Prüfungsprotokoll zur Theoretische Informatik

Fächer:	Eff. Algorithmen, Kryptographie und Angewandte Automatentheorie
Prüfer:	Prof. Hromkovic, Dr. Unger
Datum:	07.05.2004
Dauer:	45 Min
Note:	1.3

Effiziente Algorithmen - (Buch: Hromkovic)

Hromkovic: Was ist dynamische Programmierung und welche Algorithmen haben Sie dazu kennengelernt?
Ich: Dynamisches Programmieren ist eine Design-Technik, bei der man mit der Lösung der kleinsten Teilprobleme beginnt und diese dann zur Gesamtlösung zusammensetzt. Es handelt sich hierbei um einen Bottom-Up Ansatz. Als Algorithmen kenne ich den Floyd-Algorithmus und das DPKP als pseudopolynomialzeit Algorithmus.

Hromkovic: Was sind die Unterschiede zu Divide-and-Conquer?
Ich: Der Unterschied liegt darin begründet, dass Divide-and-Conquer ein Top-Down Ansatz ist. Hierbei wird das Problem in Teilprobleme geteilt und diese wiederum geteilt bis das Problem leicht zu lösen ist. Divide-and-Conquer ruft sich dabei rekursiv auf den Subproblemen auf.

Hromkovic: Kennen Sie ein Beispiel wo dynamische Programmierung besser funktioniert als Divide-and-Conquer?
Ich: Ja, Fibunacci (Erklärt)

Hromkovic: Sie redeten eben über den Floyd-Algorithmus, können Sie diesen Erklären?
Ich: Der Floyd Algorithmus dient zur Berechnung der kürzesten Pfade zwischen je zwei Knoten v_i und v_j. Dazu berechnet er für $k = 0, ..., n$ jeweils $cost_k(i, j)$, was den Kosten des kürzesten Pfades zwischen v_i und v_j entspricht, wobei dieser über die internen Knoten $(v_1, ..., v_k)$ geht.

Hromkovic: Erklären Sie die Rekursion.
Ich: Formel aufgeschrieben - hat gereicht.

Hromkovic: Sie sprachen auch über Pseudo-Polynomialzeit - Was ist das?
Ich: Zunächst erkläre ich einmal, was ein Integer-Valued problem ist:
Ein Integer-Valued Problem ist ein Problem, dessen Eingabe als Menge von Ganzzahlen gesehen werden kann. Zum Beispiel TSP, MST, VC, ...

Hromkovic: Anstatt einer Menge würde ich hier eher den Begriff des Vektors verwenden.
Ich: Ja, genau :-)
Ich: Also dann ist ein pseudopolynomialzeit-Algorithmus wie folgt definiert:
Ein Algorithmus A für ein Integer-Valued Problem U heißt Pseudo-Polynomiell, wenn

- er U löst

- für alle Instanzen $x \in U$ ein Polynom p existiert, so dass $Time(x) \leq O(p(|x|, Max-Int(x)))$

Danach noch kurz erläutert, was $Max - Int(x)$ bedeutet und diese Definition aussagt.

Hromkovic: Können Sie ein Beispiel nennen?
Ich: Ja! DPKP, also den Dynamic-Programming Knapsack Algorithmus.

Hromkovic: Erklären Sie diesen bitte!
Ich: Der DPKP Algorithmus erhält als Eingabe eine Instanz des Knapsack Problems, also $I = (c_1, ..., c_n, w_1, ..., w_n, b)$. Er arbeitet mit Tripeln der Form: (k, w, T) mit:

- $k :=$ Summe der Kosten c_i
- $w :=$ Summe der Gewichte w_i
- $T :=$ Menge der Indizes über die summiert wird

Hromkovic: Ja, die ganze Konstruktion können wir uns wohl sparen, erklären Sie bitte, warum dieser Algorithmus effizienter ist, als ein naive Ansatz.
Ich: Das liegt darin begründet das man im zweiten Schritt des Algorithmus aus der Hilfsmenge $SET(i+1)$ nur die Lösungen in $TRIPLE(i+1)$ übernimmt, die bei gleichen Kosten das minimale Gewicht haben sprich für k Triple mit gleichen Kosten aus $SET(i+1)$ nimmt man immer nur das Tripel mit dem kleinsten Gewicht in die Menge $TRIPLE(i+1)$.

