Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjar

2015

66115

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: 7

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Aufgabe 1:
Die Sprache L über dem Alphabet $\Sigma = \{0, 1\}$ enthält alle Wörter, bei denen beim Lesen von links nach rechts der Unterschied in der Zahl der 0en und 1en stets höchstens 3 ist. Also ist $w \in L$ genau dann, wenn für alle u, v mit $w = uv$ gilt $||u|_0 - |u|_1| \leq 3$. Erinnerung: $|w|_a$ bezeichnet die Zahl der $a's$ im Wort w.

a) Begründen Sie, dass L regulär ist.

b) Jemand behauptet, diese Sprache sei nicht regulär und gibt folgenden "Beweis" dafür an: *Wäre L regulär, so sei n eine entsprechende Pumping-Zahl. Nun ist $w = (01)^n \in L$. Zerlegt man nun $w = uxv$, wobei $u = 0$, $x = 1$, $v = (01)^{n-1}$, so ist zum Beispiel $ux^5v \notin L$, denn es ist $ux^5v = 011111010101\ldots$.*
Legen Sie genau dar, an welcher Stelle dieser "Beweis" fehlerhaft ist.

c) Sei $A = (Q, \delta, q_0, E)$ ein nichtdeterministischer endlicher Automat für L. Es sei $w_1 = 111$, $w_2 = 11$, $w_3 = 1$, $w_4 = \epsilon$, $w_5 = 0$, $w_6 = 00$, $w_7 = 000$. Machen Sie sich klar, dass der Automat jedes dieser Wörter verarbeiten können muss. Folgern Sie, dass der Automat mindestens sieben Zustände haben muss. Schreiben Sie Ihre Argumentation schlüssig und vollständig auf.

d) In anderen Fällen können nichtdeterministische endliche Automaten echt kleiner sein als die besten deterministischen Automaten. Ein Beispiel ist die Sprache $L_2 = \Sigma^*1\Sigma$ aller Wörter, deren vorletztes Symbol 1 ist. Geben Sie einen nichtdeterministischen Automaten mit nur drei Zuständen an, der L_2 erkennt.

e) Führen Sie auf Ihrem Automaten die Potenzmengenkonstruktion und anschließend den Minimierungsalgorithmus durch. Wie viele Zustände muss ein deterministischer Automat für L_2 also mindestens haben?

Aufgabe 2:
Gegeben sind eine Menge A von Angestellten und für jede Angestellte a eine Menge $F(a)$ von Fähigkeiten, die diese mitbringt. Außerdem gibt es eine Menge Z von Paaren von Angestellten, die nicht gut miteinander zusammenarbeiten. Für eine gegebene Menge P ("Projekt") von Fähigkeiten soll nun entschieden werden, ob eine Teilmenge T ("Team") der Angestellten existiert, sodass $P \subseteq \bigcup_{a \in T} F(a)$ und für kein Paar $(a, a') \in Z$ sowohl a, als auch a' in P sind. Alle Mengen sind endlich und durch explizite Aufzählung gegeben.

a) Begründen Sie, dass dieses Problem in NP liegt.

b) Zeigen Sie, dass das Problem NP-vollständig ist, indem Sie (zum Beispiel) eine Reduktion von 3SAT angeben.

Fortsetzung nächste Seite!

Aufgabe 3:

a) Geben Sie für jede der Chomsky-Sprachklassen I-IV eine Sprache an, die in der jeweiligen Klasse liegt, aber nicht in der nächstniedrigeren.

b) Nennen Sie auch eine Sprache, die nicht einmal in der Klasse IV liegt.

Aufgabe 4:

In der automatischen Programmverifikation kommt folgende graphentheoretische Grundaufgabe häufig vor.

Gegeben ist ein gerichteter Graph $G = (V, E)$, zwei Teilmengen $A, B \subseteq V$ und ein Startknoten $v_0 \in V$. Ein "Lasso" ist eine Folge von Knoten $v_0, v_1, v_2, \ldots, v_m, v_{m+1}, \ldots, v_n$ beginnend beim Startknoten mit $m < n$ und $(v_i, v_{i+1}) \in E$ für $i = 0, \ldots, n-1$ und $v_n = v_m$. Gefragt ist, ob ein Lasso existiert, in dem kein Knoten aus A vorkommt und dessen Schleife einen Knoten aus B enthält. Es soll also $A \cap \{v_0, \ldots, v_n\} = \emptyset$ und $B \cap \{v_m, v_{m+1}, \ldots, v_n\} \neq \emptyset$ sein.

(Bemerkung: Der Graph könnte die Zustände und Zustandsübergänge eines nebenläufigen Systems repräsentieren.)

a) Beschreiben Sie, wie man dieses Problem unter Zuhilfenahme der Tiefensuche lösen kann.

