Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	Frühjahr 2019	46115
Kennwort: _____		
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: 8

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Aufgabe 1

Antworten Sie mit „Stimmt“ oder „Stimmt nicht“. Begründen Sie Ihr Urteil.

(a) Sei $\Sigma = \{a, b\}$. Wenn $L_1 \leq \Sigma^*$ regulär ist, dann ist auch das Komplement $\Sigma^* \setminus L_1$ regulär.

(b) Für reguläre Ausdrücke α, β gilt $L((\alpha + \beta)^*) = L(\alpha^* + \beta^*)$.

(c) Kontextfreie Sprachen sind unter der Kleene-Stern-Operation abgeschlossen, d. h. für kontextfreies $L \subseteq \Sigma^*$ ist auch L^* kontextfrei.

(d) Wenn es einen deterministischen Algorithmus gibt, der in polynomieller Zeit entscheidet, ob eine vorgelegte aussagenlogische Formel eine erfüllende Belegung hat, dann ist $\mathcal{P} = \mathcal{NP}$.

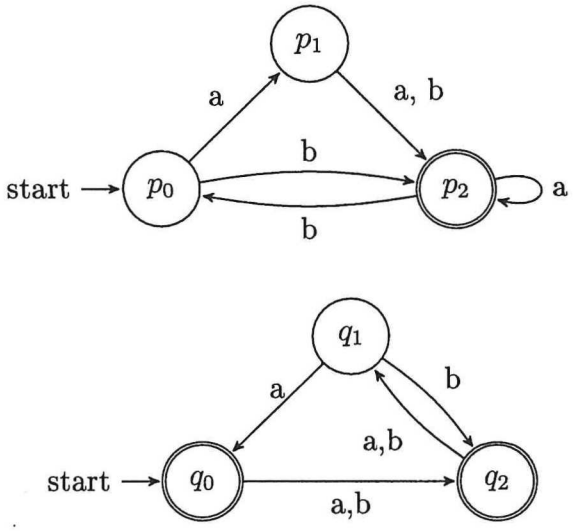
(e) $\mathcal{NP}$-vollständige Probleme sind unentscheidbar.

(f) Die Menge der (Kodierungen von) Turingmaschinen, die bei leerer Eingabe nach maximal 1000 Schritten halten, ist entscheidbar.

Fortsetzung nächste Seite!

Aufgabe 2

(a) Konstruieren Sie einen endlichen Automaten, dessen Sprache die Schnittmenge der Sprachen der folgenden zwei deterministischen endlichen Automaten ist.



(b) Sei $L = \{a^i b^j c^{2i} \mid i, j \in \mathbb{N}\}$. Stellen Sie sich vor, dass ein Schüler/eine Schülerin S behauptet, einen deterministischen endlichen Automaten A konstruiert zu haben, der genau die Sprache L akzeptiert. Sie argumentieren mit dem Pumping-Lemma, dass L nicht regulär ist und deshalb die Sprache von A nicht L sein kann. S hält Ihre Argumentation für zu abstrakt und ist immer noch von der Korrektheit von A überzeugt.

Sie können S überzeugen, wenn Sie ein Wort w angeben, das Element von L ist, aber nicht von A akzeptiert wird oder von A akzeptiert wird und nicht Element von L ist.

Beschreiben Sie ein Verfahren, wie Sie in Abhängigkeit von A ein solches Wort w finden.

Aufgabe 3

Gegeben sei die kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit Sprache $L(G)$, wobei $V = \{S\}$ und $\Sigma = \{a, b\}$. P bestehe aus den folgenden Produktionen:

$$S \to SS \mid aSb \mid \varepsilon$$

(a) Geben Sie alle neun Wörter $w \in L(G)$ mit Länge $|w| \leq 6$ an.

(b) Bringen Sie G in Chomsky-Normalform und erklären Sie Ihre Vorgehensweise.

Fortsetzung nächste Seite!

Aufgabe 4

Gegeben sei die Turingmaschine $M = (Q, q_0, F, \Sigma, \Gamma, \square, \delta)$ mit Zustandsmenge $Q = \{q_0, q_1, q_2, q_f\}$, wobei q_0 der Startzustand ist. Die Turingmaschine hält, wenn sie den Zustand q_f annimmt ($F = \{q_f\}$). Das Eingabealphabet ist $\Sigma = \{a, b\}$, das Bandalphabet ist $\Gamma = \Sigma \cup \{\square\}$ mit Blank-Zeichen $\square$ für leeres Feld. Die Übergangsfunktion $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$, 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 $\delta(q_0, a) = (q_1, b, L)$):

	a	b	$\square$
q_0	(q_1, b, L)	(q_0, b, R)	$(q_1, \square, L)$
q_1	(q_1, a, L)	(q_1, b, L)	$(q_2, \square, R)$
q_2	(q_2, a, L)	(q_2, b, R)	$(q_f, \square, N)$

(a) Beschreiben Sie möglichst exakt, welche Schritte (Zustände, Schreibvorgänge, Bewegung des SL-Kopfes) die Turingmaschine bei der Abarbeitung des Wortes *bab* ausführt.