Hromkovic: Sie haben auch Probleme kennengelernt, für die keine pseudopolynomialzeit Algorithmen existieren. Was können Sie mir dazu sagen?
Ich: Wenn ein Problem U stark NP-Schwer ist, dann existiert kein pseudopolynomialzeit Algorithmus für U.

Hromkovic: Was bedeutet das: 'stark NP-Schwer'?
Ich: (Da musste ich ziemlich im Kopf kramen - aber letztendlich habe ich mich doch erinnert) Ein Problem U ist stark NP-Schwer wenn ein Polynom p existiert, so dass das Problem value(p)-U NP-Schwer ist.

Hromkovic: Und wie ist dieses Value(p)-U definiert?
Ich: (Hier hat er gemerkt, das ich Lücken hatte und so musste ich wieder kramen) Also ein p-value-bounded subproblem von U ist ein Problem welches durch einschränken der Eingabeinstanzen x von U durch $Max - Int(x) \leq h(|x|)$ entsteht. (Meine Definition war in der Prüfung nicht ganz so sauber).

Hromkovic: Können Sie auch zeigen, dass es keinen pseudopolynomialzeit Algorithmus für TSP gibt?
Ich: (Und Jackpot - diese Reduktion habe ich mindestens 10x gelesen, aber mehr oder weniger 'überlesen' - aber na gut versuchen wir es mal) Also dafür muss ich folgende Reduktion machen: $HC \leq_p Lang_{Value(p)-TSP}$.

Hromkovic: Ja, bitte machen Sie diese Reduktion mal.
Ich: Habe die Reduktion gemacht, aber weil ich nicht mehr wusste, wie $Lang_{Value(p)-TSP}$ definiert war, habe ich ewig gebraucht (und auch nur nach etlichen Hilfen von Prof. Hromkovic) um zu beweisen, dass der Graph G einen Hamiltonkreis enthält, wenn der $((K_n, c), n \in Lang_{Value(p)-TSP}$.

Hromkovic: Kennen Sie PTAS?
Ich: Ein Approximationsalgorithmus A ist ein PTAS (polynomial time approximation scheme) für U, wenn A eine Lösung für U mit einem relativen Fehler von höchstens ε liefert und dabei $Time_A(x, \varepsilon^{-1})$ in einem Polynom über $|x|$ beschränkt ist.

Hromkovic: Kennen Sie auch ein Beispiel?
Ich: Ja, PTAS für SKP. Dann musste ich diesen Aufschreiben und die Güte beweisen. Am Ende des Beweises bin ich abgebrochen worden.

Kryptographie - (Buch: Delfs und Knebel)

Unger: Was ist Public Key Kryptograpie, was ist klassische Kryptographie?
Ich: Bei der klassischen Kryptographie besitzen Sender und Empfänger den selben Schlüssel. Dieser ist geheim zu halten, sowie initial geheim zu übertragen. Kennt jemand diesen Schlüssel, so kann er alle Kryptogramme entschlüsseln. Bei der Public-Key Kryptographie hingegen gibt es je einen Public- und einen Secret-Key. Der Public-Key wird zur Verschlüsselung genutzt, der Secret Key zur Entschlüsselung. Es muss initial kein Gehimnis ausgetauscht werden. (Ungefähr so erklärt)

Unger: Was sind Vor- bzw. Nachteile von PK-Verfahren?
Ich: Vorteil von oben (keine Geheime Übertragung des Schlüssels notwendig) erläutert, dann noch erklärt, dass Klassische Verfahren meist schneller und einfacher zu implementieren sind und in der Regel Schlüsselaustausch per PK-Verfahren realisiert wird und dann Übertragung durch klassische Verfahren gesichert wird.

