| Prüfungsteilnehmer | Prüfungstermin | Einzelprüfungsnummer |
|---|---|---|
| Kennzahl: _____ | | |
| Kennwort: _____ | **Herbst 2018** | **46115** |
| Arbeitsplatz-Nr.: _____ | | |

# Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

## — Prüfungsaufgaben —

| | |
|---|---|
| Fach: | **Informatik (Unterrichtsfach)** |
| Einzelprüfung: | **Theoretische Informatik/Algorithmen/Datenstrukturen** |
| Anzahl der gestellten Themen (Aufgaben): | 2 |
| Anzahl der Druckseiten dieser Vorlage: | 13 |

Bitte wenden!

**Thema Nr. 1**
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

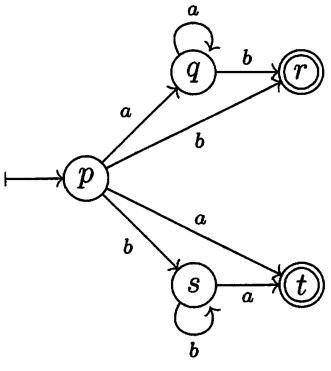
## Aufgabe 1 (Reguläre Sprachen) [24 PUNKTE]

(a) [4 PUNKTE] Betrachten Sie die formale Sprache $L \subseteq \{a, b\}^*$ aller Wörter, die mit $aa$ beginnen oder mit $bb$ enden.

Geben Sie einen regulären Ausdruck für die Sprache $L$ an.

(b) [10 PUNKTE] Entwerfen Sie einen nichtdeterministischen endlichen Automaten, der die Sprache $L$ aus Teilaufgabe (a) akzeptiert.

(c) [10 PUNKTE] Wandeln Sie den folgenden nichtdeterministischen Automaten mit Hilfe der Potenzmengenkonstruktion in einen äquivalenten deterministischen Automaten um. (Berechnen Sie dabei nur erreichbare Zustände.)




## Aufgabe 2 (Kontextfreie Sprachen) [32 PUNKTE]

(a) [4 PUNKTE] Betrachten Sie die formale Sprache $L$ über dem Alphabet $\Sigma = \{0, 1, 2\}$ aller Wörter $wv$ mit $w \in \{0, 1\}^*$ und $v \in \{1, 2\}^*$ so dass die Anzahl der 0en in $w$ genau doppelt so hoch ist wie die der 2en in $v$.

Geben Sie zwei Wörter über $\Sigma$ an, die in $L$ enthalten sind und zwei Wörter über $\Sigma$, die nicht in $L$ enthalten sind. Die Wörter sollen mindestens Länge 5 haben.

(b) [10 PUNKTE] Geben Sie eine kontextfreie Grammatik für die Sprache $L$ an.

Erklären Sie den Zweck der einzelnen Nichtterminale (Variablen) und der Grammatikregeln Ihrer Grammatik.

Fortsetzung nächste Seite!

(c) [2 PUNKTE] Geben Sie für eines Ihrer beiden Wörter aus Teilaufgabe (a), das in $L$ enthalten ist, einen Ableitungs*baum* des Wortes mit Ihrer Grammatik aus Teilaufgabe (b) an.

(d) [10 PUNKTE] Beweisen Sie, dass die Sprache $L$ aus Teilaufgabe (a) nicht regulär ist.

(e) [6 PUNKTE] Ist die folgende Aussage richtig oder falsch? Begründen Sie Ihre Antwort:
„Jede Teilmenge einer kontextfreien Sprache liegt in der Klasse $\mathcal{P}$."

**Aufgabe 3 (Entscheidbarkeit)** [34 PUNKTE]

(a) [6 PUNKTE] Betrachten Sie das folgende Entscheidungsproblem:

**Eingabe:** eine (geeignet codierte) Turingmaschine $M$, ein Eingabewort $w$ und eine natürliche Zahl $n$

**Aufgabe:** entscheiden, ob die Turingmaschine $M$ auf das Eingabewort $w$ nach höchstens $n$ Schritten hält

Ist dieses Problem entscheidbar? Begründen Sie Ihre Antwort.

