Protokoll Diplomprüfung
Theoretische Informatik

Martin Jansen
<martin.jansen@rwth-aachen.de>

16.04.2007

- Fächer:
  1. Effiziente Algorithmen (Vorlesung Prof. Vöcking, WS 04/05)
  2. Compilerbau (Prof. Indermark, SS 05)
  3. Algorithmische Kryptographie (Dr. Unger, WS 06/07)
- Dauer der Prüfung: Ich habe vergessen, zu Beginn auf die Uhr zu sehen, aber die Prüfung war um 13:35 Uhr beendet. Da wir irgendwann nach 12:50 Uhr angefangen haben, dürfte die Dauer damit ca. 40 Minuten betragen haben.
- Note: 1,3

Vorbereitet auf die Prüfung habe ich mich mit dem Skript von Prof. Vöcking für Effiziente Algorithmen, der studentischen Mitschrift für Compilerbau und den Folien von Dr. Unger für Algorithmische Kryptographie. In Compilerbau haben wir zusätzlich im Sommersemester 2005, als Prof. Indermark zum letzten Mal Compilerbau gelesen hat, die Übungen bearbeitet.

Die Atmosphäre in der Prüfung war locker und die beiden Prüfer waren sehr freundlich. Ich kann den i1 für Prüfung rundum empfehlen.

1 Effiziente Algorithmen

V(öcking): Wo mit möchtest du anfangen?
I(ch): Flüsse wären gut.

V: Gut, dann erkläre doch mal, was Flussnetzwerke sind.
I: Ich habe dann so ziemlich alles erzählt, was ich zu Flussnetzwerken wusste. (Gerichteter Graph, Quelle, Senke, Kapazitätsfunktion, Flussfunktion, Kapazitätsbeschränkung, Flusserhaltung, Suche nach dem maximalen Fluss, ...)

V: Erkläre doch mal das erste Verfahren, das wir zur Suche nach einem maximalen Fluss betrachtet haben.
I: Ford-Fulkerson erläutert, Restnetzwerk definiert. Bei der Definition der rest_f-Funktion hat Prof. Vöcking dann nachgehakt und wollte recht detailiert wissen, warum denn nun auf den entgegengesetzten Kanten auch eine Restkapazität anliegt. Ich habe dann ein wenig rumgesülzt, bin aber im Endeffekt nicht drauf gekommen. Er wollte wohl hören, dass das für den Korrektheitsbeweis so sein muss ...

V: Wie war das denn mit der Laufzeit bei Ford-Fulkerson?
I: Ich habe dann beide Laufzeitabschätzungen angegeben. Bei der pseudopolynomiellen habe ich kurz erklärt, wie die obere Schranke für den maximalen Fluss zustande kommt. Bei $O(n^5) = O(m^2 \cdot n)$-Laufzeit musste ich dann detailiert beweisen, warum sie gilt. Ein m wegen Breitensuche, maximal $2m$ Kanten im Restnetzwerk, jede Kante kann höchstens $\frac{n}{2}$ mal entfernt werden. Bei letzterem Punkt wollte Prof. Vöcking es dann auch wieder recht genau wissen.

V: Und wie geht die Suche nach einem max. Fluss noch besser?
I: Dinic und Forward-Backward-Propagation. Idee der Suche nach mehrerer kürzesten fv-Wegen erläutert und Laufzeiten genannt.

V: Springen wir mal zu Approximationsalgorithmen. Wir hatten da ja den Begriff des Approximationsfaktors. Kannst du den erklären?
I: Ich habe den Faktor dann nach Def. aus dem Skript erklärt, habe aber wohl vergessen zu erwähnen, dass der Approximationsfaktor für *alle* Eingaben I gelten muss. Prof. Vöcking war das sehr wichtig und er hat daher nachgehakt. Ich bin aber dummerweise nicht drauf gekommen, was er hören wollte.

V: Erkläre doch mal die Approximation für Min-VC.
I: Gesagt, dass man ein inklusionsmaximales Matching bestimmt und die Endpunkte der Kanten dieses Matchings als Vertex-Cover ausgibt. Dann habe ich den Approximationsfaktor kurz bewiesen und erzählt, dass es sich beim Ergebnis tatsächlich um ein Vertex-Cover handelt.

V: Wie ermitteln wir denn so ein inklusionsmaximales Matching?
I: Greedy-Algorithmus in $O(m)$ mit m = Anzahl der Kanten.

V: Wir hatten da auch noch sowas mit Hypergraphen. Was waren denn nochmal Hypergraphen?
I: Graph mit Kanten, die mehr als zwei Knoten verbinden.

V: Genau. Und dann haben wir uns das Set-Cover-Problem angeschaut. Erzähl doch mal was dazu.
I: Na toll, da lerne ich ein Thema nicht und genau dazu fragt er was. Ich habe dann gesagt, dass ich passen muss, weil ich keinen Plan von Set-Cover habe.

V: Ok, das kannst du nicht. Nicht so schlimm. Wir bleiben aber noch ein wenig bei Approximationsalgorithmen. Wir haben uns auch mit Scheduling-Problemen befasst. Erklär doch mal.
I: Scheduling erklärt, Makespan beschrieben, erzählt, dass wir Scheduling auf identischen und allgemeinen Maschinen betrachtet haben.