(b) Beschreiben Sie möglichst exakt, welche Schritte (Zustände, Schreibvorgänge, Bewegung des SL-Kopfes) die Turingmaschine bei der Abarbeitung des Wortes *baba* ausführt.

(c) Charakterisieren Sie, bei welchen Eingaben die Turingmaschine hält und bei welchen nicht. Begründen Sie, ob Sie damit das Halteproblem gelöst haben.

Fortsetzung nächste Seite!

Aufgabe 5 (Algorithmen und Datenstrukturen)

Gegeben sind eine endliche Menge von Wörtern $T = \{s_0, ..., s_{m-1}\}$ und ein Zielwort w der Länge n. Gefragt ist, ob das Zielwort w als Konkatenation von Wörtern aus T zusammengesetzt werden kann, wobei die Wörter in T beliebig oft verwendet werden dürfen.

Beispiel:

$$w = \text{apfelmuffinsundschokomuffins}$$

$$T = \{\text{f, lmu, e, ap, fin, ins, sund, kom, u, n,}$$

$$\text{su, dscho, unds, nd, uff, hok, felm}\}$$

$$w = \text{ap felm uff ins u n dscho kom uff ins}$$

Mit $w(i, j)$ bezeichnen wir das Segment vom i-ten (Start bei 0) bis zum (einschl) j-ten Buchstaben von w. Im Beispiel: $w(3, 10) = \text{elmuffin}$

Der folgende Algorithmus löst das Problem in exponentieller Zeit.

```
write(i,j): /* kann w(i,j) dargestellt werden? */
  if (i > j) return true;
  for (k=0..m-1)
       if (w(i,j) == s_k) return true;
  for (l=i..j-1)
       if (write(i,l) && write(l+1,j)) return true;
  return false;

main:
write (0,n-1);
```

(a) Entwerfen Sie mithilfe von dynamischer Programmierung einen Algorithmus, der das Problem in polynomieller Zeit löst.

(b) Geben Sie die Laufzeit Ihres Algorithmus in O-Notation an, wobei Sie zur Vereinfachung annehmen dürfen, dass m, die Zahl der Muster, konstant ist.

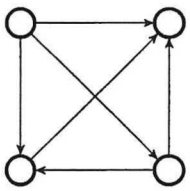
(c) Nennen Sie zwei weitere Beispiele für die Anwendungen dynamischer Programmierung.

Thema Nr. 2
(Aufgabengruppe)

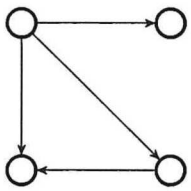
Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Aufgabe 1 (Turniergraph)

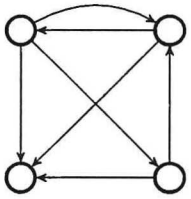
Ein *Turniergraph* ist ein gerichteter Graph auf n Knoten, in dem zwischen je zwei verschiedenen Knoten u, v genau eine Kante existiert - entweder (u, v) oder (v, u), aber niemals beide oder keine, siehe Abbildung 1.



(a) Turniergraph



(b) Kein Turniergraph



(c) Kein Turniergraph

Abbildung 1: Beispiele zur Veranschaulichung der Definition des Turniergraphs

(a) Zeigen Sie, dass in jedem Turniergraph **mindestens** ein Knoten existiert, von dem mindestens $(n-1)/2$ Kanten ausgehen.

(b) Eine Knotenmenge S *dominiert*, falls für jeden Knoten $x \notin S$ ein Knoten $s \in S$ existiert, so dass (s, x) eine Kante des Graphen ist.

Geben Sie einen Algorithmus an, der in jedem Turniergraphen eine dominierende Knotenmenge der Größe $\mathcal{O}(\log n)$ findet. Argumentieren Sie, dass Ihr Algorithmus korrekt ist. Sie dürfen hierzu die Aussage von Teilaufgabe (a) nutzen.

