Schriftliche Abiturprüfung 2012

Fach: Informatik
Prüfungsart: G-Kurs-Niveau
Dauer: 3 Stunden
Hilfsmittel: Taschenrechner

Aufgabe 1:

1.1 Strukturierte Programmierung, Objektorientierung

Vektoren des $\mathbb{R}^3$ haben drei reellwertige Koordinaten.
Man kann sie addieren, subtrahieren und mit reellen Zahlen multiplizieren. Zusätzlich gibt es zwei Arten der Multiplikation zweier Vektoren: das Skalarprodukt, das eine reelle Zahl ergibt, und das Vektorprodukt (Kreuzprodukt), das einen Vektor ergibt. Darüber hinaus kann man den Betrag eines Vektors berechnen und testen, ob ein Vektor orthogonal oder parallel zu einem anderen ist.

1.1.1 Stellen Sie eine Klasse `TVektor` in Form eines Klassendiagramms dar!

1.1.2 Implementieren Sie die Methoden `skalarprodukt(v: TVektor)` und `istOrthogonalZu(v: TVektor)` der Klasse `TVektor` in Delphi oder `skalarprodukt(TVektor v)` und `istOrthogonalZu(TVektor v)` der Klasse `TVektor` in Java!

Hinweise:

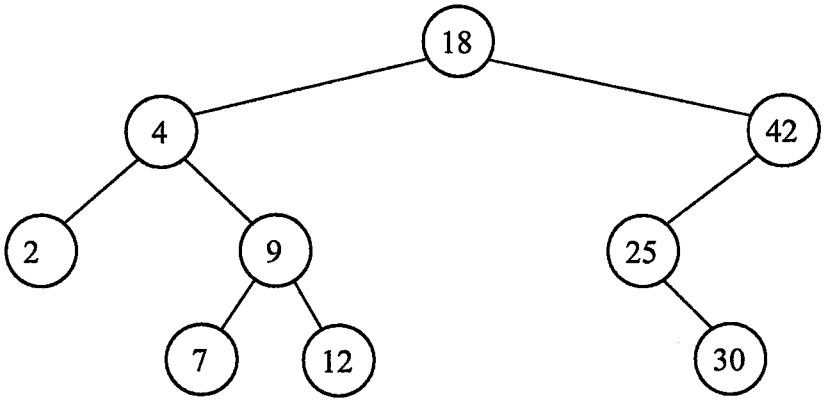
Sind $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ und $\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$ zwei Vektoren, so lässt sich ihr Skalarprodukt berechnen als

$x_1 \cdot y_1 + x_2 \cdot y_2 + x_3 \cdot y_3$.

Zwei Vektoren sind orthogonal, wenn ihr Skalarprodukt den Wert 0 hat.

1.2 Binäre Bäume

Gegeben ist folgender binärer Suchbaum:



1.2.1 Fügen Sie die folgenden Zahlen in dieser Reihenfolge in den Baum ein und zeichnen Sie den entstandenen Baum!

20 45 41 43

1.2.2 Entfernen Sie die folgenden Zahlen in der angegebenen Reihenfolge aus dem ursprünglichen Baum und zeichnen Sie jeweils den entstandenen Baum! Beschreiben Sie kurz, wie Sie beim Entfernen vorgehen!

2 4 18

1.2.3 Schreiben Sie die Elemente des ursprünglichen Baumes in Postorder- Reihenfolge auf!

1.2.4 Ein Binärbaum (kein Suchbaum) wurde in Preorder- und in Inorder- Reihenfolge ausgegeben. Dabei ergaben sich folgende Darstellungen

Preorder:	3	7	9	8	2	1	5	6
Inorder:	9	7	2	8	3	5	6	1

Zeichnen Sie den Baum!

