Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr 2013

66115

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (vertieft studiert)**

Einzelprüfung: **Theoret. Informatik, Algorithmen**

Anzahl der gestellten Themen (Aufgaben): **2**

Anzahl der Druckseiten dieser Vorlage: **9**

Bitte wenden!

Thema Nr. 1

Aufgabe 1 Endliche Automaten und reguläre Sprachen

Gegeben sei folgender nicht-deterministischer endlicher Automat *M*:

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \delta, \{0,1\}, \{q_0\}, \{q_4\})$$

mit dem Eingabealphabet $\Sigma = \{0,1\}$, dem Startzustand q_0, dem Endzustand q_4 und der Übergangsfunktion δ definiert durch

$\delta\ (q_0,0) = \{q_0, q_1\}$,

$\delta\ (q_0,1) = \{q_0\}$,

$\delta\ (q_1,1) = \{q_2\}$,

$\delta\ (q_2,1) = \{q_3\}$,

$\delta\ (q_3,0) = \{q_4\}$,

$\delta\ (q_3,1) = \{q_2\}$,

$\delta\ (q_4,0) = \delta\ (q_4, 1) = \{q_4\}$

a) Geben Sie einen regulären Ausdruck für die von *M* akzeptierte Sprache *L(M)* an.

b) Geben Sie eine reguläre, ε-freie Grammatik *G* an, die die Sprache *L(M)* erzeugt.

c) Wandeln Sie den nicht-deterministischen endlichen Automaten *M* durch Teilmengenkonstruktion in einen deterministischen endlichen Automaten *N* um und minimieren Sie den Automaten *N* bzw. weisen Sie nach, dass der Automat *N* bereits minimal ist.

Fortsetzung nächste Seite!

Aufgabe 2 Kontextfreie Grammatiken

Gegeben sei die Grammatik $G = (V, \Sigma, P, S)$ mit V= {S, A, B, C}, Σ = {a,b}, dem Startsymbol S und

$P = \{S \rightarrow AB,\ S \rightarrow CS,\ A \rightarrow BC,\ A \rightarrow BB,\ A \rightarrow a,\ B \rightarrow AC,\ B \rightarrow b,\ C \rightarrow AA,\ C \rightarrow BA\}.$

$L = L(G)$ ist die von G erzeugte Sprache.

a) Zeigen Sie, dass G mehrdeutig ist.

b) Entscheiden Sie mithilfe des Algorithmus von Cocke, Younger und Kasami (CYK), ob das Wort $w = babaaa$ zur Sprache L gehört. Begründen Sie Ihre Entscheidung.

c) Geben Sie eine Ableitung für $w = babaaa$ an.

Aufgabe 3 Turing-Maschinen

Konstruieren Sie eine deterministische Turing-Maschine M, die die Sprache
$L = L(\gamma)\ mit\ \gamma = (01|10)^+$
entscheidet.
Geben Sie die Übergangsfunktion als Tabelle an und erläutern Sie die Bedeutung der Zustände in der von Ihnen konstruierten Maschine.

Aufgabe 4 Algorithmen

Die Potenz y einer Zahl x für zwei Zahlen x und y $(y \in \mathbb{N}_0, x \in \mathbb{Z})$ kann als fortgesetzte Multiplikation realisiert werden:

$$x^y = \exp(x,y) = \begin{cases} x \cdot x^{y-1} & falls\ y > 0 \\ 1 & falls\ y = 0 \end{cases}$$

a) Schreiben Sie eine rekursive Implementierung für diese Funktion in einer Programmiersprache Ihrer Wahl.

b) Implementieren Sie diese Funktion nun in der Programmiersprache Ihrer Wahl in iterativer Form.

c) Geben Sie für beide Algorithmen jeweils die Speicher- und Laufzeitkomplexität an.

Fortsetzung nächste Seite!

Aufgabe 5 Doppelt verkettete Listen und Sortieralgorithmen

a) Implementieren Sie in einer objektorientierten Programmiersprache Ihrer Wahl eine doppelt verkettete Liste für ganze Zahlen. Jedes Listenelement soll die Methoden `get()` und `set()` der Attribute `value`, `predecessor` und `successor` enthalten.