(b) [20 PUNKTE] Beweisen Sie mit Hilfe eines Reduktionsbeweises, dass das folgende Problem nicht entscheidbar ist:

**Eingabe:** eine (geeignet codierte) Turingmaschine $M$ und sowie ein Eingabewort $w$

**Aufgabe:** entscheiden, ob $M$ auf Eingabewort $w$ gestartet hält und das Wort $w$ *nicht* akzeptiert

(c) [8 PUNKTE] Beweisen Sie, dass das Problem aus (b) semi-entscheidbar (= rekursiv aufzählbar) ist.

Fortsetzung nächste Seite!

**Aufgabe 4 (Hashing)** [71 PUNKTE]

Ein Wörterbuch, das die Operationen SUCHEN, EINFÜGEN und LÖSCHEN zur Verfügung stellt, kann mit einer Hashtabelle implementiert werden. Im Folgenden seien $m, n \in \mathbb{N}$ beliebig, $T[0, \ldots, m-1]$ eine Hashtabelle der Größe $m$ und $S \subset \mathcal{U}$ eine Menge aus dem Universum $\mathcal{U}$ mit $|S| = n$ die in $T$ mittels der Hashfunktion $h : \mathcal{U} \to \{0, \ldots, m-1\}$ gespeichert werden soll.

**Hashing mit Verkettung**

4.1 [2 Punkte] Beschreiben Sie *kurz*, wie eine Hashtabelle aufgebaut ist, die *Verkettung* zur Kollisionsbehandlung verwendet.

4.2 [6 Punkte] Beschreiben Sie *kurz*, wie die Wörterbuchoperationen SUCHEN, EINFÜGEN und LÖSCHEN in einer Hashtabelle wie in Aufgabe 4.1 beschrieben umgesetzt werden.

4.3 [6 Punkte] Zeigen Sie, dass für jede endliche Menge $\mathcal{U}$ mit $|\mathcal{U}| \geq (n-1)m + 1$ und jede Hashfunktion $h : \mathcal{U} \to \{0, \ldots, m-1\}$ eine Menge $S \subseteq \mathcal{U}$ mit $|S| = n$ existiert, so dass alle Elemente von $S$ durch $h$ auf denselben Eintrag der Hashtabelle abgebildet werden.

*Hinweis:* Betrachten Sie die *größte* der Urbildmengen $U_i := \{s \in \mathcal{U} \mid h(s) = i\}$ (für $0 \leq i < m$) von $[0, \ldots, m-1]$ unter $h$.

4.4 [6 Punkte] Wir betrachten eine Hashtabelle, die *Verkettung* zur Kollisionsbehandlung verwendet und nehmen an, dass die Hashfunktion $h$ in *konstanter* Zeit ausgewertet werden kann.

Für $0 \leq i < m$ sei $S_i := \{s \in S \mid h(s) = i\} \subset S$ die Menge der Urbilder von $i$ in $S$.
Begründen Sie, warum die asymptotische worst-case-Laufzeit einer Suchoperation $\Theta(1 + \max_i |S_i|)$ ist!

4.5 [3 Punkte] Was ist die asymptotische worst-case Laufzeit einer Einfüge- bzw. einer Löschoperation in einer Hashtabelle, die *Verkettung* zur Kollisionsbehandlung verwendet? Begründen Sie Ihre Antworten!

**Hashing durch lineares Sondieren**

4.6 [2 Punkte] Um die Position von $x \in S$ in einer Hashtabelle $T$, die *lineares Sondieren* zur Kollisionsbehandlung verwendet, zu bestimmen, wird mittels $h(x)$ sukzessive eine Indexfolge $(i_0, \ldots, i_{m-1})$ berechnet (wobei $0 \leq i_j < m$ für alle $0 \leq j < m$). Geben Sie an, nach welcher Vorschrift die $i_j$ berechnet werden.

Fortsetzung nächste Seite!

4.7 [4 Punkte] Beschreiben Sie *kurz*, was in einem Eintrag $T[i]$ einer Hashtabelle $T$, die *lineares Sondieren* zur Kollisionsbehandlung verwendet, gespeichert sein kann. Geben Sie an, wie die Tabelle unmittelbar nach der Initialisierung aussieht.

*Hinweis*: Es gibt drei verschiedene Arten von Einträgen in $T$!

