Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr 2014

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: **13**

Bitte wenden!

Thema Nr. 1

Aufgabe 1: „Rekursion und Induktion"

a) Gegeben sei die Methode **`BigInteger lfBig(int n)`** zur Berechnung der eingeschränkten Linksfakultät:

$$!n := \begin{cases} n \cdot !(n-1) - (n-1) \cdot !(n-2) & falls \quad 1 < n < 32767 \\ 1 & falls \quad n = 1 \\ 0 & sonst \end{cases}$$

```
import java.math.BigInteger;
import static java.math.BigInteger.*;

public class LeftFactorial {
	// returns the left factorial !n
	BigInteger lfBig(int n) {
		if (n <= 0 || n >= Short.MAX_VALUE) {
			return ZERO;
		} else if (n == 1) {
			return ONE;
		} else {
			return sub(mul(n, lfBig(n - 1)), mul(n - 1, lfBig(n - 2)));
		}
	}
}
```

Implementieren Sie unter Verwendung des Konzeptes der *dynamischen Programmierung* die Methode **BigInteger dp(int n)**, die jede $!n$ auch bei mehrfachem Aufrufen mit dem gleichen Parameter höchstens einmal rekursiv berechnet. Sie dürfen der Klasse **LeftFactorial** genau ein Attribut beliebigen Datentyps hinzufügen und die in **lfBig(int)** verwendeten Methoden und Konstanten ebenfalls nutzen.

Fortsetzung nächste Seite!

b) Betrachten Sie nun die Methode **`lfLong(int)`** zur Berechnung der vorangehend definierten Linksfakultät ohne obere Schranke. Nehmen Sie im Folgenden an, dass der Datentyp **`long`** unbeschränkt ist und daher kein Überlauf auftritt.

```
long lfLong(int n) {
	if (n <= 0) {
		return 0;
	} else if (n == 1) {
		return 1;
	} else {
		return n * lfLong(n - 1) - (n - 1) * lfLong(n - 2);
	}
}
```

Beweisen Sie *formal* mittels *vollständiger Induktion*:

$$\forall n \geq 0 : lfLong(n) \equiv \sum_{k=0}^{n-1} k\,!$$

Fortsetzung nächste Seite!

Aufgabe 2: „Binäre Bäume"

Implementieren Sie in einer objekt-orientierten Sprache Ihrer Wahl eine Klasse namens `BinBaum`, deren Instanzen binäre Bäume mit ganzzahligen Datenknoten darstellen, nach folgender Spezifikation:

a) Beginnen Sie zunächst mit der Datenstruktur selbst:

- Mit Hilfe des Konstruktors soll ein neuer Baum erstellt werden, der aus einem einzelnen Knoten besteht, in dem der dem Konstruktor als Parameter übergebene Wert (ganzzahlig, in Java z.B. `int`) gespeichert ist.
- Die Methoden `setLeft(int value)` bzw. `setRight(int value)` sollen den linken bzw. rechten Teilbaum des aktuellen Knotens durch einen jeweils neuen Teilbaum ersetzen, der seinerseits aus einem einzelnen Knoten besteht, in dem der übergebene Wert `value` gespeichert ist. Sie haben keinen Rückgabewert.
- Die Methoden `getLeft()` bzw. `getRight()` sollen den linken bzw. rechten Teilbaum zurückgeben (bzw. `null`, wenn keiner vorhanden ist).
- Die Methode `int getValue()` soll den Wert zurückgeben, der im aktuellen Wurzelknoten gespeichert ist.

b) Implementieren Sie nun die Methoden `preOrder()` bzw. `postOrder()`. Sie sollen die Knoten des Baumes mit Tiefensuche traversieren, die Werte dabei in pre-order bzw. post-order Reihenfolge in eine Liste (z.B. `List<Integer>`) speichern und diese Ergebnisliste zurückgeben.Die Tiefensuche soll dabei zuerst in den linken und dann in den rechten Teilbaum absteigen.

c) Ergänzen Sie schließlich die Methode `isSearchTree()`.Sie soll überprüfen, ob der Binärbaum die Eigenschaften eines *binären Suchbaums* erfüllt. Beachten Sie, dass die Laufzeit-Komplexität Ihrer Implementierung linear zur Anzahl der Knoten im Baum sein muss.