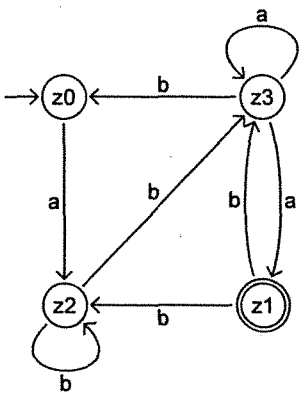
b) Schätzen Sie die Laufzeit Ihrer Lösung als Funktion von $|V|$ und $|E|$ unter Verwendung der O-Notation ab und begründen Sie Ihre Abschätzung. Ihre Lösung sollte mindestens die Schranke $O(|V|(|V| + |E|))$ erfüllen.

c) Man kann diese Aufgabe auch in Zeit $O(|V| + |E|)$ lösen. Diskutieren Sie mögliche Verbesserungsansätze Ihrer Lösung. Sie müssen das $O(|V| + |E|)$-Verfahren nicht finden oder kennen und dürfen in diesem Teil auch nicht vollständig durchgeführte Ideen ansprechen. Sollten Sie bereits oben ein $O(|V| + |E|)$-Verfahren gefunden haben, entfällt diese Teilaufgabe.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Aufgabe 1:
Gegeben ist der folgende nichtdeterministische, endliche Automat A, der Wörter über dem Alphabet $\{a, b\}$ verarbeitet. Die von A akzeptierte Sprache bezeichnen wir mit $L(A)$.



a) Geben Sie eine Typ-3-Grammatik (auch bekannt unter dem Namen *reguläre Grammatik*) für die Sprache $L(A)$ an. Dazu müssen Sie die Menge der Terminale, die Menge der Nichtterminale, das Startsymbol und die Regelmenge angeben.

b) Konstruieren Sie einen deterministischen, endlichen Automaten, der die Sprache $L(A)$ akzeptiert.

Aufgabe 2:
Beweisen Sie, dass folgende Menge nicht regulär ist.

$$B = \{w \in \{0\}^* \mid \text{die Länge von } w \text{ ist eine Zweierpotenz}\}$$

Aufgabe 3:
Für ein Wort w und einen Buchstaben e bezeichnet $|w|_e$ die Anzahl der Buchstaben e im Wort w.

$$C = \{w \in \{a, b, c\}^* \mid |w|_a + |w|_b = |w|_c\}$$

a) Geben Sie einen nichtdeterministischen Kellerautomaten an, der die Sprache C akzeptiert.

b) Erklären Sie die Arbeitsweise Ihres Kellerautomaten.

Aufgabe 4:
Sei $M_0, M_1, \ldots$ eine Gödelisierung aller Registermaschinen (RAMs). Geben Sie für jede der Mengen D_1, D_2, D_3 an, ob sie entscheidbar ist und ob sie aufzählbar ist. Beweisen Sie Ihre Behauptungen, wobei Sie die Aufzählbarkeit und Unentscheidbarkeit des speziellen Halteproblems

Fortsetzung nächste Seite!

$K_0 = \{x \in \mathbb{N} \mid M_x \text{ hält bei Eingabe } x\}$ verwenden dürfen.

$$\begin{aligned} D_1 &= \{x \in \mathbb{N} \mid x < 9973 \text{ und } M_x \text{ hält bei Eingabe } x\} \\ D_2 &= \{x \in \mathbb{N} \mid x \geq 9973 \text{ und } M_x \text{ hält bei Eingabe } x\} \\ D_3 &= \{x \in \mathbb{N} \mid M_x \text{ hält nicht bei Eingabe } x\} \end{aligned}$$

Aufgabe 5:

Gegeben seien die Standarddatenstrukturen Stapel (Stack) und Schlange (Queue) mit folgenden Standardoperationen:

Stapel	Schlange
Boolean Empty()	Boolean Empty()
Push(Zahl e)	Enqueue(Zahl e)
Zahl Pop()	Zahl Dequeue()
Zahl Top()	Zahl Head()

Beim Stapel gibt die Operation Top() das gleiche Element wie Pop() zurück, bei der Schlange gibt Head() das gleiche Element wie Dequeue() zurück. Im Unterschied zu Pop(), beziehungsweise Dequeue(), wird das Element bei Top() und Head() nicht aus der Datenstruktur entfernt.

a) Geben Sie in Pseudocode einen Algorithmus Sort(Stapel S) an, der als Eingabe einen Stapel S mit n Zahlen erhält und die Zahlen in S sortiert. (Sie dürfen die Zahlen wahlweise entweder aufsteigend oder absteigend sortieren.) Verwenden Sie als Hilfsdatenstruktur ausschließlich eine Schlange Q. Sie erhalten volle Punktzahl, wenn Sie außer S und Q keine weiteren Variablen benutzen. Sie dürfen annehmen, dass alle Zahlen in S verschieden sind.

b) Analysieren Sie die Laufzeit Ihrer Methode in Abhängigkeit von n.

