Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr
2018

46115

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: Reguläre Ausdrücke

Gegeben sei die Sprache L. L besteht aus der Menge aller Worte über dem Alphabet $\Sigma = \{0, 1, 2\}$, die mit 0 beginnen und bei denen das vorletzte und das letzte Zeichen übereinstimmen.

a) Geben Sie alle Worte bis zur Länge vier von L an.

b) Zeigen Sie durch Angabe eines regulären Ausdrucks, dass die Sprache L regulär ist.

c) Konstruieren Sie (graphisch) einen nichtdeterministischen endlichen Automaten ohne Spontanübergänge (= ε-Kanten), der L akzeptiert.

Aufgabe 2: Komplexitätsklassen

a) Definieren Sie die Komplexitätsklassen P und NP.

b) Warum gilt P $\subseteq NP$?

Aufgabe 3: Turingmaschinen

Gegeben sei die Sprache L = $\{a^n b^{2n} | n \geq 1\}$.

a) Geben Sie eine Turingmaschine M an, die L erkennt. Die Eingabeworte sind durch die Sonderzeichen $\in$ vorne und \$ hinten begrenzt.

b) Beschreiben Sie in Worten, wie Ihre Turingmaschine arbeitet.

c) Konstruieren Sie M durch Angabe der Befehle und kommentieren Sie, was die Befehle in Bezug auf die Arbeitsweise von M unter a) leisten.

Fortsetzung nächste Seite!

Aufgabe 4: Pumping-Lemma

Das Pumping-Lemma für reguläre Sprachen ist bekanntlich folgendermaßen definiert:

Satz (Pumping-Lemma)
- $L \in \text{REG} \Rightarrow (\exists k \in \mathbb{N}_0 \forall w \in L : |w| \geq k \Rightarrow \exists$ Zerlegung $w = xyz$ mit
  1. $|y| \geq 1$
  2. $|xy| \leq k$
  3. $\forall i \in \mathbb{N}_0 : xy^i z \in L)$

Zeigen Sie damit, dass die Sprache L = $\{a^n b^{n+3} | n \geq 2\}$ nicht regulär ist.

Aufgabe 5: Mastertheorem

Der Hauptsatz der Laufzeitfunktionen ist bekanntlich folgendermaßen definiert:

Satz (Mastertheorem)
- Sei $T(n) = \begin{cases} d \in \Theta(1), & \text{falls } n \leq k \\ a \cdot T(\frac{n}{b}) + g(n), & \text{sonst} \end{cases}$ mit $k \in \mathbb{N}$, $a \geq 1$ und $b > 1$
- Dann gilt
  1. $g(n) \in \mathcal{O}(n^{\log_b a - \epsilon})$ für ein $\epsilon > 0 \Rightarrow T(n) \in \Theta(n^{\log_b a})$
  2. $g(n) \in \Theta(n^{\log_b a}) \Rightarrow T(n) \in \Theta(n^{\log_b a} \cdot \log n) = \Theta(g(n) \cdot \log n)$
  3. $g(n) \in \Omega(n^{\log_b a + \epsilon})$ für ein $\epsilon > 0$ und
     $a \cdot g(\frac{n}{b}) \leq c \cdot g(n)$ für fast alle n und ein c mit $0 < c < 1$
     $\Rightarrow T(n) \in \Theta(g(n))$

Sei nun folgende Funktion für die Anzahl an Schritten eines Algorithmus auf der Eingabegröße n gegeben:

T(n) = 4 T(n/2) + 2n log5(n) + 7n.

Ordnen Sie T(n) einem der drei Fälle des Mastertheorems zu. Beweisen Sie, dass T(n) die Voraussetzungen des Falles erfüllt und geben Sie die resultierende Laufzeitkomplexität von T(n) an.

Fortsetzung nächste Seite!

Aufgabe 6: Sortieren

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.

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

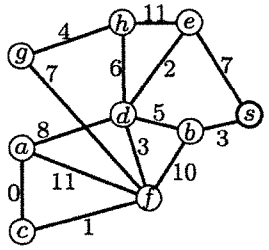
(i) Welche Worst-Case-Laufzeit (O-Notation) hat Mergesort für n Elemente?

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

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

Aufgabe 7: Kürzeste Wege

a) Berechnen Sie mithilfe des Algorithmus von Dijkstra die kürzesten Wege vom Knoten s zu allen anderen Knoten im folgenden aufgezeichneten 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, δ) mit v als Knotenname, p als aktueller Vorgängerknoten und δ 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, δ) auf, wobei diese sog. "grauen Randknoten" in folgenden Durchgängen erneut betrachtet werden müssen. Zeichnen Sie anschließend den entstandenen Wegebaum, d.h. den Graphen G, in dem nur noch diejenigen Kanten vorkommen, die Teil der kürzesten Wege von s zu allen anderen Knoten sind.

Fortsetzung nächste Seite!

b) Nehmen Sie jetzt an, dass die Kantengewichte auch negativ sein können. Geben Sie ein kleines Beispiel (max. 3 Knoten) an und machen Sie daran deutlich, dass der Dijkstra-Algorithmus dann falsche Ergebnisse liefern kann.

Aufgabe 8: Spannbaum

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

	s	a	b	c	d	e	f	g	h
s	-	-	3	-	-	7	-	-	-
a	-	-	-	0	8	-	11	-	-
b	3	-	-	-	5	-	10	-	-
c	-	0	-	-	-	-	1	-	-
d	-	8	5	-	-	2	3	-	6
e	7	-	-	-	2	-	-	-	11
f	-	11	10	1	3	-	-	7	-
g	-	-	-	-	-	-	7	-	4
h	-	-	-	-	6	11	-	4	-

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 aktuellen 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.