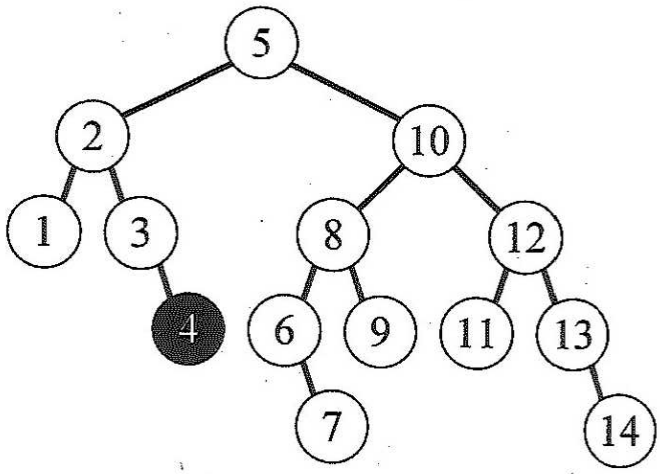
Fortsetzung nächste Seite!

Aufgabe 3: „AVL-Bäume“

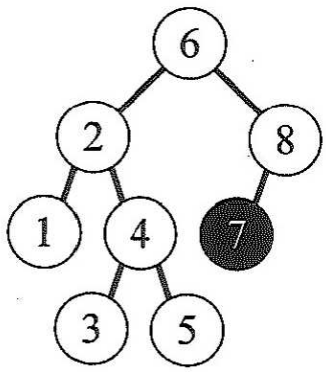
a) Fügen Sie die Zahlen (7, 6, 2, 1, 5, 3, 8, 4) in dieser Reihenfolge in einen anfangs leeren AVL Baum ein. Stellen Sie die AVL Eigenschaft ggf. nach jedem Einfügen mit geeigneten Rotationen wieder her. Zeichnen Sie den AVL Baum einmal vor und einmal nach jeder einzelnen Rotation.

b) Entfernen Sie den jeweils markierten Knoten aus den folgenden AVL-Bäumen. Stellen Sie die AVL-Eigenschaft ggf. durch geeignete Rotationen wieder her. Zeichnen Sie nur den resultierenden Baum.

i) Baum 1:

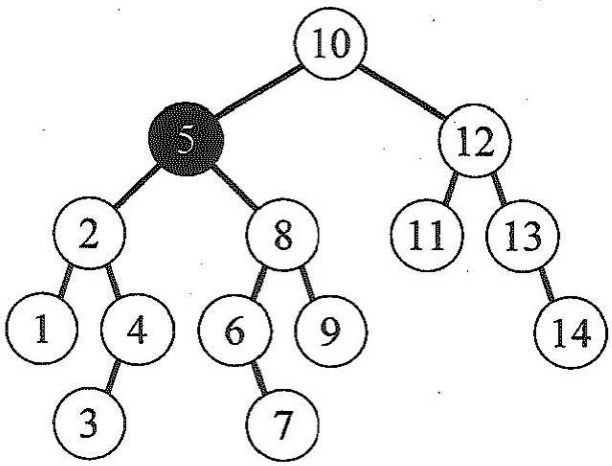


ii) Baum 2:



Fortsetzung nächste Seite!

iii) Baum 3:



Fortsetzung nächste Seite!

Aufgabe 4:

Zeigen oder widerlegen Sie die folgenden Aussagen (die jeweiligen Beweise sind sehr kurz):

a)Sei $L \subseteq \Sigma^*$. Ist L regulär, so gilt das Pumping-Lemma für kontextfreie Sprachen auch für L.

b)Sei H das (allgemeine) Halteproblem (kodiert mit Zeichen über Σ), und sei $\overline{H} = \Sigma^* \setminus H$ das Komplement des (allgemeinen) Halteproblems. Dann kann $\overline{H}$ auf H reduziert werden, d. h. es gilt: $\overline{H} \leq H$.

c)Der Standard-Beweis dafür, dass CLIQUE NP-schwer ist, zeigt, dass SAT $\leq_p$ CLIQUE. Stimmt es auch, dass CLIQUE $\leq_p$ SAT ist?

Schreiben Sie zuerst zur Aussage „Stimmt“ oder „Stimmt nicht“ und dann Ihre Begründung.

Fortsetzung nächste Seite!

Aufgabe 5:

a) Definieren Sie die zum Halteproblem für Turing-Maschinen bei fester Eingabe $m \in \mathbb{N}_0 = \{0, 1, 2, \ldots\}$ gehörende Menge H_m.

b) Gegeben sei das folgende Problem E:

- Entscheide, ob es für die deterministische Turing-Maschine mit der Gödelnummer n eine Eingabe $w \in \mathbb{N}_0$ gibt, so dass w eine *gerade* Zahl ist und die Maschine n gestartet mit w hält.

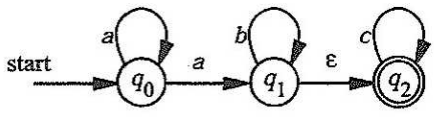
Zeigen Sie, dass E nicht entscheidbar ist. Benutzen Sie, dass H_m aus (a) für jedes $m \in \mathbb{N}_0$ nicht entscheidbar ist.

c) Zeigen Sie, dass das Problem E aus (b) partiell-entscheidbar (= rekursiv aufzählbar) ist.

