Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr 2017

46115

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Th. Informatik, Algorith./Datenstr.**

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!

Die Fragen in dieser Klausur sollen in erster Linie das grundlegende Verständnis der Konzepte und Zusammenhänge prüfen. Man kann sie oft in sehr unterschiedlicher Tiefe und Detaillierungsgrad beantworten. Die Anzahl der Punkte für die Teilfragen gibt einen ersten Hinweis darauf, welcher Detailgrad der Antwort erwartet wird. Um einen weiteren Hinweis zu geben, wird bei jeder Frage die Länge der Antwort in der Musterlösung, in der Anzahl getippter Zeilen, angegeben.

87 Punkte

Aufgabe 1: (Verständnis Automaten) (17 Pkt.)

a) Beschreiben Sie den strukturellen Unterschied zwischen einem nicht-deterministischen (NFA) und einem deterministischen (DFA) endlichen Automaten. (2 Zeilen) (1 Pkt.)

b) Kann ein NFA ebenfalls zum Lösen des Wortproblems benutzt werden? Wenn ja, was muss dann beim Erkennen eines Wortes beachtet werden? Wenn nein, wieso nicht? (1 Zeile)(2 Pkt.)

c) Wenn man einen NFA mit n Knoten zu einem DFA macht, wieviel Knoten kann der DFA dann im schlimmsten Fall haben? (1 Zeile)(1 Pkt.)

d) Sei A ein DFA, welcher die Sprache L_A akzeptiert. Sei B derjenige DFA, der aus A entsteht, wenn Endzustände mit Nicht-Endzuständen „vertauscht" werden. Drücken Sie die von B akzeptierte Sprache L_B in Abhängigkeit von L_A aus. (1 Zeile)(1 Pkt.)

e) Welche Eigenschaft muss ein DFA haben, wenn er eine Sprache mit unendlich vielen Wörtern akzeptiert? (1 Zeile)(1 Pkt.)

f) NFAs und DFAs akzeptieren die gleiche Menge von Sprachen. Gilt das auch für nicht-deterministische und deterministische Kellerautomaten? Falls ja, wieso, falls nein, geben Sie ein Gegenbeispiel an. (1 Zeile)(2 Pkt.)

g) Auf welche Weise benutzt man Kellerautomaten zum Erkennen von Worten einer Typ-2-Sprache (nach „Chomsky"), deren Grammatik gegeben ist? Erklären Sie die Top-down-Methode.

(5 Zeilen)(4 Pkt.)

h) Auf welche Weise benutzt man Turingmaschinen zum Erkennen von Wörtern einer Typ-1- oder Typ-0-Sprache (nach „Chomsky"), deren Grammatik gegeben ist? Erklären Sie die Bottom-up-Methode. (3 Zeilen)(3 Pkt.)

i) Wie äußert sich bei der Bottom-up-Methode in der Turingmaschine der Unterschied zwischen Typ-1- und Typ-0-Sprachen? (3 Zeilen)(2 Pkt.)

Fortsetzung nächste Seite!

Aufgabe 2: (Verständnis Formale Sprachen) (14 Pkt.)

a) Was ist das Wortproblem einer formalen Sprache? (1 Zeile)(1 Pkt.)

b) Was ist der Unterschied zwischen dem Typ einer Sprache und dem Typ einer Grammatik? (1 Zeile)(1 Pkt.)

c) Wieso ist das Wortproblem für Sprachen vom Typ 1, 2 und 3 (nach „Chomsky") entscheidbar? Skizzieren Sie ein einfaches Entscheidungsverfahren und begründen Sie, weshalb es immer terminiert. Warum funktioniert es nicht bei Typ-0-Sprachen (nach „Chomsky") ? (6 Zeilen) (4 Pkt.)

d) Wenn das Wortproblem einer Sprache L entscheidbar ist, ist dann auch das Wortproblem des Komplements von L entscheidbar? Begründen Sie Ihre Antwort. (1 Zeile)(1 Pkt.)

e) Wie kann man eine Abschätzung für die Pumping-Zahl bei Typ-3-Sprachen (nach „Chomsky") bekommen? (1 Zeile)(1 Pkt.)

f) Wie kann man eine Abschätzung für die Pumping-Zahl bei Typ-2-Sprachen (nach „Chomsky") bekommen? (1 Zeile)(1 Pkt.)

g) Wieso ist die Komplexität des Wortproblems bei Typ-2-Sprachen (nach „Chomsky") höchstens $O(n^3)$? (3 Zeilen)(2 Pkt.)

h) Geben Sie ein Beispiel für eine Sprache, die nicht vom Typ 0 (und auch nicht vom Typ 1,2,3) nach „Chomsky" ist, und skizzieren Sie einen Beweis. (3 Zeilen)(3 Pkt.)