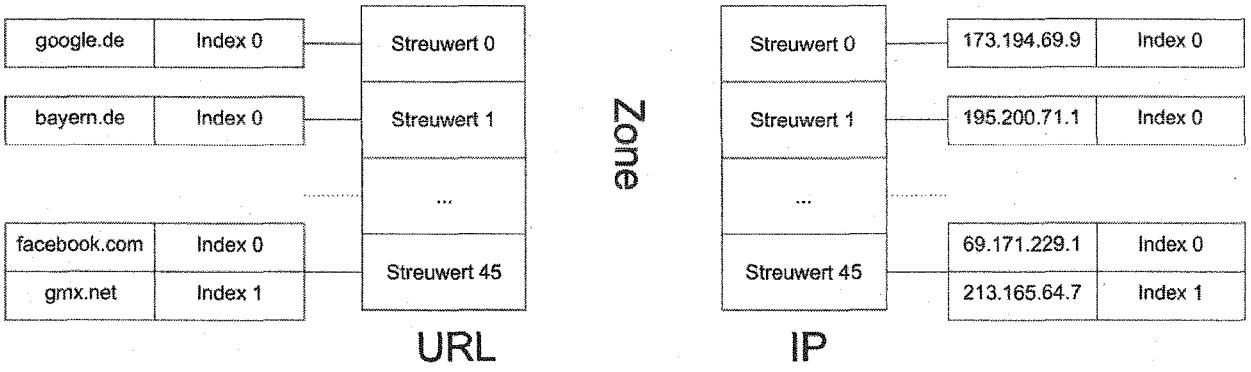
b) Implementieren Sie die Methoden `head()` und `tail()`, die das erste und letzte Element der Liste ausgeben.

c) Implementieren Sie eine Methode `sort()`, die die Liste nach den Werten von `value` aufsteigend sortiert.

d) Welche Komplexität hat Ihr Sortieralgorithmus beim Sortieren der doppelt verketteten Liste aus c jeweils im besten und schlechtesten Fall?

Aufgabe 6 Streutabellen (Hash-Tables)

Um die URL (zum Beispiel *google.de*) und die zugehörige IP des Servers (hier *173.194.69.9*) zu verwalten, werden Streutabellen verwendet, die eine bestimmte Zone von Adressen abbilden. Die Streutabellen werden als zwei dynamische Arrays (in Java: ArrayLists) realisiert. Kollisionen innerhalb einer Zone werden ebenfalls in dynamischen Arrays verwaltet.



Um zu einer URL die IP zu finden, berechnet man zunächst mittels der Funktion hash() den entsprechenden Streuwert, entnimmt dann den Index der Tabelle URL und sucht schließlich an entsprechender Stelle in der Tabelle IP die IP-Adresse.

Fortsetzung nächste Seite!

a) Erläutern Sie am vorgestellten Beispiel, wie ein Hash-Verfahren zum Speichern großer Datenmengen prinzipiell funktioniert und welche Voraussetzungen und Bedingungen daran geknüpft sind.

b) Nun implementieren Sie Teile dieser IP- und URL-Verwaltung in einer objektorientierten Sprache Ihrer Wahl. Verwenden Sie dabei die folgende Klasse (die Vorgaben sind in der Sprache Java gehalten):

```
class Zone {
   private ArrayList<ArrayList<String>> urlList =
         new ArrayList<ArrayList<String>>();
   private ArrayList<ArrayList<String>> ipList =
         new ArrayList<ArrayList<String>>();
   public int hash(String url) {/* calculates hash-value h, >=0 */}
}
```

i) Prüfen Sie in einer Methode `boolean exists(int h)` der Klasse `Zone`, ob bereits mindestens ein Eintrag für einen gegebenen Streuwert vorhanden ist. Falls `h` größer ist als die derzeitige Größe der Streutabelle, existiert der Eintrag nicht.

ii) Die Methode `int getIndex(string url, ArrayList<String> urlList)` soll den Index einer URL in der Kollisionsliste berechnen. Ist die URL in der Kollisionsliste nicht vorhanden, soll `-1` zurückgeliefert werden.

iii) Ergänzen Sie die Klasse Zone um eine Methode `String lookup(String url)`, die in der Streutabelle die IP-Adresse zur `url` zurückgibt. Wird eine nicht vorhandene Adresse abgerufen, wird eine Fehlermeldung zurückgegeben.

Thema Nr. 2

1. Es bezeiche $G = \langle V, T, P, A\rangle$ die kontextfreie Grammatik mit dem Variablenalphabet $V = \{A, B\}$, dem Terminalalphabet $T = \{a, b\}$, dem Startsymbol A und der Menge P von Produktionen:

$$P = \{\, A \to aBa \,|\, bBa \;,\; B \to aBa \,|\, bBa \,|\, \lambda \,\}$$

wobei λ für das leere Wort steht.

   (a) Welches ist die von G generierte Sprache $L(G)$? Geben Sie eine Beschreibung von $L(G)$, die nicht auf die Grammatik G Bezug nimmt. Beweisen Sie Ihre Behauptung!

   (b) Zeigen Sie dass die Sprache $L(G)$ nicht regulär ist.