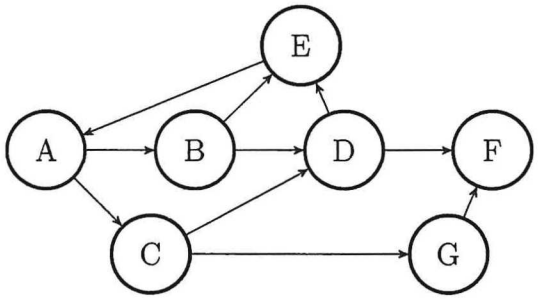
(c) Angenommen, es ist bekannt, dass die kleinste dominierende Knotenmenge $\mathcal{O}(\log n)$ Knoten enthält. Geben Sie einen Algorithmus an, der diese Knotenmenge in n^s Schritten berechnet, wobei $s \in \mathcal{O}(\log n)$ gilt.

Aufgabe 2 (Tiefensuche)

(a) Führen Sie auf dem angegebenen Graphen eine Tiefensuche, ausgehend vom Knoten A, aus. Dokumentieren Sie tabellarisch, in welcher Reihenfolge die Kanten betrachtet werden und ob die jeweils betrachtete Kante dem Tiefensuchbaum hinzugefügt wird. Falls mehrere Knoten zur Auswahl stehen, wählen Sie stets den Knoten mit der lexikographisch niedrigeren Bezeichnung.

Zeichnen Sie den resultierenden Baum.

Fortsetzung nächste Seite!



(b) Gegeben sei nun ein gerichteter Graph $G = (V, E)$, dessen Knoten mit natürlichen Zahlen beschriftet sind. Für jeden Knoten v soll der Knoten mit geringster Beschriftung bestimmt werden, der von v aus erreichbar ist. Geben Sie einen Algorithmus mit Laufzeit $\mathcal{O}(|V|(|V| + |E|))$ an, der dieses Problem löst.

(c) Für das Problem aus Teil (b) existiert auch ein Algorithmus mit Laufzeit $\mathcal{O}(|V| + |E|)$. Geben Sie den verbesserten Algorithmus an. Argumentieren Sie, dass Ihr Algorithmus die geforderte Laufzeit erreicht.

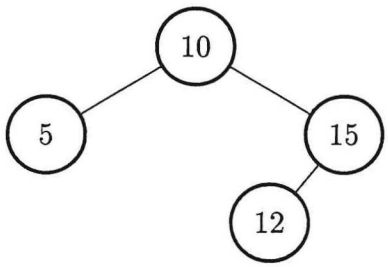
Aufgabe 3 (AVL-Bäume)

(a) Zeigen oder widerlegen Sie die folgende Aussage: Wird ein Element in einen AVL-Baum eingefügt und unmittelbar danach wieder gelöscht, so befindet sich der AVL-Baum wieder in seinem Ursprungszustand.

(b) Fügen Sie in den gegebenen Baum den Schlüssel **11** 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.

Tipp: Zeichnen Sie nach jedem Schritt die Höhenbalancen in den Baum ein.



Fortsetzung nächste Seite!

Aufgabe 4 (Formale Sprachen und Komplexität)

Das Shuffle-Produkt $L_1 \star L_2$ zweier Sprachen L_1, L_2 ist bekanntlich definiert durch

$$L_1 \star L_2 = \{u_1 v_1 u_2 v_2 \dots u_n v_n \mid u_1 u_2 \dots u_n \in L_1 \text{ und } v_1 v_2 \dots v_n \in L_2\}$$

wobei $u_1, \dots, u_n$, $v_1, \dots, v_n$ Wörter sind (möglicherweise leer). Die Sprache $L_1 \star L_2$ enthält also alle Wörter, die man durch Verzahnen eines Wortes in L_1 mit einem Wort in L_2 erhalten kann.

Beispiel:

$$\{aab, abab\} \star \{aa\} = \{aaaab, aaaba, aabaa, aaabab, aabaab, aababa, abaaab, abaaba, ababaa\}$$

Shuffle-Produkte spielen bei der Verifikation nebenläufiger Programme eine wichtige Rolle.

(a) Es sei $L = \{a^m b^n | m, n \geq 0\}$, also die Sprache des regulären Ausdrucks $a^* b^*$. Zeigen Sie, dass gilt $L \star L = \{a, b\}^*$.

(b) Das Shuffle-Produkt regulärer Sprachen ist wiederum regulär. Begründen Sie dies durch eine geeignete Automatenkonstruktion. Mit anderen Worten: Beschreiben Sie, wie man aus zwei endlichen Automaten $A_1 = (\Sigma, Q, \delta, q_I, F)$ und $A_2 = (\Sigma, Q', \delta', q'_I, F')$ über demselben Alphabet einen Automaten für $L(A_1) \star L(A_2)$ konstruiert.

(c) Begründen Sie, ob die folgende Aussage wahr ist:
"Jedes NP-vollständige Problem ist entscheidbar."