Aufgabe 3: (Verständnis Berechenbarkeitstheorie) (22 Pkt.)

a) Was müsste man tun, um die Church-Turing-These zu widerlegen? (2 Zeilen)(2 Pkt.)

b) Die Berechenbarkeitstheorie betrachtet nur Funktionen, die natürliche Zahlen auf natürliche Zahlen abbildet. Geben Sie zwei Methoden an, um Probleme, die für Zeichenketten definiert sind, auf Probleme von natürlichen Zahlen abzubilden. (6 Zeilen)(4 Pkt.)

c) Haben Turingmaschinen mit parallelen Bändern mehr Berechnungskraft als solche mit nur einem Band? Begründen Sie Ihre Antwort kurz. (2 Zeilen)(2 Pkt.)

d) Haben Turingmaschinen, die mit beliebigen Symbolen arbeiten mehr Berechnungskraft als solche, die nur mit 0 und 1 arbeiten? Begründen Sie Ihre Antwort kurz. (1 Zeile)(2 Pkt.)

e) Warum ist die Loop-Programmiersprache schwächer als die While-Programmiersprache? (3 Zeilen)(2 Pkt.)

f) Kann man aus einem Semientscheidungsverfahren für eine Menge M und einem Semientscheidungsverfahren für das Komplement von M ein Entscheidungsverfahren für M machen? Wenn ja, wie, wenn nein, warum nicht? (2 Zeilen)(2 Pkt.)

Fortsetzung nächste Seite!

g) Unter welcher Bedingung kann man aus einem Semientscheidungsverfahren für eine Menge M ein Aufzählungsverfahren für M machen? Skizzieren Sie das Verfahren. (4 Zeilen)(3 Pkt.)

h) Ist die Menge der terminierenden Turingmaschinen mit leerer Eingabe rekursiv aufzählbar? Wenn ja, wie, wenn nein, warum nicht? (3 Zeilen)(3 Pkt.)

i) Braucht man für die Bearbeitung von n*n-Matrizen immer geschachtelte Schleifen? Begründen Sie Ihre Antwort. (1 Zeile)(2 Pkt.)

Aufgabe 4: (Verständnis Komplexitätstheorie) (13 Pkt.)

a) Warum betrachtet man O(n) und z.B. O(2n) als gleichwertig? (2 Zeilen)(1 Pkt.)

b) Die Komplexität von binärer Suche im sortierten Array ist O(log(n)). Finden Sie ein intuitives Argument, welches auch Schülerinnen und Schüler verstehen können, warum es kein schnelleres Verfahren gibt. (3 Zeilen)(2 Pkt.)

c) Wie ist das Verhältnis von Platz-Komplexität und Zeit-Komplexität? Begründen Sie Ihre Antwort. (2 Zeilen)(2 Pkt.)

d) Ist das SAT-Problem Loop-berechenbar? Wenn ja, warum, wenn nein, warum nicht? (4 Zeilen)(3 Pkt.)

e) Das Konvexe-Hüllen Problem in 2 Dimensionen, ist das Problem, für eine 2D-Punktmenge das kleinste umfassende konvexe Polygon (Folge von 2D-Punkten) zu finden. Es gibt einen Algorithmus, der dieses Problem in O(n log(n)) Zeit löst. Wie müsste man vorgehen, um zu beweisen, dass es keinen schnelleren Algorithmus geben kann? (3 Zeilen)(3 Pkt.)

f) Könnte man einen polynomiellen Algorithmus zur Lösung des Graph-Färbeproblems auch zur Lösung des Travelling-Salesman Problems benutzen? Begründen Sie Ihre Antwort. (2 Zeilen)(2 Pkt.)

Fortsetzung nächste Seite!

Aufgabe 5: (Verständnis Algorithmen und Datenstrukturen) (21 Pkt.)

a) Betrachte eine Hashtabelle der Größe m = 10.

Welche der folgenden Hashfunktionen für Hashing mit verketteten Listen ist dafür besser geeignet? Begründen Sie Ihre Wahl.

1) $h_1(x) = (4x+4) \bmod m$

2) $h_2(x) = (5x+5) \bmod m$. (2 Zeilen 3Pkt.)

b) Betrachte wieder eine Hashtabelle der Größe m = 10.

Welche der folgenden Hashfunktionen für Hashing mit *offener Adressierung* ist dafür am besten geeignet? Begründen Sie Ihre Wahl.