   (c) Transformieren Sie die Grammatik G in eine äquivalente Grammatik G' in Chomsky-Normalform und geben Sie in G' eine Linksableitung für das Wort ab^2a^5 an.

2. Es sei $\Sigma = \{0, 1\}$. Das Alphabet Σ_2 bestehe aus allen Paaren von Elementen aus Σ, geschrieben als Spaltenvektoren der Länge 2 über Σ, also

$$\Sigma_2 = \left\{ \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix} \right\}$$

Ein Wort $w = w_1 w_2 \dots w_n \in \Sigma_2^n$ mit $w_i = \begin{bmatrix}x_i\\y_i\end{bmatrix}$, $1 \le i \le n$, kann aufgefasst werden als ein Paar $(x, y) \in \Sigma^n \times \Sigma^n$ mit $x = x_1 x_2 \dots x_n$, $y = y_1 y_2 \dots y_n$, d.h.

$$w = \begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}x_1\\y_1\end{bmatrix}\begin{bmatrix}x_2\\y_2\end{bmatrix}\dots\begin{bmatrix}x_n\\y_n\end{bmatrix}$$

Jedes Wort $a = a_1 a_2 \dots a_{n-1} a_n \in \Sigma^n$ stellt die natürliche Zahl

$$\mathtt{bin}(a) = a_1 \cdot 2^{n-1} + a_2 \cdot 2^{n-2} + \dots + a_{n-1} \cdot 2 + a_n$$

dar (binäre Zahldarstellung).

   (a) Die Sprache der "Multiplikation mit 3" ist

$$\mathtt{M3} := \left\{ w = \begin{bmatrix}x\\y\end{bmatrix} \in \Sigma_2^* \,;\, \mathtt{bin}(y) = 3 \cdot \mathtt{bin}(x) \right\}$$

   Es gilt also beispielsweise

$$\begin{bmatrix}0011\\1001\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} \in \mathtt{M3}, \quad \begin{bmatrix}011\\010\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} \notin \mathtt{M3}.$$

   Zeigen Sie, dass die Sprache M3 regulär ist.
   Hinweis: Sie können – falls Ihnen das hilfreich erscheint – hier die Tatsache verwenden, dass eine Sprache L genau dann regulär ist, wenn die gespiegelte Sprache $L^{\mathcal{R}} = \{w^{\mathcal{R}} \,;\, w \in L\}$ regulär ist. Dabei ist $(w_1 w_2 \dots w_{n-1} w_n)^{\mathcal{R}} = w_n w_{n-1} \dots w_2 w_1$.

   (b) Zeigen Sie, dass die Sprache

$$\left\{ w = \begin{bmatrix}x\\y\end{bmatrix} \in \Sigma_2^* \,;\, y = x^{\mathcal{R}} \right\}$$

   nicht regulär ist.

Fortsetzung nächste Seite!

3. Es sei $X = \{x_1, x_2, \ldots, x_n\}$ eine Variablenmenge und es bezeichne $\mathtt{AL}(X)$ die Menge der aussagenlogischen Formeln über X mit Konnektiven für die Konjunktion ($\wedge$), die Disjunktion ($\vee$) und die Negation ($\neg$). Es bezeichne $\mathtt{Sat}(X)$ die Menge der erfüllbaren Formeln und $\mathtt{Taut}(X)$ die Menge der Tautologien aus $\mathtt{AL}(X)$. Eine aussagenlogische Formel heisse 2-*erfüllbar*, wenn es mindestens zwei verschiedene Bewertungen ihrer Variablen gibt, die diese Formel wahr machen. $\mathtt{Sat}_2(X)$ bezeichne die Menge der 2-erfüllbaren aussagenlogischen Formeln über X.

   (a) Geben Sie eine induktive Definition für $\mathtt{AL}(X)$ in Form einer Grammatik an.

   (b) Geben Sie für den Fall $X = \{x_1, x_2, x_3\}$ Beispiele für Formeln folgender Art an:

      i. eine Formel F_1, die unerfüllbar ist, d.h. $F_1 \in \mathtt{AL}(X) \setminus \mathtt{Sat}(X)$;

      ii. eine Formel F_2, die erfüllbar, aber nicht 2-erfüllbar ist, d.h. $F_2 \in \mathtt{Sat}(X) \setminus \mathtt{Sat}_2(X)$;

      iii. eine Formel F_3, die 2-erfüllbar ist, aber keine Tautologie, d.h. $F_3 \in \mathtt{Sat}_2(X) \setminus \mathtt{Taut}(X)$;

      iv. eine Tautologie F_4, d.h. $F_4 \in \mathtt{Taut}(X)$.

