben Sie die asymptotische worst-case Laufzeit der Prozedur **countup** in der Θ-Notation in Abhängigkeit von den Eingabewerten n und/oder m an. Begründen Sie Ihre Antwort.

Betrachten Sie nun die folgende Prozedur **countdown**, die aus zwei ganzzahligen Eingabewerten n und m einen ganzzahligen Ausgabewert berechnet:

```
1  procedure countdown(n, m : integer) : integer
2  var x, y : integer;
3  begin
4    x := n;
5    y := 0;
6    while (n > 0) do
7      if (y < m) then
8        x := x − 1;
9        y := y + 1;
10     else
11       y := 0;
12       n := n / 2; /* Ganzzahldivision */
13     end if
14   end while
15   return x;
16 end
```

Fortsetzung nächste Seite!

d) 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.

e) Gibt es Eingabewerte von n und m, für die die Prozedur **countdown** nicht terminiert? Begründen Sie Ihre Antwort.

f) Geben Sie die asymptotische Laufzeit der Prozedur **countdown** in der Θ-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.

Aufgabe 2 (Bäume) [29 PUNKTE]

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

Der folgende rekursive Algorithmus sei gegeben:

```
1  procedure magic(i, n : integer) : boolean
2  begin
3    if (i > (n − 2) / 2) then
4      return true;
5    end if
6    if (A[i] ≤ A[2∗i + 1] and A[i] ≤ A[2∗i + 2] and
7        magic(2∗i + 1, n) and magic(2∗i + 2, n)) then
8      return true;
9    end if
10   return false;
11 end
```

Fortsetzung nächste Seite!

b) 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?

c) Wie nennt man die Eigenschaft, die der Algorithmus **magic** auf dem Feld **A** prüft? Wie lässt sich diese Eigenschaft formal beschreiben?

d) Welche Ausgaben sind durch den Algorithmus **magic** möglich, wenn das Eingabefeld aufsteigend sortiert ist? Begründen Sie Ihre Antwort.

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.

f) Betrachten Sie folgende Variante **almostmagic** der oben bereits erwähnten Prozedur **magic**, bei der die Anweisungen in Zeilen 3 bis 5 entfernt wurden:

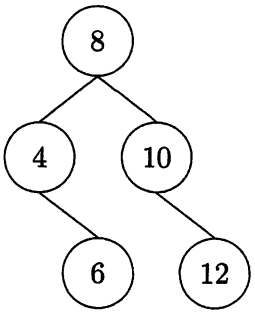
```
1  procedure almostmagic(i, n : integer) : boolean
2  begin
3    // leer
4    // leer
5    // leer
6    if (A[i] ≤ A[2*i + 1] and A[i] ≤ A[2*i + 2] and
7        magic(2*i + 1, n) and magic(2*i + 2, n)) then
8      return true;
9    end if
10   return false;
11 end
```

Beschreiben Sie die Umstände, die auftreten können, wenn **almostmagic** auf einem Feld der Größe **n** aufgerufen wird. Welchen Zweck erfüllt die entfernte bedingte Anweisung?

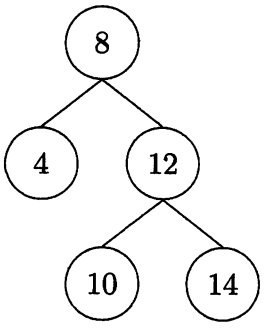
Fortsetzung nächste Seite!

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.