1.2.5 Ein Binärbaum hat 300 Knoten. Geben Sie die minimale und maximale Höhe des Baumes an!

Aufgabe 2:

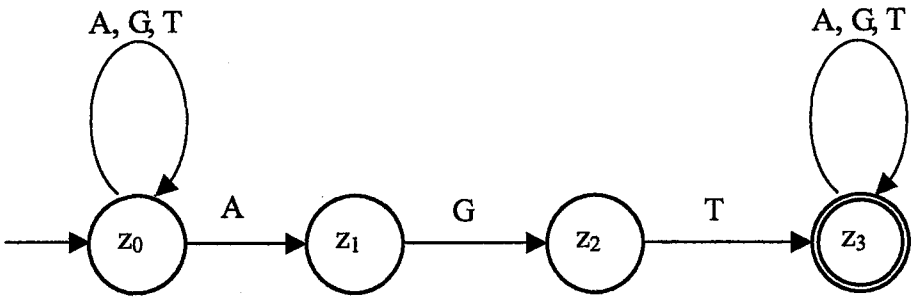
Automaten und formale Sprachen

In einem Science-Fiction-Roman wird der Planet Aurelia beschrieben, der um einen roten Riesen kreist. Die Aurelianer besitzen – ähnlich wie die Menschen – eine DNA, die den Aufbau der Proteine kodiert. Im Wesentlichen ist eine DNA-Sequenz eines Aurelianers ein aus den Grundbausteinen A, G und T zusammengesetztes Wort. Die Umweltbedingungen auf Aurelia haben zur Konsequenz, dass sich die Grundbausteine zunächst zu den Wörtern AGT, AGAA, GAAT und GAA verbinden. Die Proteine werden durch Konkatenation beliebig vieler dieser Wörter beschrieben.

2.1.1 Geben Sie begründet an, ob die Wörter GAATTTAG sowie GAATAGTGAA korrekte Beschreibungen aurelianischer Proteine sind.

2.1.2 Geben Sie einen endlichen Automaten an, der neben dem leeren Wort genau die Wörter akzeptiert, die eine korrekte Beschreibung aurelianischer Proteine darstellen.

2.2 Gegeben ist das folgende Übergangsdiagramm des nichtdeterministischen endlichen Automaten, der alle Wörter über dem Alphabet {A, G, T} akzeptiert, die die Zeichenkette „AGT" enthalten.



2.2.1 Geben Sie einen regulären Ausdruck an, der die Wörter erzeugt, die obiger Automat akzeptiert.

2.2.2 Erzeugen Sie mit der Teilmengenkonstruktion einen deterministischen Automaten, der die gleiche Sprache wie der abgebildete Automat akzeptiert und geben Sie alle Komponenten des neuen Automaten an! Geben Sie die Übergangsfunktion in tabellarischer Form an! Zeichnen Sie den zugehörigen Übergangsgraph!

2.3 Von besonderer Bedeutung sind die DNA-Sequenzen, die gleich oft die Wörter AGT und GAA enthalten. Geben Sie alle Komponenten einer Grammatik an, die solche besonderen DNA-Sequenzen erzeugt und geben Sie begründet an, ob es sich dabei um eine reguläre Sprache handelt.

Aufgabe 3:

3.1 Der Didaktische Computer (DC)

Hinweis: In der Anlage finden Sie den Befehlssatz des DC.

3.1.1 Betrachten Sie folgendes DC-Programm:

```
0     JMP  5
1     DEF  0
2     DEF  7
3     DEF  1
4     DEF  0
5     INM  1
6     LDA  1
7     SUB  2
8     STA  1
9     JPL  6
10    JZE  13
11    OUT  3
12    JMP  14
13    OUT  4
14    END
```

Bestimmen Sie die Ausgabe des Programms bei den Eingaben 28 und 19! Erklären Sie zusätzlich die Funktionalität dieses Programms und dabei auch die Bedeutung der möglichen Ausgabewerte!

3.1.2 Schreiben Sie ein DC-Programm zur Bestimmung der Summe der ersten n natürlichen Zahlen. Der Benutzer gibt die Zahl n ein, das Programm gibt die Summe aus.

3.2 Kryptografie

3.2.1 In einer Gruppe von 10 Personen soll es mit Hilfe eines Verschlüsselungssystems möglich sein, dass je zwei Personen miteinander kommunizieren können, ohne dass die anderen Personen die Nachrichten entschlüsseln können. Wie viele Schlüssel sind hierzu notwendig, wenn man

a) symmetrische Verschlüsselung verwendet?

b) asymmetrische Verschlüsselung verwendet?