Aufgabe 6:

Ein binäres Feld (ein Feld über den Zahlen 0 und 1) sei genau dann ein *gutes* Feld, wenn die Methode Boolean IstGut(Feld A) als Ausgabe true zurückgibt.

Die Laufzeit von IstGut(Feld A) ist linear zu der Anzahl der Zahlen in A. Sie dürfen in den folgenden Teilaufgaben die Methode IstGut(Feld A) aufrufen.

a) Geben Sie einen Algorithmus int AnzahlGute(int n) in Pseudocode an, der die Anzahl aller guten binären Felder der Länge n zurückgibt. Was ist die Laufzeit Ihres Algorithmus in Abhängigkeit von n?

Fortsetzung nächste Seite!

b) Geben Sie einen *iterativen* Algorithmus int AnzahlGutelt(int n) in Pseudocode an, der die Anzahl aller guten binären Felder der Länge n zurückgibt, die genau zwei Einser enthalten. Die Laufzeit soll in $O(n^3)$ sein. Sie brauchen die Laufzeit nicht zu zeigen.

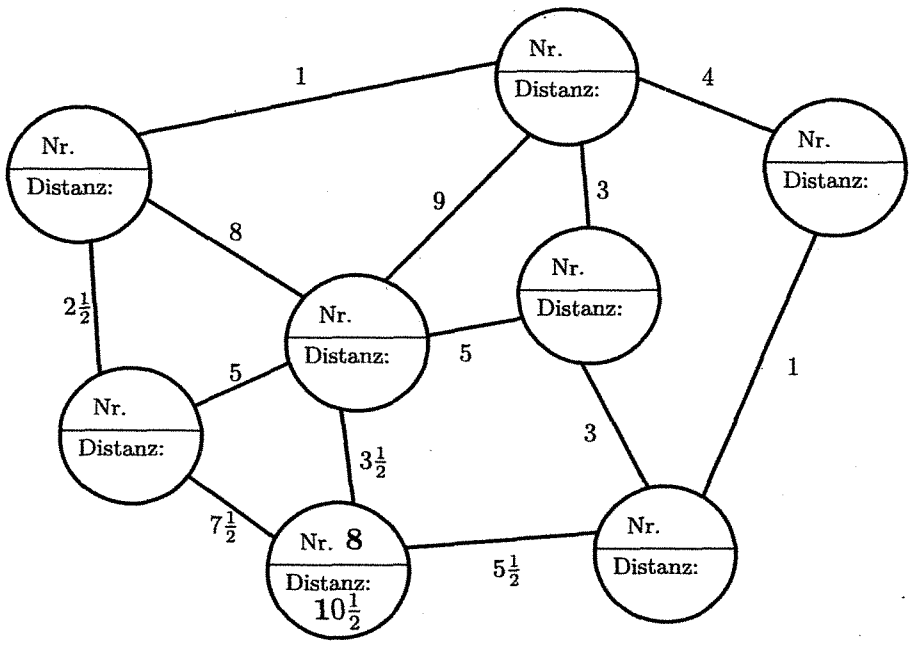
c) Geben Sie einen *rekursiven* Algorithmus int AnzahlGuteRek(int n, int k) in Pseudocode an, der die Anzahl aller guten binären Felder der Länge n zurückgibt, die genau k Einser enthalten. Die Laufzeit soll in $O(n\binom{n}{k})$ sein. Sie brauchen die Laufzeit nicht zu zeigen.

Aufgabe 7:

Auf folgendem ungerichteten, gewichteten Graphen wurde der Dijkstra-Algorithmus (wie auf der nächsten Seite beschrieben) ausgeführt, doch wir wissen lediglich, welcher Knoten als letztes schwarz (*black*) wurde (Nr. 8) und was seine Distanz zum Startknoten (Nr. 1) ist. Die Gewichte der Kanten sind angegeben.

Finden Sie zunächst den Startknoten, nummerieren Sie anschließend die Knoten in der Reihenfolge, in der sie schwarz wurden, und geben Sie in jedem Knoten die Distanz zum Startknoten an.

Hinweis: Der Startknoten ist eindeutig.



Fortsetzung nächste Seite!

Dijkstra(WeightedGraph G, Vertex s)

```
Initialize(G, s) ;
S = ∅ ;
Q = new PriorityQueue(V, d) ;
while not Q.Empty() do
    u = Q.ExtractMin() ;
    S = S ∪ {u} ;
    foreach v ∈ Adj[u] do
        Relax(u, v; w);
    u.color = black;
```

Initialize(Graph G, Vertex s)

```
foreach u ∈ V do
    u.color = white ;
    u.d = ∞ ;
s.color = gray ;
s.d = 0 ;
```

Relax$(u, v; w)$

```
if v.d > u.d + w(u, v) then
    v.color = gray ;
    v.d = u.d + w(u, v) ;
    Q.DecreaseKey(v, v.d)
```