Unger: Aber da gibt es noch Vorteile der PK-Verfahren!
Ich: Mhh... (Nach einiger Zeit dann:) Achja, Signatur bzw. die Möglichkeit interaktive Beweissysteme o.ä.

Unger: Ja genau, dazu noch etwas erläutert, weil ich mir nicht ganz sicher war, ob man nicht auch mit klassischer Verschlüsselung Signieren kann. (Das ist laut Dr. Unger übrigens nicht so, sprich man braucht ja ein Geheimnis für Signaturen, etc.). Und noch ein Vorteil von PK-Verfahren. Denk mal an die Schlüssel!
Ich: (Nach einiger Zeit auch darauf gekommen). Ach ja, die Anzahl der zu übertragenden Schlüssel ist kleiner als bei der klassischen Verschlüsselung. (Endlich :-))

Unger: Ja, wie viele Schlüssel müssen denn bei einem klassischen verfahren übertragen werden, wenn k-Personen miteinander kommunizieren möchten?
Ich: Also auch da bin ich nach etwas 'Starthilfe' drauf gekommen. Habe das mit einem vollständigen Graphen mit k Knoten modelliert und dann auch irgendwann verstanden, dass eine Kante einen zu übertragenden Schlüssel repräsentiert. Also $\frac{k \cdot (k-1)}{2}$ Schlüssel.

Unger: Welche Public-Key-Verfahren kennen Sie?
Ich: RSA, El-Gamal, Rabin

Unger: Erklären Sie Rabin
Ich: Rabin ist ein Public-Key Verfahren, dass auf dem mathematischen Problem des modularen quadrierens basiert. Im Gegensatz zu El-Gamal und RSA ist beweisbar so schwer wie die Faktorisierung großer Zahlen.

Schlüsselgenerierung:

1. Wähle zwei große Primzahlen $p,\ q$ mit $n = pq$ (Gut für die Entschlüsselung: $p, q \equiv 3\ mod\ 4$)
2. Public-Key: (n) Private-Key: (p, q)

Verschlüsselung:

1. $E : x \to x^2$

Enschlüsselung:

1. Zur Bestimmung von $x \equiv \sqrt{c}\ mod\ n$ berechne:
2. $a \equiv c\ mod\ p$
   $b \equiv c\ mod\ q$
3. Ziehe dann die Wurzeln von a, b in Z_p, Z_q
   $x \equiv \sqrt{a}\ mod\ p$
   $x \equiv \sqrt{b}\ mod\ q$
4. Erhalte so bis zu 4 Lösungen, und wähle (durch Zusatzinformation, oder durch raten) die richtige Lösung. Mit dem Chinesischen Restsatz folgt dann $x \equiv \sqrt{c}\ mod\ n$

Unger: Wie sicher ist Rabin, wie sicher sind die anderen?
Ich: Rabin ist beweisbar so schwer wie die Faktorisierung großer Zahlen. Bei RSA und El-Gamal glaubt man zwar, dass auch diese Verfahren so schwer sind wie die Faktorisierung, man kann es jedoch nicht beweisen.

Unger: Können Sie beweisen, dass Rabin so schwer ist wie die Faktorisierung?
Ich: Mhhh - nein! (Dr. Unger hat mir dann erzählt, dass dieser Beweis nur ein Einzeiler ist und irgendwo da steht. Wie ich nach der Prüfung herausgefunden habe, steht dieser Beweis im Anhang - naja)

Unger: Gut, ist nicht so schlimm! Kennen Sie Merkle's Meta Methode?
Ich: Ja, merkle's Meta Methode ist ein Verfahren um aus einer kollisionsfreien Kompressionsfunktion eine kollisionsfreie Hashfunktion zu berechnen. Verfahren komplett erklärt.

Unger: Können Sie auch zeigen, dass h kompressionsfrei wenn f kompressionsfrei?
Ich: Leider wieder nein! Ich habe zweimal angesetzt, konnte allerdings beide Ansätze nicht zu ende führen.