3.2.2 Eine Verschlüsselung durch Substitution, bei der jeder Buchstabe durch genau einen anderen Buchstaben ersetzt wird, kann durch eine Häufigkeitsanalyse leicht geknackt werden. Folgende Idee soll die Häufigkeitsanalyse verhindern: Nach der Substitution wird zwischen je zwei Zeichen des Geheimtextes ein weiteres Zeichen so eingefügt, dass schließlich die Häufigkeitsverteilung der Buchstaben mit derjenigen eines unverschlüsselten Textes ungefähr übereinstimmt. Beim Entschlüsseln wird zuerst jedes zweite Zeichen gestrichen und danach werden die verbleibenden Zeichen wie gewohnt entschlüsselt.

Entscheiden Sie im Hinblick auf das Prinzip von Kerckhoff, ob dieser Vorschlag das Verfahren sicherer gemacht hat!

3.2.3 Gegeben ist das RSA-Verfahren mit den Primzahlen $p = 31$ und $q = 53$.

a) Berechnen Sie für den öffentlichen Schlüssel ($e = 29$, $n = p \cdot q$) den privaten Schlüssel (d, n).

b) Beschreiben Sie wie eine Nachricht m verschlüsselt wird und die verschlüsselte Nachricht c vom Empfänger decodiert wird. Gehen Sie dabei auf die Bedeutung der jeweiligen Schlüssel ein.

c) Worauf beruht die Sicherheit des RSA-Verfahrens?

Anlage: Befehlssatz des DC

Grundbefehle

Mnemo	Binär	Bedeutung
LDA	000000	LOAD INTO ACCUMULATOR—Lade den Wert der angegebenen Speicherstelle in den Akkumulator.
STA	000001	STORE ACCUMULATOR TO MEMORY — Speichere den Inhalt des Akkumulators an der angegebenen Speicherstelle ab.
ADD	000010	ADD TO ACCUMULATOR — Addiere den Wert der angegebenen Speicherstelle zum Inhalt des Akkumulators.
SUB	000011	SUBTRACT FROM ACCUMULATOR—Subtrahiere den Wert der angegebenen Speicherstelle vom Inhalt des Akkumulators.
NEG	010001	NEGATE ACCUMULATOR— Negiere den Inhalt des Akkumulators.
INC	010010	INCREMENT ACCUMULATOR — Erhöhe den Akkumulatorinhalt um 1.
DEC	010011	DECREMENT ACCUMULATOR— Erniedrige den Akkumulatorinhalt um 1.
OUT	001010	OUTPUT MEMORY — Gib den Wert der angegebenen Speicherstelle an die Output-Einheit.
INM	011100	INPUT TO MEMORY — Speichere die von der Input–Einheit gelesene Zahl an der angegebenen Adresse ab.
END	001011	ENDE— Programm beenden.
DEF	--------	DEFINE WORD — Beispiel: Mit 34 DEF 3 erhält die Speicherstelle mit der Adresse 34 den Wert 3 zugewiesen.

Sprungbefehle

Mnemo	Binär	Bedeutung
JMP	000100	JUMP — Unbedingter Sprung. Springe zur angegebenen Speicherstelle und fahre mit dem dort stehenden Befehl fort.
JMS	000101	JUMP IF MINUS — Sprung, wenn Akkumulatorinhalt < 0.
JPL	001000	JUMP IF PLUS — Sprung, wenn Akkumulatorinhalt > 0.
JZE	001001	JUMP IF ZERO — Sprung, wenn Akkumulatorinhalt $= 0$.
JNM	011010	JUMP IF NOT MINUS — Sprung, wenn Akkumulatorinhalt ≥ 0.
JNP	011011	JUMP IF NOT PLUS — Sprung, wenn Akkumulatorinhalt ≤ 0.
JNZ	010100	JUMP IF NOT ZERO — Sprung, wenn Akkumulatorinhalt $\neq 0$.