1) $h_1(x,i) = (3x+(i+1)\ m) \bmod m$

2) $h_2(x,i) = (3x+(i+1)(m-1)) \bmod m$. (2 Zeilen 3Pkt.)

c) Angenommen Sie haben für die Implementierung einer beliebigen, aber sortierten Menge von Ganzzahlen die Wahl zwischen einem Array, welches die Zahlen in sortierter Reihenfolge enthält, und verketteten Listen. Falls ‚insert' die bei weitem häufigste Operation ist, welche Version würden Sie wählen? Begründen Sie Ihre Wahl. (4 Zeilen 4Pkt.)

d) Warum könnte man auch mit AVL-Bäumen Priority-Queues implementieren?

(1 Zeilen 2Pkt.)

e) Der A*-Algorithmus ist ein Verfahren um in einem Graphen mit gewichteten Kanten den kürzesten Weg zwischen zwei Knoten zu finden. Er benutzt dazu eine Gewichtungsfunktion f(k) = g(k) + h(k), wobei g(k) die tatsächliche Entfernung zwischen Startknoten und Knoten k angibt, und h(k) die geschätzte Entfernung von Knoten k zum Ziel. A* führt in einer OpenList die aktuelle Menge der zu untersuchenden Knoten. Am Anfang enthält sie den Startknoten. Danach wählt er einen Knoten aus der OpenList mit dem kleinsten f-Wert und ersetzt ihn durch seine Nachfolgeknoten.

Frage: Soll der A*-Algorithmus terminieren, sobald er zum ersten mal den Zielknoten erreicht hat? Wenn ja, warum, wenn nein, geben Sie ein Gegenbeispiel. (5 Zeilen(3 Pkt.)

f) Könnte man eine Implementierung des A*-Algorithmus auch so steuern, dass er sich wie der Dijkstra-Algorithmus zum Finden kürzester Pfade verhält? Wenn ja, wie müsste man es tun, wenn nein, warum nicht? (1Zeile)(2 Pkt.)

g) Angenommen, Sie wollen mit Ihrer Klasse 5 Barockkirchen in Bayern besuchen, und versuchen, den kürzesten Weg durch alle 5 Kirchen zu planen. Skizzieren Sie für diesen Fall Vor- und Nachteile des A*-Algorithmus gegenüber dem Dijkstra-Algorithmus.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

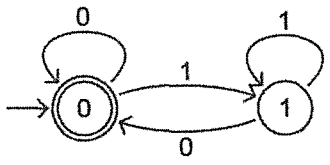
Aufgabe 1:
Für $w \in \{a, b\}^*$ bezeichne w^R das „gespiegelte" Wort zu dem Wort w (d.h. das Wort, das entsteht, wenn die Buchstaben von w in umgekehrter Reihenfolge geschrieben werden, z.B. $aaaba^R = abaaa$).
Geben Sie eine kontextfreie Grammatik für folgende Sprache über dem Alphabet $\{a,b\}$ an!

$$L_1 = \{vwv^R \mid v, w \in \{a, b\}^* \text{ und } |w| \text{ ist ungerade}\}$$

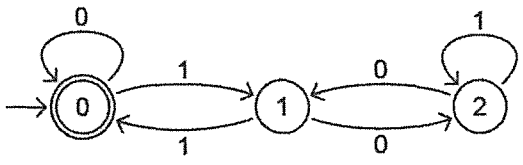
Aufgabe 2:
a) Für $w \in \{a, b\}^*$ bezeichne $|w|_a$ die Anzahl der Buchstaben a in w und $|w|_b$ die Anzahl der Buchstaben b in w. Ein Wort $u \in \{a, b\}^*$ heißt Präfix eines Wortes $w \in \{a, b\}^*$, falls es ein $v \in \{a, b\}^*$ mit der Eigenschaft $w = uv$ gibt.
Geben Sie einen deterministischen, endlichen Automaten an, der folgende Sprache akzeptiert!

$$L_2 = \{w \in \{a, b\}^* \mid |w|_a = |w|_b \text{ und für jedes Präfix } u \text{ von } w \text{ gilt } -2 \leq |u|_a - |u|_b \leq 2\}$$

b) Gegeben seien die folgenden deterministischen, endlichen Automaten A und B. Die von ihnen akzeptierten Sprachen bezeichnen wir mit L_A und L_B.



Automat A



Automat B

Konstruieren Sie einen nichtdeterministischen, endlichen Automaten, der folgende Sprache akzeptiert!

$$L_3 = \overline{L_A \cap L_B}$$

Aufgabe 3:
Beantworten Sie folgende Fragen und beweisen Sie Ihre Antworten!

a) Existiert eine berechenbare Funktion f: $D \to \mathbb{N}$, $D \subseteq \mathbb{N}$, mit endlichem Definitionsbereich und unentscheidbarem Wertebereich?