Unger: Ok, kommen wir zu Zero-Knowledge, bitte definieren Sie mal Zero-Knowledge
Ich: Ein interaktives Beweissystem zwischen Prover und Verifier (P, V) ist Zero-Knowledge, wenn ein probabilistischer Simulator existiert $S(V^{\cdot}, x)$, der für einen beliebigen Verifier $V^{\cdot}$ ein akzeptierendes Transkript $tr_{P,V^{\cdot}}(x)$ in erwarteter Polynomzeit berechnet. Dieses Transkript darf in seiner Verteilung nicht von einem normalen zu unterscheiden sein.

Unger: Welche Zero-Knowledge verfahren kennen Sie?
Ich: Fiat-Shamir

Unger: Erklären Sie simplified Fiat-Shamir
Ich:

- $n = pq$, $QR_n :=$ Menge der Quadrate in Z_n

- $x \in QR_n$ und $y^2 = x$. y ist Peggy's Geheimnis.

Das Protokoll durchläuft folgende Schritte:

1. Peggy wählt eine Zahl $r \in Z_p^{\cdot}$ zufällig und sendet $a = r^2$ an Vic
2. Vic wählt ein $e \in \{0, 1\}$ zufällig und sendet dieses an Peggy.
3. Peggy berechnet $b = r \cdot y^e$ und sendet b an Vic
4. Vic überprüft, ob $b^2 = a \cdot x^e = r^2 \cdot (y^2)^e = r^2 \cdot (y^e)^2 = (r \cdot y^e)^2 = (b)^2$

Unger: Und wie zeigt man, dass dieses Verfahren Zero-Knowledge ist?
Ich: Dieses Protokoll ist Zero-Knowledge, da ein probabilistischer Simulator wie folgt zu erstellen ist:
Der Simulator wählt zunächst ein $\widetilde{e}$ und schickt $a = r^2 \cdot x^{\widetilde{e}}$ an Vic. Vic, egal ob honest oder nicht sendet ein e an den Simulator. Wenn $e = \widetilde{e}$ sendet der Simulator $b = r$ notiert der Simulator den Schritt (a,e,b). Das Transkript kann dabei in Polynomzeit erstellt werden, weil der Simulator mit der Wahrscheinlichkeit $p = 0,5$ das richtige e rät.

Unger: Ok, wir hatten auch ein Verfahren, dass wie das beschriebene arbeitet, jedoch nicht interaktiv ist. Können Sie mir dazu etwas sagen?
Ich: Ja, Sie sprechen sicher auf das Fiat-Shamir signature scheme an.

Unger: Genau, erklären Sie mal grob wie das funktioniert.
Ich: Ich habe ihm erklärt, das man sich die Challenge quasi selbst stellt und dann so die Signatur erstellt.

Unger: Und warum kann ich nicht betrügen, wenn ich mir die Challenge selbst stelle?
Ich: Mhhh, da stand ich wieder mal auf dem Schlauch. Ich hatte dieses Verfahren nur marginal gelernt, weil ich wieder mal nicht erwartet hatte. Trotz Tipps, dass es etwas mit Hash-Funktionen zutun hat, bin ich nicht drauf gekommen. (Es ist übrigens sicher, weil man das Challenge nicht zufällig wählen kann, sondern dies ein Hash der Nachricht ist.)

Angewandte Automatentheorie - Vorlesung SS 2003

Hromkovic: Sie kennen die Minimierung von endlichen Automaten, was können Sie mir dazu sagen?
Ich: Ich habe erklärt, dass die Minimierung von DEA's auf einer Zusammenfassung von Zuständen bezüglich einer bestimmten Äquivalenz, der sogen. Nerode Äquivalenz entspricht.