Fortsetzung nächste Seite!

Aufgabe 6:

a)Gegeben sei der folgende nichtdeterministische endliche Automat N:



Konstruieren Sie zu N mit den Potenzmengen-Konstruktionen einen äquivalenten deterministischen endlichen Automaten A. Zeichnen Sie nur die vom Startzustand erreichbaren Zustände ein, die aber *alle*. Die Zustandsnamen von A müssen erkennen lassen, wie sie zustande gekommen sind. Führen Sie *keine* „Vereinfachungen" durch.

Hinweis: In einem deterministischen endlichen Automaten darf es keine ε-Übergänge geben, und es muss an jedem Zustand für jedes Zeichen einen Übergang geben.

b)Geben Sie einen regulären Ausdruck $\alpha(N)$ für die Sprache, die der nichtdeterministische endliche Automat N aus (a) akzeptiert, an.

c)Beweisen oder widerlegen Sie die folgende Behauptung.

Beh.: Ist die Sprache L nicht regulär, dann ist auch jede echte Obermenge von L nicht regulär.

d)Sei A ein deterministischer endlicher Automat mit Zustandsmenge $Q = \{q_1, \ldots, q_n\}$, Startzustand q_1, Alphabet Σ und Endzustandsmenge F.

Im Zusammenhang der Umwandlung endlicher Automaten in reguläre Ausdrücke wird die Menge $R_{ij}^k = \{w \in \Sigma^* \mid \delta(q_i, w) = q_j$, kein echter Zwischenzustand hat Index $> k\}$ betrachtet.

d1) Wie lautet die Rekursionsformel für die R_{ij}^k, die die Grundlage des Umwandlungsalgorithmus ist?

Hinweis: Es werden die drei Fälle unterschieden: (i) $k = 0$, $i \neq j$, (ii) $k = 0$, $i = j$ und (iii) der Sonst-Fall.

d2) Wie ergibt sich aus den Mengen R_{ij}^k dann die Sprache $L(A)$ des Automaten?

Fortsetzung nächste Seite!

Aufgabe 7:

a) Sei SAT das Erfüllbarkeitsproblem, und sei H_m das Halteproblem bei fester Eingabe m (siehe Aufgabe 5).

Zeigen Sie: SAT kann in *polynomieller* Zeit auf H_m reduziert werden. (Als Relation: $\text{SAT} \leq_\text{p} H_m$)

b) Angenommen, es wurde gezeigt, dass P = NP ist. Zeigen Sie, dass dann *jede* Sprache $L \in \text{P}$ über dem Alphabet Σ mit $L \neq \emptyset$ und $L \neq \Sigma^*$ sogar NP-vollständig ist.

Thema Nr. 2

Teilaufgabe I

Seien Σ, Δ Alphabete. Ein Homomorphismus ist eine Funktion $f : \Sigma^* \to \Delta^*$ sodass $f(uv) = f(u)f(v)$ für alle $u, v \in \Sigma^*$. Es ist klar, dass dann f schon eindeutig durch die Werte $f(a)$ für $a \in \Sigma$ bestimmt ist.

Seien $f, g : \Sigma^* \to \Delta^*$ Homomorphismen. Der *Differenzkern* $E(f, g) \subseteq \Sigma^*$ ist definiert durch

$$E(f, g) = \{w \mid f(w) = g(w)\}$$

Ist zum Beispiel $f(a) = 00$, $f(b) = 0$, $g(a) = 0$ und $g(b) = 00$, so gilt für $w = ababab$ dass $f(w) = 00\ 0\ 00\ 0\ 00\ 0 = 0\ 00\ 0\ 00\ 0\ 00 = g(w)$ und es ist hier $E(f, g) = \{w \mid 2|w|_a + |w|_b = |w|_a + 2|w|_b\} = \{w \mid |w|_a = |w|_b\}$. Wie üblich ist $|w|_x$ die Zahl der Buchstaben x im Wort w.

Von den folgenden vier Aussagen ist genau eine falsch, die anderen drei sind wahr. Identifizieren Sie die falsche Aussage, erklären Sie, warum sie falsch ist und zeigen Sie, dass die anderen drei Aussagen wahr sind.