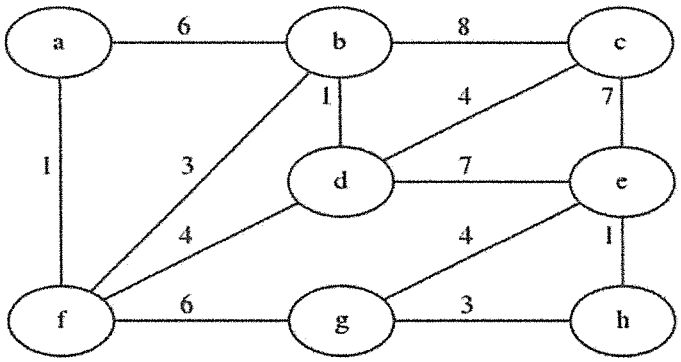
b) Existiert eine berechenbare Funktion g : $D \to \mathbb{N}$, $D \subseteq \mathbb{N}$, mit unentscheidbarem Definitionsbereich und entscheidbarem Wertebereich?

c) Existiert eine berechenbare Funktion h : $\mathbb{N} \to \mathbb{N}$ mit entscheidbarem Definitionsbereich und unentscheidbarem Wertebereich?

Fortsetzung nächste Seite!

Aufgabe 4:

Gegeben sei der folgende ungerichtete Graph $G = (V, E)$ mit Kantenlänge $l(e)$ für jede Kante $e \in E$.



a) Berechnen Sie mit einem Algorithmus Ihrer Wahl einen minimalen Spannbaum von G und geben Sie diesen an.

b) Nutzen Sie den Algorithmus von Dijkstra, um die kürzesten Wege vom Startknoten a zu allen anderen Knoten zu finden. Die Startdistanz jedes Knotens betrage $d(v) = \infty$ für $v \in V \setminus \{a\}$. Geben Sie für jeden Schritt den gewählten Knoten und ggf. die neue Distanz der bereits besuchten Knoten an.

c) Existiert ein Eulerpfad in G? Falls ja, geben Sie ein Beispiel an. Falls nein, begründen Sie!

d) Existiert ein Eulerkreis in G? Falls ja, geben Sie ein Beispiel an. Falls nein, begründen Sie!

Aufgabe 5:

Bei Bubblesort wird eine unsortierte Folge von Elementen a_1, a_2,..., a_n von links nach rechts durchlaufen, wobei zwei benachbarte Elemente a_i und a_{i+1} getauscht werden, falls sie nicht in der richtigen Reihenfolge stehen. Dies wird so lange wiederholt, bis die Folge sortiert ist.

a) Sortieren Sie die folgende Zahlenfolge mit Bubblesort. Geben Sie die neue Zahlenfolge nach jedem (Tausch-)Schritt an: 3, 2, 4, 1

b) Geben Sie den Bubblesort-Algorithmus für ein Array von natürlichen Zahlen in einer Programmiersprache Ihrer Wahl an. Die Funktion `swap(index1, index2)` kann verwendet werden, um zwei Elemente des Arrays zu vertauschen.

c) Geben Sie eine obere Schranke für die Laufzeit an. Beschreiben Sie mögliche Eingabedaten, mit denen diese Schranke erreicht wird.

Fortsetzung nächste Seite!

Aufgabe 6:

Es wird das Verhalten von Warteschlange (Queue), Keller (Stack) sowie Vorratswarteschlange (Priority Queue) untersucht.

Alle genannten Datentypen implementieren die Methoden push(x) und pop(). Dabei legt push(x) ein Element in der jeweiligen Datenstruktur nach deren Regeln ab, pop() hingegen entfernt ein Element. Für die Priorität der Vorratswarteschlange gilt:

Prioritätsbedingung 1: „x vor y" genau dann, wenn $x < y$.

Es werden nun auf allen drei Datenstrukturen die folgenden Operationen ausgeführt:

```
push(3), push(1), push(7), push(8), pop(), pop(), push(6), push(5),
pop(), push(2), pop(), pop(), pop(),pop()
```

a) Geben Sie den internen Speicherzustand der jeweiligen Datenstruktur nach jeder Operation in einer geeigneten Schreibweise an.

b) Skizzieren Sie, wie man die jeweilige Datenstruktur implementieren könnte. Was wären die resultierenden Laufzeiten für push(x) und pop()?