4.8 [2 Punkte] Geben Sie an, welcher Fehler auftreten kann, wenn ein neues Element in eine Hashtabelle eingefügt werden soll, bei der *lineares Sondieren* zur Kollisionsbehandlung verwendet wird.

4.9 [12 Punkte] Beschreiben Sie *kurz*, wie die Wörterbuchoperationen SUCHEN, EINFÜGEN und LÖSCHEN in einer Hashtabelle, die *lineares Sondieren* zur Kollisionsbehandlung verwendet, umgesetzt werden.

**Hashing am Beispiel**

Folgende Werte sollen in eine Hashtabelle eingefügt werden:

$$S = (5, 14, 17, 18, 16, 3, 20, 6, 7).$$

Wir betrachten dazu die beiden Hashfunktionen

$$h_1(k) = k \mod 11 \qquad \text{sowie} \qquad h_2(k) = (k \cdot (2 \cdot (k+1) + 3) \mod 11.$$

4.10 [6 Punkte] Erstellen Sie die Wertetabellen der beiden Hashfunktionen $h_1$ und $h_2$ für die Schlüssel in $S$.

4.11 [18 Punkte] Fügen Sie die Werte aus $S$ in der angegebenen Reihenfolge in eine anfangs leere Hashtabelle der Größe 11 ein. Verwenden Sie dazu die Hashfunktionen $h_1$ und $h_2$ und verwenden Sie jeweils einmal *Verkettung* und einmal *lineares Sondieren* zur Kollisionsbehandlung (es sind also insgesamt vier Hashtabellen zu erzeugen). Geben Sie für die beiden Hashtabellen mit linearem Sondieren für jeden Schlüssel die Folge der untersuchten Indizes an.

4.12 [4 Punkte] Löschen Sie jetzt den Wert 20 aus den vier Hashtabellen und geben Sie die Hashtabellen erneut an.

Fortsetzung nächste Seite!

**Aufgabe 5 (Korrektheit von Algorithmen)** [32 PUNKTE]

Wir betrachten den nachstehenden Algorithmus:

MINSUFFIXAVERAGE($A[1,\ldots,n]$)

**Eingabe** : Ein Feld $A[1,\ldots,n]$ von $n \geq 1$ Zahlen
**Ausgabe** : $\min_{1\leq i\leq n} \frac{1}{(n-i+1)} \sum_{k=i}^{n} A[k]$

```
1 b[0] ← ∞;
2 a[0] ← 0;
3 für s ← n runter bis 1 tue
4 |   l ← n − s + 1;
5 |   a[l] ← ((n−s)·a[l−1]+A[s]) / (n−s+1);
6 |   b[l] ← min(b[l − 1], a[l]);
7 zurück b[n];
```

5.1 [2 Punkte] Begründen Sie, dass der Wert der Variablen $l$ der Nummer des aktuellen Schleifendurchlaufs entspricht (wenn wir mit dem Zählen der Schleifendurchläufe bei 1 beginnen).

Wir wollen beweisen, dass der Algorithmus korrekt ist, d.h., dass er für die Eingabe $A[1,\ldots,n]$ den Wert

$$\min_{1\leq i\leq n} \frac{1}{(n-i+1)} \sum_{k=i}^{n} A[k]$$

zurückgibt. Für $1 \leq l \leq n$ seien dazu folgende Werte definiert:

$$a_l := \frac{1}{l} \sum_{k=n-(l-1)}^{n} A[k] \qquad \text{und} \qquad b_l := \min_{n-(l-1)\leq i\leq n} \frac{1}{(n-i+1)} \sum_{k=i}^{n} A[k]$$

Wir wollen durch Induktion zeigen, dass für alle $1 \leq l \leq n$ die Aussage

$$\mathcal{A}_l : \text{Nach dem } l\text{-ten Durchlauf der Schleife ist } a[l] = a_l$$

gilt.

5.2 [4 Punkte] Führen Sie den Induktionsanfang $l = 1$ aus.

5.3 [8 Punkte] Führen Sie den Induktionsschritt $l - 1 \to l$ (für $1 < l \leq n$) aus.

Fortsetzung nächste Seite!

Wir wollen als nächstes durch Induktion zeigen, dass für alle $1 \leq l \leq n$ die Aussage

$$\mathcal{B}_l : \text{Nach dem } l\text{-ten Durchlauf der Schleife ist } b[l] = b_l$$

gilt.