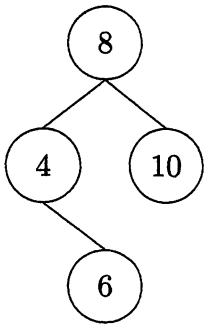
i.) Wert 7 einfügen



ii.) Wert 11 einfügen



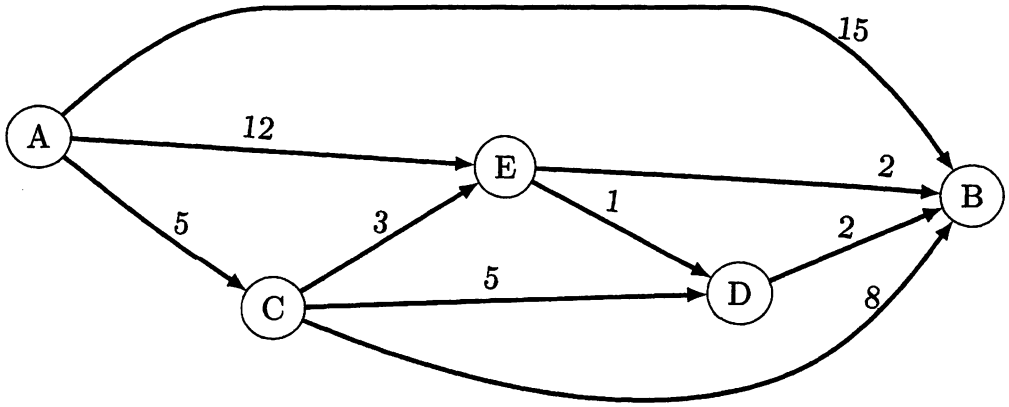
iii.) Wert 5 einfügen



Fortsetzung nächste Seite!

Aufgabe 3 (Kürzeste Wege) [20 PUNKTE]

Gegeben sei der folgende gerichtete und gewichtete Graph G:



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).

A	B	C	D	E	Prio-Queue
$\underline{0}$	∞	∞	∞	∞	[A]
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

b) Geben Sie den kürzesten Pfad vom Knoten A zum Knoten B an.

Aufgabe 4 (O-Notation) [14 PUNKTE]

a) Betrachten Sie das folgende Code-Beispiel (in Java-Notation):

```
1  int mystery(int n) {
2    int a = 0, b = 0;
3    int i = 0;
4    while (i < n) {
5      a = b + i;
6      b = a;
7      i = i + 1;
8    }
9    return a;
10 }
```

Bestimmen Sie die asymptotische worst-case Laufzeit des Code-Beispiels in O-Notation bezüglich der Problemgröße n. Begründen Sie Ihre Antwort.

Fortsetzung nächste Seite!

b) Betrachten Sie das folgende Code-Beispiel (in Java-Notation):

```
1  int mystery(int n) {
2    int r = 0;
3    while (n > 0) {
4      int y = n;
5      int x = n;
6      for (int i = 0; i < y; i++) {
7        for (int j = 0; j < i; j++) {
8          r = r + 1;
9        }
10       r = r - 1;
11     }
12     n = n - 1;
13   }
14   return r;
15 }
```

Bestimmen Sie für das Code-Beispiel die asymptotische worst-case Laufzeit in O-Notation bezüglich der Problemgröße n. Begründen Sie Ihre Antwort.

c) Bestimmen Sie eine asymptotische Lösung (in Θ-Schreibweise) für die folgende Rekursionsgleichung:
$T(n) = T(\frac{n}{2}) + \frac{1}{2}n^2 + n$

Begründen Sie Ihre Antwort.

Aufgabe 5 (Streuspeicherung) [25 PUNKTE]

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:

Fortsetzung nächste Seite!

Fach	Schlüssel k (verkettete Liste, zuletzt eingetragener Schlüssel rechts)
0	
1	
2	
3	
4	
5	
6	
7	

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.

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 $h_1, h_2, \ldots$ 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.

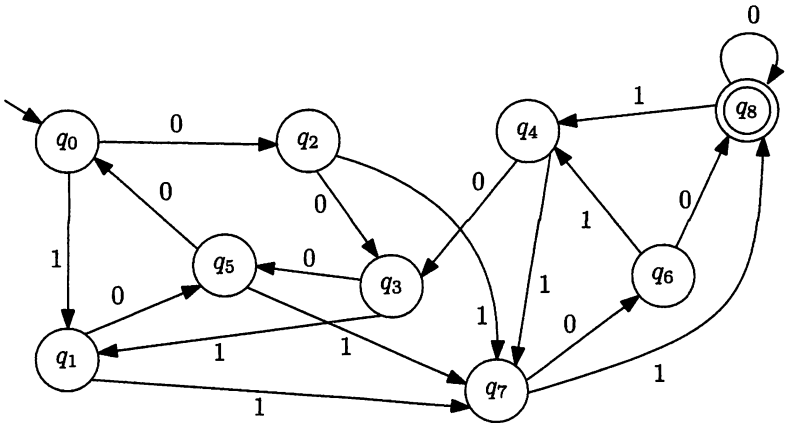
Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Theoretische Informatik

