Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr

2006

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: 7

Thema Nr. 1

1. Spracheindeutigkeit

Gegeben sei folgende Grammatik:

$G : (\{S\}, \{+, x, y, z\}, \{S \to S+S, S \to z, S \to x, S \to y\}, S)$

1.1. Welche Sprache L wird von G erzeugt (ohne Beweis)?

1.2. Beweisen oder widerlegen Sie: Die Grammatik G ist eindeutig.

1.3. Ist die Sprache L eindeutig? Begründen Sie Ihre Antwort.

2. Chomsky-Hierarchie

Gegeben sei die folgende Grammatik:

$G := (\{S, M\}, \{x, \#\}, \{S \to SMx, S \to \#, xM \to Mx, \# M \to xx\#,\}, S)$

Fortsetzung nächste Seite!

2.1. Welchen Typ (Namen und Nummer in der Chomsky-Hierarchie) hat die Grammatik G?

2.2. Welche Sprache L wird von G erzeugt (ohne Beweis)?

2.3. Zu welchen Sprachtypen gehört die Sprache L und zu welchen gehört sie nicht? Zitieren Sie, soweit möglich, die Chomsky-Hierarchie. Geben Sie an der entscheidenden Stelle eine erzeugende Grammatik an und benutzen Sie das entsprechende Pumping-Lemma.

3. Turing-Maschinen

Eine deterministische Turingmaschine ist ein Tupel $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$. Dabei ist Q die endliche Zustandsmenge, Σ das endliche Eingabealphabet, Γ das endliche Bandalphabet, $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$ die Übergangsfunktion, $q_0 \in Q$ der Startzustand, B das Blanksymbol und $F \subseteq Q$ eine Menge von Endzuständen. Eine solche, deterministische Turingmaschine liest also in jedem Schritt das aktuelle Zeichen unter dem Schreib-/Lesekopf, entscheidet abhängig vom aktuellen Zustand und dem gelesenen Zeichen welches neue Zeichen geschrieben, in welchen Folgezustand übergegangen und ob dabei der Schreib-/Lesekopf nach rechts (R), nach links (L) oder gar nicht (N) bewegt werden soll.

Sei nun $\Sigma = (0,1)$ und $\Gamma = (0,1,B)$. Konstruieren Sie eine Turingmaschine, welche eine Zahl ungleich Null in Binärdarstellung um Eins dekrementiert und die Zahl Null ggf. unberührt lässt. Beachten Sie dabei die folgenden Vorgaben:
Auf dem Band steht die gegebene Zahl in Binärdarstellung mit dem niederwertigsten Bit ganz rechts. Führende Nullen sind zugelassen. Beispiel für die Dezimalzahl 13: $B\underline{B}00001101BB$. Der Schreib-/Lesekopf steht zu Beginn der Verarbeitung auf dem ersten B links von der Eingabe und soll auch am Ende wieder dort stehen.

Schreiben Sie die Übergangsfunktion δ in Tabellenform nieder, pro Zustand eine Spalte, pro Zeile ein Bandsymbol und dann in jeder Zelle den Folgezustand, das zu schreibende Zeichen sowie die Kopfbewegung.

4. Endliche Automaten

Gegeben sei folgender nichtdeterministischer, endlicher Automat A_a:

$$A_a = (Q, \Sigma, \delta, q_0 \{q_4\})$$
$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$$
$$\Sigma = \{a, b\}$$
$$\delta \subseteq (Q \times \Sigma) \times Q,\ \delta = \{((q_0, a), q_1), ((q_0, a), q_2), ((q_0, b), q_3), ((q_2, b), q_4),$$
$$((q_3, a), q_4), ((q_4, a), q_5), ((q_5, a), q_4), ((q_1, a), q_0)\}$$

Fortsetzung nächste Seite!

4.1. Stellen Sie den Automaten A_a graphisch dar.

4.2. Beschreiben Sie die vom Automaten A_a akzeptierte Sprache durch einen regulären Ausdruck (ohne Beweise).

4.3. Konstruieren Sie aus A_a einen äquivalenten, deterministischen, endlichen Automaten A_b und stellen Sie ihn graphisch dar.

4.4. Ist der in 4.3. konstruierte Automat minimal? Begründen Sie Ihre Antwort.
Ist der Automat noch nicht minimal, so geben Sie den Minimalautomaten in graphischer Form an.

5. Komplexitätstheorie

Sei $RUCKSACK := \left\{(A,g,w,G,W) \middle| A \text{ endliche Menge}, g : A \to \mathbb{N}, w : A \to \mathbb{N}, G \in \mathbb{N}, W \in \mathbb{N}\right\}$
und

$$RUCKSACK^+ := \left\{(A,g,w,G,W) \in RUCKSACK \middle| \exists B \subseteq A : \sum_{a \in B} g(a) \leq G \wedge \sum_{a \in B} w(a) \geq W\right\}.$$

Das *Rucksackproblem* besteht darin, für ein gegebenes Tupel $x = (A,g,w,G,W) \in RUCKSACK$ zu entscheiden, ob $x \in RUCKSACK^+$ gilt.