5.4 [4 Punkte] Führen Sie den Induktionsanfang $l = 1$ aus.

5.5 [10 Punkte] Führen Sie den Induktionsschritt $l - 1 \to l$ (für $1 < l \leq n$) aus.
*Hinweis:* Verwenden Sie das Resultat aus der vorigen Aufgabe: $a[l] = a_l$ für alle $1 \leq l \leq n$.

5.6 [4 Punkte] Folgern Sie, dass der Algorithmus MinSuffixAverage korrekt ist.

**Thema Nr. 2**
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

## Aufgabe 1 (Reguläre Sprachen) [16 PUNKTE]

(a) [2 PUNKTE] Geben Sie einen regulären Ausdruck für die Sprache $L_1$ an, die alle Wörter über dem Alphabet $\{a, b\}$ enthält, die mindestens zwei $a$'s haben.

(b) [4 PUNKTE] Geben Sie einen möglichst einfachen und eleganten DEA für die Sprache $L((aa)^* + a^*b(a+b)^*)$ an. (Sie müssen dafür kein Konstruktionsverfahren nutzen.)

(c) [2 PUNKTE] Betrachten Sie die folgenden Sprachen über dem Alphabet $\Sigma = \{a, b, c\}$:

$$L_3 = L((aa)^* + ab^*c)$$
$$L_4 = L((cc)^*(c + \varepsilon))$$

Konstruieren Sie einen Homomorphismus $h$ mit $h(L_3) = L_4$. Begründen Sie Ihre Antwort.

*Anmerkung: ein Homomorphismus ist hier eine Funktion $h : \Sigma \to \Sigma$, die buchstabenweise auf die Wörter einer Sprache angewendet wird. Z. B. falls wir den Homomorphismus $h$ definieren als $h(a) := a$, $h(b) := a$ und $h(c) := c$, dann ist $h(abc) = aac$ und $h(\{abc, aa, ab, ac\}) = \{aac, aa, ac\}$.*

(d) [8 PUNKTE] Betrachten Sie die folgenden Sprachen über dem Alphabet $\Sigma = \{a, b\}$:

$L_5 = \{w_1\ abbb\ w_2 \mid w_1, w_2 \in \{a, b\}^*$, die Länge von $w_2$ ist teilbar durch $4\}$

$L_6 = \{a^{n^2} \mid n \in \mathbb{N}, n \geq 1\}$

Geben Sie für die Sprachen jeweils an, ob sie regulär sind oder nicht. Geben Sie einen Beweis für Ihre Antwort. Um Regularität zu zeigen, reicht die Angabe eines Automaten oder eines regulären Ausdrucks, der die Sprache akzeptiert.

## Aufgabe 2 (Kontextfreie Sprachen) [14 PUNKTE]

Für den Beweis von Kontextfreiheit 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-Kontextfreiheit verwenden Sie eine der üblichen Methoden.

(a) [3 Punkte] Seien $L_1$ und $L_2$ nicht-kontextfrei. Ist es dann immer so, dass auch $L_1 \cup L_2$ nicht-kontextfrei ist? Beweisen Sie Ihre Antwort.

Fortsetzung nächste Seite!

(b) [3 Punkte] Sei $L_3$ nicht-kontextfrei und $L_4$ regulär. Kann $L_3 \cup L_4$ kontextfrei sein? Kann $L_3 \cup L_4$ nicht-kontextfrei sein? Beweisen Sie Ihre Antworten.

(c) [3+5 Punkte] Wir bezeichnen mit $\#_\sigma(w)$ die Anzahl der Vorkommen des Zeichens $\sigma$ im Wort $w$. Zum Beispiel ist $\#_a(baaca) = 3$ und $\#_b(baaca) = 1$. Betrachten Sie die folgenden Sprachen über dem Alphabet $\Sigma = \{a, b, c, d\}$:

$$L_5 = \{w \in L(a^*b^*c^*d^*) \mid \#_a(w) = \#_b(w) \text{ und } \#_c(w) = \#_d(w)\}$$
$$L_6 = \{w \in L((a+b+c+d)^*) \mid \#_a(w) = \#_b(w) \text{ und } \#_c(w) = \#_d(w)\}$$

Sind $L_5$ und $L_6$ jeweils kontextfrei oder nicht? Beweisen Sie Ihre Antwort.