Aufgabe 1 (Reguläre Sprachen) [35 PUNKTE]

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.



b) Beweisen oder widerlegen Sie für folgende Sprachen über dem Alphabet $\Sigma = \{a, b, c\}$, dass sie regulär sind.

(i) $L_1 = \{a^i cub^j vac^k : u, v \in \{a, b\}^* \text{ und } i, j, k \in \mathbb{N}_0\}$

(ii) $L_2 = \{a^i cub^j vac^k : u, v \in \{a, b\}^* \text{ und } i, j, k \in \mathbb{N}_0 \text{ mit } k = i + j\}$

c) Sei L eine reguläre Sprache über dem Alphabet Σ. 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.

Fortsetzung nächste Seite!

Aufgabe 2 (Kontextfreie Sprachen) [25 PUNKTE]

a) Sei $L = \{0^n1^m1^p0^q \mid n+m = p+q,\ n,m,p,q \in \mathbb{N}_0\}$. Geben Sie eine kontextfreie Grammatik für L an. Sie dürfen dabei ε-Produktionen der Form $A \to \varepsilon$ verwenden.

b) Für eine Sprache L sei $L^r = \{x^r \mid x \in L\}$ die Umkehrsprache. Dabei bezeichne x^r das Wort, das aus r 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.

Aufgabe 3 (Komplexitätstheorie) [30 PUNKTE]

Das Problem SUBSETSUM besteht darin zu entscheiden, ob aus einer gegebenen Menge von ganzen Zahlen eine Teilmenge ausgewählt werden kann, die sich genau zu einer gegebenen Zahl K addiert. Formal besteht eine Instanz von SUBSETSUM aus einem Tupel (X, K) mit $X = \{a_1, \ldots, a_n\}$, $K \in \mathbb{Z}$ und gesucht ist eine Menge $A \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in A} a_i = K$.

Im Gegensatz dazu ist das Problem CHOICE-OF-2-SUBSETSUM wie folgt definiert: Eine Instanz (X, M) von CHOICE-OF-2-SUBSETSUM besteht aus einer Menge X von Paaren von ganzen Zahlen $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n) \in \mathbb{Z}^2$ und einer Zahl $M \in \mathbb{Z}$. Die Frage besteht darin, ob es möglich ist, aus jedem der Tupel entweder die Zahl a_i oder die Zahl b_i auszuwählen, sodass die Summe der ausgewählten Zahlen genau M ergibt. Formal sind also Mengen A, B gesucht mit $A \cap B = \emptyset$ und $A \cup B = \{1, \ldots, n\}$, sodass $\sum_{i \in A} a_i + \sum_{i \in B} b_i = M$.

a) Zeigen Sie, dass CHOICE-OF-2-SUBSETSUM in NP liegt.

b) Geben Sie eine totale, in polynomieller Zeit berechenbare Reduktionsfunktion f von SUBSETSUM auf CHOICE-OF-2-SUBSETSUM an.

(Ein Nachweis der Totalität, und dass f in polynomieller Zeit berechenbar ist, ist nicht nötig.)

c) Zeigen Sie, falls (X, K) in SUBSETSUM, so ist $f(X, K)$ in CHOICE-OF-2-SUBSETSUM.

d) Zeigen Sie, falls $f(X, K)$ in CHOICE-OF-2-SUBSETSUM, so ist (X, K) in SUBSETSUM.

Aufgabe 4 (Berechenbarkeit) [30 PUNKTE]

Entscheiden Sie für die folgenden Sprachen, ob sie entscheidbar sind. Beweisen Sie jeweils Ihre Antwort. Dabei bezeichne $< M >$ die Gödelnummer der Turingmaschine M.

a) $L_1 = \{< M > \mid M$ akzeptiert mindestens eine Primzahl$\}$

b) $L_2 = \{< M > \mid$ jeder Übergang von M bewegt den Lesekopf nach rechts und M hält für das Eingabewort $aab\}$

Fortsetzung nächste Seite!