Sei $TEILE := \left\{A \middle| A \subset \mathbb{N} \text{ endlich}\right\}$ und $TEILE^+ := \left\{A \in TEILE \middle| \exists B \subseteq A : \sum_{b \in B} b = \sum_{b \in A \setminus B} b\right\}$.

Das *Teileproblem* besteht darin, für eine gegebene Menge $A \in TEILE$ zu entscheiden, ob $A \in TEILE^+$ gilt. Das *Teileproblem* ist NP-vollständig.

5.1. Beschreiben Sie einen nichtdeterministischen Algorithmus zur Lösung des Rucksackproblems in polynomieller Zeit abhängig von der Länge der Eingabe.

5.2. Zeigen Sie, dass das *Teileproblem* polynomiell auf das Rucksackproblem reduziert werden kann $TEILE^+ \leq_{pol} RUCKSACK^+$.

5.3. Schließen Sie daraus formal die NP-Vollständigkeit des Rucksackproblems.

Fortsetzung nächste Seite!

6. Verifikation

Gegeben ist folgender Algorithmus welcher verifiziert werden soll:

$//\{x \geq 0\} = P$
$y = 0$
$z = 0$
while $(z \leq x - 1)$
{
$y = y + z + z + 1$
$z = z + 1$
}
$//\{\ldots\} = Q$

6.1. Geben Sie die Bedingungen an, welche nach dem Ablauf der while-Schleife für das Prädikat Q gelten und geben Sie die Schleifeninvariante an. Begründen Sie Ihre Antwort.

6.2. Führen Sie für diesen Algorithmus eine Verifikation durch, wobei die einzelnen Beweisschritte mit den Regeln der axiomatischen Semantik ausführlich zu beschreiben und zu begründen sind.

7. Rekursion und Iteration

7.1. Es gibt zwei Möglichkeiten die Fakultät $F : \mathbb{N}^+ \rightarrow \mathbb{N}^+, mit\, F(n) = \prod_{i=1}^{n} i$ zu berechnen.

Zum einen kann sie iterativ und zum anderen rekursiv berechnet werden. Geben Sie für jede der Möglichkeiten jeweils eine Methode in einer höheren Programmiersprache an.

7.2. In der Linearen Algebra ist die Determinante eine Funktion, die jeder quadratischen Matrix eine Zahl zuordnet. Zum Beispiel hat die 2x2-Matrix

$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ die Determinante $\det(A) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$

Allgemein wird die Determinante einer $n \times n$ Matrix berechnet, indem aus einer gewählten Zeile i jedem Element $a_{i,1}, \ldots a_{i,n}$ eine Untermatrix A_{ij} gebildet wird. Die Untermatrix A_{ij} entsteht aus A durch Streichen der i-ten Zeile und j-ten Spalte. Für die Determinante von A gilt: $\det(A) = \sum_{j=1}^{n} (-1)^{i+j} \cdot a_{ij} \cdot \det\left(A_{ij}\right)$.

Schreiben Sie eine rekursive Funktion in einer höheren Programmiersprache, welche die Determinante der Matrix A bestimmt. Die Funktionsdeklaration könnte in C wie folgt aussehen: $\texttt{float det}\left(\texttt{float[ ][ ] Matrix, int dimension}\right)$

Fortsetzung nächste Seite!

8. Listen, Bäume, Komplexität

8.1. Nennen Sie den Aufwand von sortierten Listen, balancierten Suchbäumen und sortierten Arrays in Bezug zueinander bei der Speicherung von n Elementen für die Operationen Element finden und Element einfügen bzw. löschen. Geben Sie ebenfalls den zusätzlichen Speicherbedarf für die Verwaltung der Datenstrukturen an.

Kriterium	Listen	Bäume	Arrays
Element finden			
Element einfügen bzw. löschen			
zusätzlicher Speicherbedarf			

8.2. Gesucht ist ein nicht notwendigerweise balancierter binärer Suchbaum, bei welchem in jedem Knoten eine Zahl $n \in \mathbb{N}$ gespeichert ist. Für jeden Knoten gilt, dass alle Knoten, welche an seinem linken (rechten) Ast hängen, kleinere (größere) Elemente als n gespeichert haben. Ferner gibt es in dem Suchbaum keine doppelten Elemente n.
Geben Sie eine Methode `insert ( n )` zum Einfügen und eine Methode `search ( n )` zum Suchen eines Elementes an. Fügen Sie mit der Methode `insert ( n )` die Elemente 5, 14, 2, 8, 14, 7 in einen leeren Baum ein und geben Sie den Baum nach jedem Einfügen eines Elementes an.