1. Seien f, g Homomorphismen. Falls $E(f, g)$ durch einen nichtdeterministischen Kellerautomaten (PDA) erkannt wird, so ist $E(f, g)$ kontextfrei.
2. Für alle f, g kann $E(f, g)$ durch einen PDA wie folgt erkannt werden: Sei die Eingabe aw mit $a \in \Sigma$, $w \in \Sigma^*$. Man vergleiche $f(a)$ mit $g(a)$. Falls keines von beiden Präfix des anderen ist, so verwerfe man; anderenfalls lege man die Differenz der beiden auf den Keller, merke sich, auf welcher Seite noch etwas fehlt und fahre fort, wobei dann natürlich der Kellerinhalt auch noch zu berücksichtigen ist.
3. Die Sprache $L = \{w \mid |w|_a = |w|_b = |w|_c\}$ ist nicht kontextfrei.
4. Für L aus 3) gilt $L = E(f, g)$ mit $f, g : \Sigma^* \to \{0, 1\}^*$ definiert durch $f(a) = 00, f(b) = 1, f(c) = 1, g(a) = 0, g(b) = 0, g(c) = 11$.

Fortsetzung nächste Seite!

Teilaufgabe II

Wir definieren das Problem INTPOLY wie folgt:

GEGEBEN: Eine Liste von Variablen $(x_1, \ldots, x_n)$ und ein Polynom $p(x_1, \ldots, x_n)$ in diesen Variablen und mit ganzzahligen Koeffizienten.

GEFRAGT: Hat dieses Polynom eine ganzzahlige Nullstelle, d.h.. existieren ganze Zahlen $z_1, \ldots, z_n$, sodass $p(z_1, \ldots, z_n) = 0$.

So sind beispielsweise $((x, y, z), x^2 + y^2 - z^2)$ und $((x, y, z), x^2 - 2xy + y^2 + 1)$ beides Instanzen, aber nur die erste ist eine JA Instanz, also formal ein Element von INTPOLY. Es ist z.B. $3^2 + 4^2 - 5^2 = 0$, aber $x^2 - 2xy + y^2 + 1 = (x - y)^2 + 1 > 0$ für alle x, y, z.

1. Zeigen Sie durch Reduktion von 3SAT, dass INTPOLY NP-schwierig ist. Zeigen Sie also, dass 3SAT $\leq_P$ INTPOLY. Legen Sie zunächst dar, aus welchen Daten so eine Reduktion besteht und was über diese Daten zu zeigen ist.

   Weitere Hinweise: $p(\vec{x})q(\vec{x}) = 0 \iff p(\vec{x}) = 0 \vee q(\vec{x}) = 0$ und $p(\vec{x})^2 + q(\vec{x})^2 = 0 \iff p(\vec{x}) = 0 \wedge q(\vec{x}) = 0$. Für jede Boole'sche Variable A bietet es sich an, zwei ganzzahlige Variablen x_A und $x_{\neg A}$ einzuführen.

2. Interessanterweise ist INTPOLY ein unentscheidbares Problem (das wurde in den 1970er Jahren recht aufwendig bewiesen). Nun behauptet aber jemand, INTPOLY sei in NP, denn, wenn $\vec{x}$ und $p(\vec{x})$ gegeben sind, dann könne man ja nichtdeterministisch ganze Zahlen $z_1, \ldots, z_n$ raten und dann prüfen, ob $p(\vec{z}) = 0$ ist. Beschreiben Sie genau, an welcher Stelle diese Person irrt.

3. Gibt es unentscheidbare Probleme in NP ?

Fortsetzung nächste Seite!

Teilaufgabe III

Gegeben sei ein Array $A[1], \ldots, A[n]$ von Zahlen. Es soll ein zweidimensionales Array $B[1,1] \ldots B[n,n]$ bestimmt werden, sodass gilt

$$B[i,j] = A[i] + A[i+1] + \cdots + A[j]$$

falls $i \leq j$ und $B[i,j] = 0$, falls $i > j$. Folgender Algorithmus leistet das Verlangte:

```
for i := 1,...,n
  for j := 1,...,n
    B[i,j]:=0;
    for t := i...j
      B[i,j]:=B[i,j]+A[t];
```

- Geben Sie eine Funktion $f(n)$ an, sodass die Laufzeit dieses Algorithmus $\Theta(f(n))$ ist. Sollte Ihnen die Θ-Notation nicht geläufig sein, so können Sie alternativ eine *möglichst schwach wachsende* Funktion f bestimmen, sodass die Laufzeit $O(f(n))$ ist.
- Finden Sie einen Algorithmus für dieselbe Problemstellung mit einer echt besseren Laufzeit. Gesucht ist also ein Algorithmus für unser Problem mit einer Laufzeit $O(g(n))$ wobei $\lim_{n\to\infty} g(n)/f(n) = 0$. Geben Sie sowohl den Algorithmus, als auch die neue Laufzeitschranke $g(n)$ an.

[Nach Kleinberg-Tardos. Algorithm Design. Pearson 2005]