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.

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

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Aufgabe 1:

Gegeben sei der nichtdeterministische endliche Automat $N = (Q, \{0,1\}, \delta, q_0, F)$ mit $Q = \{A, B, C\}, q_0 = A, F = \{C\}$ und

δ	0	1
A	$\{B, C\}$	$\emptyset$
B	$\{A\}$	$\emptyset$
C	$\emptyset$	$\{A, B\}$

a) Wenden Sie die Potenzmengenkonstruktion an, um einen deterministischen endlichen Automaten für die Sprache $L(N)$ zu erhalten.

b) Beweisen oder widerlegen Sie: für reguläre Ausdrücke α, β gilt:

$$L((\alpha|\beta)^*) = L(\alpha^*|\beta^*)$$

c) Für eine gegebene Sprache $L \subseteq \{a, b, c\}^*$ sei $\mathsf{replace}(L)$ die Sprache bestehend aus genau den Wörtern, die entstehen, wenn in jedem Wort aus L jedes Vorkommen von c durch ab ersetzt wird.

Beispiel: $\mathsf{replace}(\{a, c, ba, ab, abc, cab\}) = \{a, ab, ba, abab\}$

i) Zeigen Sie: Wenn L regulär ist, ist auch $\mathsf{replace}(L)$ regulär.

ii) Widerlegen Sie: Wenn $\mathsf{replace}(L)$ regulär ist, ist auch L regulär.

Aufgabe 2:

a) Zeigen Sie: Die Sprache $\{(ba)^n b(ab)^n \mid n \in \mathbb{N}\}$ ist regulär.

b) Zeigen Sie, dass $\{a^i c^j b^{2i} c^k \mid i, j, k \in \mathbb{N}\}$ kontextfrei ist, indem Sie eine geeignete kontextfreie Grammatik angeben.

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

$$S \to X \mid SbY \qquad X \to dSe \mid a \qquad Y \to X \mid YcX$$

Bringen Sie G in Chomsky-Normalform.

Fortsetzung nächste Seite!

Aufgabe 3:

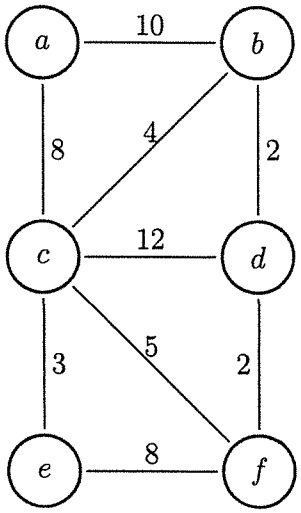
a) Ein AVL-Baum ist stets ein binärer Suchbaum. Geben Sie informell die zentrale Eigenschaft von AVL-Bäumen an, die an jedem Knoten erfüllt ist.

b) Fügen Sie nacheinander die folgenden Zahlen in einen anfangs leeren AVL-Baum ein: 5, 7, 9, 12, 15.
Zeichen Sie den Baum nach jedem Einfügen und nach jeder Rotation.

c) In welcher Reihenfolge können die Zahlen 1, 2, 3, 4, 5, 6, 7, 8, 9 in einen anfangs leeren AVL-Baum eingefügt werden, sodass keine Rotationen stattfinden? Geben Sie eine mögliche Reihenfolge an und zeichnen Sie den daraus resultierenden AVL-Baum.

Aufgabe 4:

Sei G der folgende Graph.



a) Bestimmen Sie mithilfe des Dijkstra-Algorithmus den Kürzeste-Wege-Baum im ungerichteten Graph G, ausgehend vom Knoten a. Geben Sie die Einzelschritte Ihrer Berechnung, inklusive der aktuellen Warteschlange, tabellarisch an. Zeichnen Sie den resultierenden Baum.

Fortsetzung nächste Seite!

Ihre Tabelle sollte wie folgt beginnen:

a	b	c	d	e	f	Warteschlange
0	∞	∞	∞	∞	∞	a

b) 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.

c) 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
0	∞	∞	∞	∞	∞	a

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

d) Erklären Sie, warum der Kürzeste-Wege-Baum und der minimale Spannbaum nicht notwendigerweise identisch sind.

Aufgabe 5:

Eine S-Bahnstrecke mit n Stationen wird von einer Linie von Anfang bis Ende befahren. Wegen der vielen Haltepunkte ist die Fahrzeit sehr lang. Daher soll eine Expresslinie eingerichtet werden. Diese hält nicht an allen Stationen und kann dadurch die Fahrzeit erheblich verkürzen.

Einige Haltestellen werden als Expresshaltestellen ausgewählt und die Expresslinie hält nur an diesen Punkten. Durch eine geeignete Kombination an normalen Zügen und Expresszügen kann ein Nutzer seine Fahrzeit minimieren (die Fahrzeit ist hier die Gesamtzahl der Stopps des Nutzers – insbesondere vernachlässigen wir hierbei die Zeit für das eventuelle Wechseln eines Zuges).

Ziel ist es, dass jeder Nutzer eine Fahrzeit von höchstens $\mathcal{O}(\sqrt{n})$ hat. Finden Sie eine geeignete Verteilung der Expresshaltestellen. Beschreiben Sie, wie ein Nutzer effizient von Haltestelle i zu Haltestelle j fährt. Begründen Sie die daraus resultierende (asymptotische) Worst-Case-Fahrzeit.