Teilaufgabe II: Algorithmen

Aufgabe 1 (Hashing) [20 PUNKTE]

Eine Hashfunktion h wird verwendet, um Vornamen auf die Buchstaben $\{A, \ldots, Z\}$ abzubilden. Dabei bildet h auf den ersten Buchstaben des Vornamens als Hashwert ab.

Sei H die folgende Hashtabelle (Ausgangszustand):

Schlüssel	Inhalt	Schlüssel	Inhalt
A		N	
B		O	
C		P	
D	Dirk	Q	
E		R	
F		S	
G		T	
H		U	
I	Inge	V	
J		W	
K	Kurt	X	
L		Y	
M		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.

Fortsetzung nächste Seite!

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.)

Aufgabe 2 (O-Notation) [20 PUNKTE]

Beweisen Sie die folgenden Aussagen:

a) Sei $f(n) = 2 \cdot n^3 + 3 \cdot n^2 + 4 \cdot (\log_2 n) + 7$. Dann gilt $f \in O(n^3)$.

b) Sei $f(n) = 4^n$. Dann gilt **nicht** $f \in O(2^n)$.

c) Sei $f(n) = (n+1)!$ (d. h. die Fakultät von $n+1$). Dann gilt $f \in O(n^n)$.

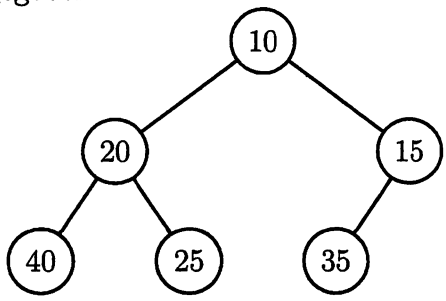
d) Sei $f : \mathbb{N} \to \mathbb{N}$ definiert durch die folgende Rekursionsgleichung:

$$f(n) = \begin{cases} 3, & \text{für } n = 1 \\ (n-1)^2 + f(n-1), & \text{für } n > 1 \end{cases}$$

Dann gilt $f \in O(n^3)$.

Aufgabe 3 (Heap) [20 PUNKTE]

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.

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.

Fortsetzung nächste Seite!

Aufgabe 4 (Dynamische Programmierung) [24 PUNKTE]

Das GUTSCHEIN-Problem ist gegeben durch eine Folge $w_1, \ldots, w_n$ von Warenwerten (wobei $w \in \mathbb{N}_0$ für $i = 1, \ldots, n$) 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 $I \subseteq \{1, \ldots, n\}$ gibt, sodass $\sum_{i \in I} w_i = G$ gilt.

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.
   (ii) Geben Sie einen Gutscheinbetrag G mit $40 < G < 115$ an, sodass die GUTSCHEIN-Instanz keine Lösung hat.

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

$$table[i, k] = \begin{cases} \texttt{true}, & \text{falls es } I \subseteq \{1, \ldots, i\} \text{ mit } \sum_{i \in I} w_i = k \text{ gibt} \\ \texttt{false}, & \text{sonst} \end{cases}$$

   Geben Sie einen Algorithmus in Pseudo-Code oder Java an, der die Tabelle *table* mit **dynamischer Programmierung** in Worst-Case-Laufzeit $O(n \times G)$ erzeugt. Begründen Sie die Korrektheit und die Laufzeit Ihres Algorithmus. Welcher Eintrag in *table* löst das GUTSCHEIN-Problem?

Aufgabe 5 (Schnelle Suche von Schlüsseln) [36 PUNKTE]

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.

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.

c) Erläutern Sie schrittweise den Ablauf Ihres Algorithmus für die Folge 1, 5, 11, 8, 4, 2 und den Suchschlüssel 4.

d) Analysieren Sie die Laufzeit Ihres Algorithmus für den Worst-Case, geben Sie diese in O-Notation an und begründen Sie diese.