## Aufgabe 3 (Entscheidbarkeit und Komplexität) [10 PUNKTE]

Sei $G = (V, E)$ ein ungerichteter Graph. Wir sagen, dass ein Knoten $u$ von einem Knoten $v$ *gesehen* wird, falls $u = v$ oder $(u, v) \in E$.

Wir nennen $A \subseteq V$ eine *alles sehende Menge (ASM)*, falls jeder Knoten in $V$ von mindestens einem Knoten in $A$ gesehen wird.

Wir nennen $A$ *teilbar*, falls $A$ partitionierbar in nicht-leere Teilmengen $A_1$ und $A_2$ ist, so dass jeder Knoten in $V$ *entweder* von mindestens einem Knoten in $A_1$ oder von mindestens einem Knoten in $A_2$ gesehen wird (aber nicht von *sowohl* einem Knoten in $A_1$ als auch von einem Knoten in $A_2$).

Wir betrachten die folgenden Probleme:

| ASM | |
|---|---|
| Gegeben: | Ungerichteter Graph $G = (V, E)$ und $k \in \mathbb{N}$ |
| Gefragt: | Hat $G$ eine ASM $A$ mit $\|A\| \leq k$? |

| Teilbare ASM | |
|---|---|
| Gegeben: | Ungerichteter Graph $G = (V, E)$ und $k \in \mathbb{N}$ |
| Gefragt: | Hat $G$ eine teilbare ASM $A$ mit $\|A\| \leq k$? |

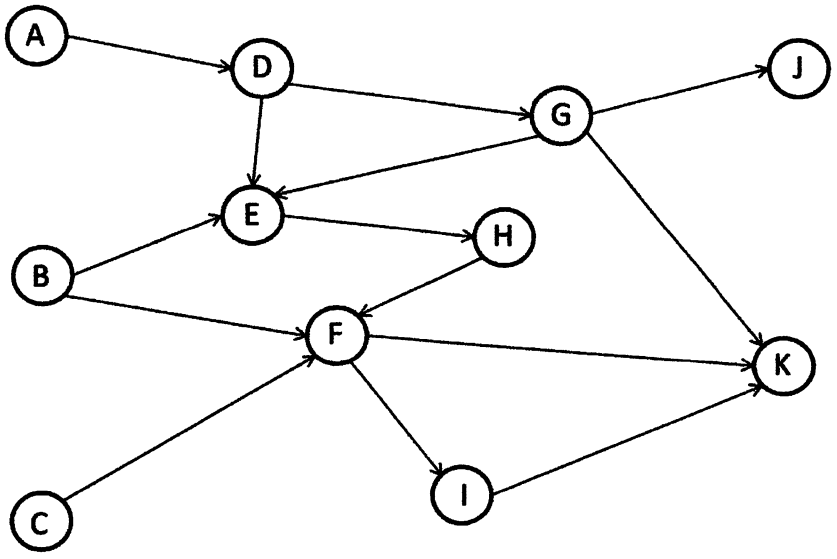
(a) [8 PUNKTE] Beweisen Sie, dass ASM in Polynomialzeit reduzierbar ist auf Teilbare ASM.

(b) [2 PUNKTE] Was können Sie aus a) folgern bezüglich der NP-Vollständigkeit der Probleme?

Fortsetzung nächste Seite!

**Aufgabe 4 (Längste Pfade)** [45 PUNKTE]

Gegeben sei der folgende azyklische, gerichtete Graph.




Der *kritische Pfad* eines solchen Graphen ist definiert als derjenige Pfad von einem beliebigen Knoten mit Eingangsgrad 0 zu einem beliebigen Knoten mit Ausgangsgrad 0 mit der maximalen Pfadlänge.

Analog ist der *kritische Pfad zu einem beliebigen Knoten* der Pfad mit maximaler Pfadlänge von einem Knoten mit Eingangsgrad 0 zu diesem Knoten. Im gegebenen Graphen ist beispielsweise der kritische Pfad zu Knoten E der von A über D und G nach E mit der Pfadlänge 3. Der Pfad von B aus ist kürzer (Pfadlänge 1), weil Knoten E nur direkt von Knoten B aus erreichbar ist. Von Knoten C aus ist Knoten E nicht erreichbar.