8.3. Im Weinkeller eines grausamen Königs befinden sich n wertvolle Weinflaschen. Seine Wächter haben einen Hexer gefangen genommen, der genau eine Flasche vergiftet hat. Unglücklicherweise wissen sie nicht welche. Das Gift ist jedoch so stark, dass man sogar dann sterben würde, wenn man den Wein aller Flaschen vermischt und davon kostet. Allerdings wirkt das Gift so langsam, dass man erst einen Monat später daran erkrankt. Mit welcher Methode könnte der König innerhalb eines Monats feststellen, welche Flasche vergiftet ist und dabei höchstens $O(\log n)$ Vorkoster einsetzen?

Thema Nr. 2

Teilaufgabe I

Wir fixieren das Alphabet $\sum = \{0,1\}$ und definieren $L \subseteq \sum^*$ durch $L = \{w |$ in w kommt genau einmal das Teilwort 0010 vor $\}$

1. Zeigen Sie, dass L regulär ist!

2. Geben Sie die Äquivalenzklassen der Myhill-Nerode Äquivalenz von L durch Repräsentanten an. (Diese Äquivalenz ist definiert durch $x \sim_L y \Leftrightarrow \forall u.xu \in L \Leftrightarrow yu \in L$

3. Zeichnen Sie den Minimalautomaten für L.

Teilaufgabe II

Sei $\sum_n$ das Alphabet $\{0,1\}^n$. Ein Buchstabe in $\sum_n$ ist also ein n-Tupel von 0en und 1en; das Alphabet $\sum_n$ hat 2^n Buchstaben. Ist $w \in \sum_n^*$ so bezeichne w_1 das Wort über $\sum$, das aus den ersten Komponenten der Buchstaben in w besteht, formal also $w_1 = f(w)$ für den durch $f\left((x_1,\ldots x_n)\right) = x_1$ definierten Homomorphismus f.

Analog definiert man w_2, w_3, etc. Also gilt für $w = (0,1,1)(1,1,0)(0,0,1)(1,1,1)$, dass $w_1 = 0101, w_2 = 1101, w_3 = 1011$. Für $w \in \sum_1^*$ bezeichne num $(w) \in \mathbb{N}$ die Bedeutung von w aufgefasst als von rechts nach links gelesene Binärdarstellung. Also num (10110000) = 13. Die "Inverse" (bis auf Nullen am Ende) von num bezeichnen wir mit bin, also bin(19) = 11001.

1. Zeigen Sie, dass $L = \left\{w : \sum_3^* \middle| \text{num}\left(w_3\right) = \text{num}\left(w_1\right) + \text{num}\left(w_2\right)\right\}$ regulär ist.

2. Für $u \in \sum^*$ und $v \in \sum_n^*$ sei $(u,v) \in \sum_{n+1}^*$ das durch

$$|(u,v)| = \max(|u|,|v|)$$
$$\text{num}\left((u,v)_1\right) = \text{num}(u)$$
$$\text{num}\left((u,v)_{i+1}\right) = \text{num}\left(v_i\right)$$

eindeutig definierte Wort.
Sei $L \subseteq \sum_{n+1}^*$ regulär. Es sei

$$L' = \left\{w \in \sum_n^* | \text{ es existiert } n \in \mathbb{N}, \text{ sodass } (\text{bin}(n), w) \in L\right\}$$

Zeigen Sie, dass L' regulär ist.

3. Programmieren Sie die vier Funktionen bin, num, $w \mapsto w_i$ und $u,v \mapsto (u,v)$ in einer funktionalen Programmiersprache Ihrer Wahl oder Pseudocode. Zur Beachtung: das möglicherweise erforderliche Auffüllen von v mit Nullen bei der Berechnung von (u, v) beinhaltet eine gewisse Schwierigkeit, da $v \in \sum_n^*$ für beliebiges n. Es bietet sich an, die Tupel als Listen zu implementieren. Programmieren Sie auch eine boole'schwertige Funktion, die die Zugehörigkeit zur in 1) definierten Sprache prüft.

Fortsetzung nächste Seite!

Teilaufgabe III

Es soll ein Mühlespiel programmiert werden.
Die beiden Spieler sollen abwechselnd mit der Maus Züge wählen; das Programm soll überprüfen, ob ein Zug erlaubt ist und das Ergebnis des Zuges auf dem Bildschirm anzeigen.

1. Skizzieren Sie einen objektorientierten Entwurf für diese Aufgabenstellung in Form eines Klassendiagramm (in UML o.ä.). Ihr Diagramm sollte nicht weniger als fünf Klassen haben und neben den Beziehungen der Klassen untereinander auch deren wichtigste Methoden und Attribute beinhalten.

2. Das Programm muss mehrere Zustände haben, aus denen hervorgeht, welcher Spieler am Zug ist, ob die Maus gedrückt wurde, ob das Spiel zu Ende ist, etc. Modellieren Sie diese Programmzustände als Ablaufdiagramm.