   (c) Zeigen Sie, dass die Frage nach der Zugehörigkeit einer Formel zu $\mathtt{Sat}_2(X)$ (für beliebige Variablenmengen X) ein NP-vollständiges Problem ist. Konstruieren Sie hierfür eine polynomielle Reduktion zwischen dem bekannten Erfüllbarkeitsproblem Sat und dem $\mathtt{Sat}_2$-Problem, die dies beweist.

   Falls Sie die verlangte Reduktion nicht finden, so geben Sie möglichst genau an, aus welchen Daten eine solche besteht und was zu beweisen wäre.
   Hinweis: Verwenden Sie zur Formulierung von Algorithmen bzw. Datentypen eine gängige höhere Programmiersprache oder einen entsprechenden Pseudocode. Erläutern Sie Ihre Lösung ausgiebig durch Kommentare.

Fortsetzung nächste Seite!

Aufgabe 4

Für das bekannte rekursive Sortierverfahren Mergesort soll eine nicht-rekursive Variante nrMergesort entwickelt werden. Die Grundidee des Verfahrens besteht dabei darin, auf den rekursiven Abstieg zu verzichten und davon auszugehen, dass zu Beginn sortierte einelementige Listen hintereinander in einem Feld vorliegen. Im Verlauf des Verfahrens werden immer länger werdende, sortierte Teillisten miteinander solange gemischt, bis die gesamte Liste sortiert ist.

a) Geben Sie eine Implementierung des nicht-rekursiven Mergesort-Verfahrens an, welche zwei Felder verwendet.

b) Werden bei diesem Verfahren die gleichen Mischoperationen ausgeführt, wie beim rekursiven Mergesort? Begründen Sie Ihre Antwort.

c) Ist folgende Behauptung wahr oder falsch: Die Laufzeit von Mergesort (rekursive Variante) hängt nicht vom Wert der Schlüssel in der Eingabedatei ab. Begründen Sie Ihre Antwort.

Fortsetzung nächste Seite!

Aufgabe 5

Drei Missionare und drei Kannibalen befinden sich am Ufer eines Flusses und möchten diesen überqueren. Dazu steht ihnen ein Boot zur Verfügung, das ein oder zwei Personen befördern kann. Verbleiben an einem Ufer mehr Kannibalen als Missionare, so werden die Missionare verspeist. Die Frage besteht nun darin, wie *alle* Personen unversehrt auf die andere Seite des Flusses gelangen.

Im Anfangszustand befinden sich alle Personen und das Boot auf einer Seite des Flusses. Nehmen Sie an, es sei die linke Seite des Flusses. Im Zielzustand befinden sich alle Personen und das Boot auf der anderen Seite des Flusses. Jeden Zustand kann man durch folgendes Fünftupel beschreiben:

$$Missionare_{links} \times Kannibalen_{links} \times Missionare_{rechts} \times Kannibalen_{rechts} \times Bootposition$$

Dabei werden die Elemente für Missionare und Kannibalen durch natürliche Zahlen und die Bootsposition durch einen der Strings „links" oder „rechts" modelliert. Der Anfangszustand wäre somit also (3, 3, 0, 0, „links") und der Zielzustand (0, 0, 3, 3, „rechts").

Schreiben Sie Algorithmen zur Lösung dieses Suchproblems und retten Sie die Missionare! Es ist empfehlenswert (aber nicht zwingend), hierzu eine funktionale Programmiersprache zu verwenden. Gehen Sie im Einzelnen vor, wie folgt:

a.) Schreiben Sie eine Funktion, die alle möglichen Überfahrten von links nach rechts oder umgekehrt modelliert, d. h. die zu einem gegebenen Zustand die Liste der möglichen Folgezustände im Sinne der o. g. Regeln berechnet. Gehen Sie dazu von allen möglichen Überfahrten aus und überprüfen Sie für jede konkrete Überfahrt mittels einer geeigneter Funktion, ob diese zu einem zulässigen Zustand führt. Zustände, die die Missionare nicht überleben, gelten im Sinne des Rettungsvorhabens ebenfalls als nicht zulässig. Nicht zulässige Zustände werden nicht in die Liste der möglichen Folgezustände eingefügt.

b.) Geben Sie eine Funktion an, die feststellt, ob ein Zyklus vorliegt, d. h. ob ein Zustand in einer Liste bereits besuchter Zustände schon enthalten ist.

c.) Verwenden Sie Ihre Ergebnisse aus a.) und b.), um eine Funktion anzugeben, die dieses Suchproblem mittels Breitensuche löst. (Sie können die Funktionen aus a.) und b.) hier auch dann verwenden, wenn Sie diese Teilaufgaben nicht vollständig gelöst haben.) Die Funktion erhält als Eingabe einen Start- und einen Zielzustand und liefert als Ergebnis die erste gefundene Liste von Zuständen, die das Problem löst.