(a) [5 PUNKTE] Bestimmen Sie den kritischen Pfad des gegebenen Graphen. Geben Sie die Knotenfolge des kritischen Pfades und seine Pfadlänge an.

(b) [10 PUNKTE] Da der gegebene Graph azyklisch ist, können seine Knoten topologisch sortiert werden. Formal ist die *topologische Sortierung eines Graphen* $G = (V, E)$ eine injektive Abbildung

$$ord : V \to \{1, \ldots, |V|\}$$

sodass gilt:

$$\forall (v, w) \in E : ord(v) < ord(w)$$

Dabei bezeichnet $V$ die Knotenmenge und $E$ die Kantenmenge des Graphen.

Führen Sie eine topologische Sortierung des Graphen durch. Geben Sie eine mögliche topologische Sortierung der Knoten des Graphen an.

Fortsetzung nächste Seite!

(c) [5 PUNKTE] Angenommen, Sie haben im gegebenen Graphen die kritischen Pfade zu allen Vorgängerknoten eines Knotens $v$ bestimmt. Wie berechnet sich dann der kritische Pfad zum Knoten $v$?

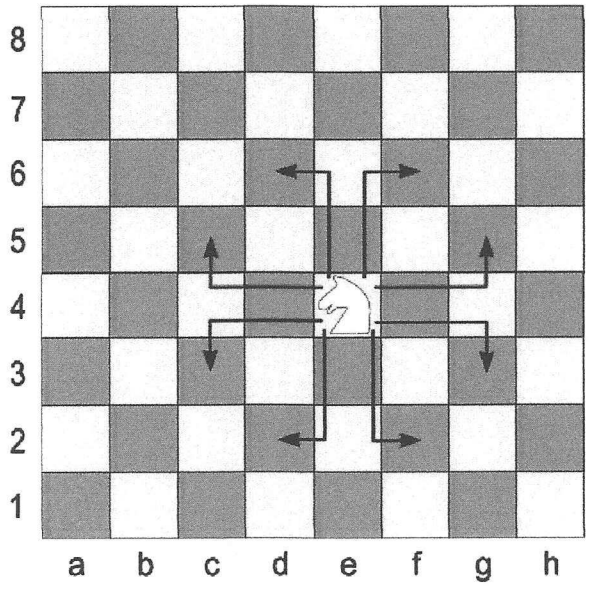
(d) [25 PUNKTE] Formulieren Sie anhand der Kenntnisse, die Sie durch die Aufgabenteile b und c gewonnen haben, einen Algorithmus, der die Länge des kritischen Pfades eines gerichteten, azyklischen Graphen berechnet und geben Sie die asymptotische Laufzeit Ihres Algorithmus in der O-Notation an.

Hinweis: Sie dürfen in Ihrem Algorithmus eine Funktion *topsort*$(G)$ verwenden. Diese liefert eine topologische Sortierung der Knoten des Graphen $G$ als Liste in einer asymptotischen Laufzeit in $O(|V| + |E|)$ zurück.

## Aufgabe 5 (Backtracking) [25 PUNKTE]

Das *Springerproblem* ist ein kombinatorisches Problem, das darin besteht, für einen Springer auf einem leeren Schachbrett eine Route von einem gegebenen Startfeld aus zu finden, auf der dieser jedes Feld des Schachbretts genau einmal besucht.

Ein Schachbrett besteht aus 8×8 Feldern. Ein Springer kann bei einem Zug von einem Ausgangsfeld aus eines von maximal 8 Folgefelder betreten, wie dies in der folgenden Abbildung dargestellt ist. Der Springer darf selbstverstndlich nicht über den Rand des Schachbretts hinausspringen.




Fortsetzung nächste Seite!