Hromkovic: Erklären Sie mal die Nerode Äquivalenz
Ich: Die Nerode Kongruenz ($\sim_L$) ist wie folgt definiert: $u \sim_L v \Leftrightarrow \forall w \in \Sigma^{\cdot} gilt : uw \in L \Leftrightarrow vw \in L$

Hromkovic: Un wie verwendet man diese um einen DEA zu minimieren?
Ich: Die Idee bei der DEA Minimierung ist es Wörter zu finden, die nicht-Äquivalente Zustände trennen. Dabei geht man über die Länge der Wörter und beginnt mit dem leeren Wort. Zunächst teilt man die Zustandsmenge in zwei Blöcke,

nämlich F und $Q \backslash F$, da diese durch das leere Wort getrennt werden un daher nicht äquivalent sind. Diese Blockeinteilung wird dann schrittweise durch den sogenannten Blockverfeinerungs-Algorithmus verfeinert. Dieser spaltet in einem Schritt einen Block B_i bezüglich B_j und a auf wenn gilt: $p, q \in B_i$ und $\delta(p, a) \in B_j$ aber $\delta(q, a) \notin B_j$. Der Algorithmus verfeinert die Blöcke so lange, bis keine Aufteilung mehr möglich ist.

Hromkovic: Wie ist das bei der NEA Minimierung?
Ich: Hier ist die Berechnung von nicht-Äquivalenten Zuständen nicht ganz so einfach, daher gibt es zwei verfahren. Zum einen die suboptimale Minimierung über die feinere Bisimulationsäquivalenz, zum anderen den Ansatz über den sogenannten Universal NEA.

Hromkovic: Ist die Minimierung über den Universal NEA den optimal?
Ich: Ja!

Hromkovic: Das stimmt nicht so genau.
Ich: Mhh, ich habe es so verstanden, dass der Universal NEA alle minimalen NEA's, also die optimalen Lösungen als Teilautomaten induziert. Das Problem ist es eben einen minimalen zu finden - wahrscheinlich wird deshalb hier ggf. eine suboptimale Lösung ausgegeben.

Hromkovic: Ja genau! Wie schwer ist das finden von Äquivalenten zuständen bei der NEA Minimierung?
Ich: PSPACE Schwer (Nein es ist raus, jetzt muss er eigentlich nach der Reduktion fragen..)

Hromkovic: Können Sie das auch beweisen?
Ich: Ja! Ich habe die Reduktion (Nicht Universalitätsproblem für NEA's ist PSPACE schwer) erklärt. War mir nicht ganz sicher, ob alles wirklich korrekt war, aber hat offensichtlich alles gestimmt. Dann habe ich noch erklärt, das das nicht-Universalitätsproblem das spezielle Äquivalenz und das spezielle Minimierungsproblem sind.

Hromkovic: Dankeschön, bitte gehen Sie kurz raus.

Fazit

Herr Prof. Hromkovic ist ein sehr guter und fairer Prüfer, was man sicher auch an der Note sieht. Er hat es so begründet, dass offensichtlich ist, das ich sehr gut gelernt habe, ich jedoch Lücken hatte, die mir die 1.0 verbaut haben (S. Beweis Rabin, Beweis Merkles-Meta-Methode, ...). Zum Anfang meiner Prüfung wurde ich gefragt, ob ich Dr. Unger als Prüfer für Krypto akzeptiere. Dr. Unger hat vergleichsweise hohe Erwartungen in Krypto und fragt wenig Standard. Darüber hinaus versucht er krampfhaft witzig zu sein, was in den meisten Fällen daneben geht, so sind Spürche wie: 'Freiversuchler lassen wir doch immer durchfallen' vor einer Prüfung nicht unbedingt motivierend - naja also mein Tipp wenn es zu vermeiden ist nehmt Dr. Unger nicht als Co-Prüfer. Die Fächer sind alle überschaubar, mein Tipp: lernt wirklich alles und verlasst euch nicht ausschließlich auf das was in vorherigen Prüfungen gefragt wurde. Dinge wie FPTAS, Wahlen, ... sind anstrengend, aber werden ggf. gefragt. Allen die die Prüfung noch vor sich haben, viel Glück!