Eine Lösung des Springerproblems mit Startfeld `h1` sieht wie folgt aus. Die Felder sind in ihrer Besuchsreihenfolge durchnummeriert. Der Springer bewegt sich also von `h1` nach `f2`, dann von `f2` nach `h3` usw.

```
41 10 29 26 49 12 31 16
28 25 40 11 30 15 50 13
 9 42 27 56 61 48 17 32
24 39 58 47 64 53 14 51
43  8 55 62 57 60 33 18
38 23 46 59 54 63 52  3
 7 44 21 36  5  2 19 34
22 37  6 45 20 35  4  1
```

Formulieren Sie einen *rekursiven* Algorithmus zur Lösung des Springerproblems von einem vorgegebenen Startfeld aus. Es sollen dabei alle möglichen Lösungen des Springerproblems gefunden werden. Die Lösungen sollen durch *Backtracking* gefunden werden. Hierbei werden alle möglichen Teilrouten systematisch durchprobiert, und Teilrouten, die nicht zu einer Lösung des Springerproblems führen können, werden nicht weiterverfolgt. Dies ist durch rekursiven Aufruf einer Lösungsfunktion *huepf*$(x, y, z)$ zu realisieren, wobei

- $x$ und $y$ die Koordinaten des als nächstes anzuspringenden Feldes sind, und
- $z$ die aktuelle Rekursionstiefe enthält. Wenn die Rekursionstiefe 64 erreicht und das betreffende Feld noch unbesucht ist, ist eine Lösung des Springerproblems gefunden.

Der initiale Aufruf Ihres Algorithmus kann beispielsweise über den Aufruf

$$huepf(1, 8, 1)$$

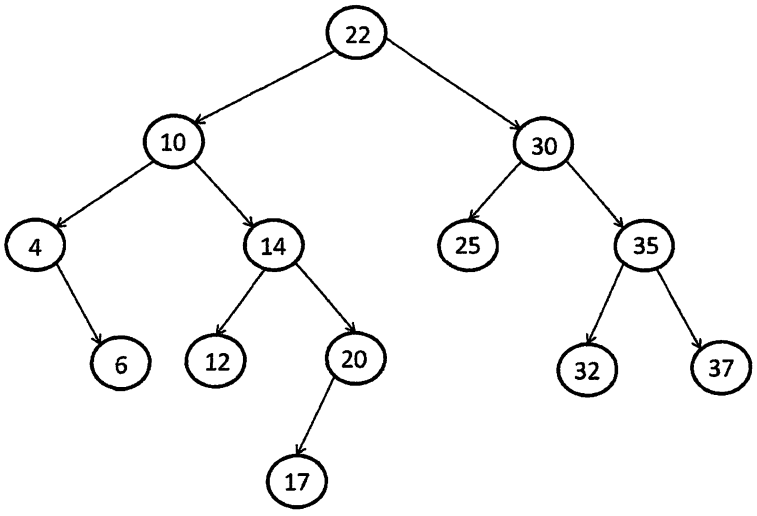
erfolgen.

Wählen Sie geeignete Datenstrukturen zur Verwaltung der unbesuchten Felder und zum Speichern gefundener (Teil)Lösungen. Der Algorithmus soll eine gefundene Lösung in der oben angegebenen Form ausdrucken, also als Matrix mit der Besuchsreihenfolge pro Feld.

Fortsetzung nächste Seite!

**Aufgabe 6 (Balancierte Bäume)** [20 PUNKTE]

Es sei der folgende AVL-Baum mit Schlüsseln aus $\mathbb{N}$ gegeben:




(a) [10 Punkte] Fügen Sie in den gegebenen Baum den Schlüssel **15** ein!

Rebalancieren Sie anschließend den Baum so, dass die AVL-Eigenschaft wieder erreicht wird. Zeichnen Sie den Baum nach jeder Einfach- und Doppelrotation und benennen Sie die Art der Rotation (Links-, Rechts-, Links-Rechts-, oder Rechts-Links-Rotation). Argumentieren Sie jeweils über die Höhenbalancen der Teilbäume. Zeichnen Sie nach jedem Schritt die Höhenbalancen in den Baum ein.

(b) [10 Punkte] Löschen Sie den Schlüssel **25** aus dem gegebenen Baum! Rebalancieren Sie anschließend den Baum so, dass die AVL-Eigenschaft wieder erreicht wird. Zeichnen Sie den Baum nach jeder Einfach- und Doppelrotation und benennen Sie die Art der Rotation (Links-, Rechts-, Links-Rechts-, oder Rechts-Links-Rotation). Argumentieren Sie jeweils über die Höhenbalancen der Teilbäume.