V: Richtig. Wir hatten bei identischen Maschinen ja zwei recht einfach Algorithmen. Weisst du, welche ich meine?
I: Er meinte natürlich LL und LPT. Ich habe dann beide erklärt und musste jeweils zeigen, wie die Approximationsfaktoren zustande kommen. Bei LPT bin ich zum Schluss etwas ins Straucheln gekommen, ging aber gerade noch (mit etwas Hilfe von Prof. Vöcking) halbwegs gut.

V: Gut, machen wir zum Schluss noch ein bisschen Online-Algorithmen. Wir hatten da das Thema Paging. Was macht man denn da so?
I: Paging erläutert ...

V: Wie war das denn da mit der Competitivity für deterministische Algo-

rithmen?
I: Ich habe dann erläutert, dass kein det. Algorithmus besser als k-competitive ist und dass wir gezeigt haben, dass det. Markierungsalgorithmen k-competitive sind. In dem Zusammenhang habe ich dann auch Markierungsalgorithmen erklärt.

V: Wie war das denn, wenn man einen randomisierten Markierungsalgorithmus verwendet?
I: MARK erläutert; gesagt, dass man für die Competitivity den Beweis in zwei Teilen ($\frac{f}{2}$ viele Fehler für OPT, $2 \cdot H_k$ viele für MARK) durchführt.

V: Zeige mir doch mal den ersten Teil.
I: Beweis wie im Skript gezeigt.

2 Compilerbau

V: Welche Phasen haben wir denn so im Compilerbau?
I: Phasen runtergebetet.

V: Erzähl doch mal, was du über die lexikalische Phase weisst.
I: Ich habe dann ziemlich lange am Stück geredet und habe dabei IIRC auch so ziemlich alles relevante über die lexikalische Phase erzählt. (Einfaches und erweitertes Matching-Problem, Thompson-Konstruktion, Potenzmengen-Konstruktion, Produktautomat, Backtrack-DFA, flm-Zerlegung, ...).

V: Wie ist denn die Laufzeit der lexikalischen Analyse?
I: Habe erklärt, dass sie im Worst Case quadratisch in der Länge der Eingabe ist, dass man in der Praxis aber üblicherweise linearen Aufwand hat.

V: Genau. So beknackte Bezeichner verwendet eigentlich niemand, dass man da quadratischen Aufwand hat. Kannst du mir ein Beispiel aus einer Programmiersprache nennen, wo man die Anwendung der flm-Zerlegung sieht?
I: Konnte ich auf Anhieb nicht und habe daher ein wenig rumgelabert. Im Endeffekt ist Prof. Vöcking mir die richtige Antwort auch schuldig geblieben `;-)`.

V: Beschreibe doch mal kurz, was in der Syntaxanalyse gemacht wird.

I: Nachdem die Begriffe Ableitungsbaum, TD- und BU-Analyse gefallen waren, unterbrach Prof. Vöcking mich und meinte, dass der rest wohl klar sei.

V: Wie heissen eigentlich die speziellen Grammatiken, die man in der Syntaxanalyse verwendet?
I: Er wollte nur LR(k) und LL(k) hören. Die Definitionen haben ihn nicht interessiert.

V: Bei der semantischen Analyse verwenden wir Attribute. Welche zwei Typen kennst du und was kann man damit machen?
I: Synthetische und inherite Attribute erklärt und das Standardbeispiel der Deklariertheit von Bezeichnern gebracht.

3 Algorithmische Kryptographie

U(nger): Wir hatten da ein PK-Verfahren, wo der Krypttext doppelt so groß ist. Welches war das denn?
I: ElGamal erklärt, keine Hintertür durch das System sondern durch die Person, die verschlüsselt. Aufbau erklärt.

U: Bei welchem Verfahren war das denn noch so?
I: Hmm, bei homomorpher Verschlüsselung hatten wir auch Tupel für den Krypttext.

U: Ja, aber ich denke an ein anderes Verfahren.
I: Ach ja, elliptische Kurven.

U: Genau. Kommen wir mal kurz zu RSA. Wie war das denn da mit der Bitsicherheit?
I: Ich habe dann die beiden Angriffsszenarien mit Hilfe der Orakel erläutert und gesagt, dass RSA damit so sich wie die Teilinformationen ist. Beim ersten Orakelangriff habe ich dann auch kurz erklärt, dass man die binäre Suche durch Multiplikation mit 2^{x^e} löst.

U: Springen wir mal zu Zero-Knowledge-Proofs. Welche kennst du?
I: Alle aufgelistet, die wir in der Vorlesung behandelt haben. Zusätzlich habe dann auch noch gesagt, dass es für alle Probleme aus NPC ZKPs gibt.

U: *Dr. Unger hat dann ein "Taubenschießen-Problem" skizziert, bei dem man (wer hätte es gedacht) mit einem Gewehr auf Tauben schießen kann. Mit einem Schuß trifft man entweder keine Taube, eine Taube oder genau zwei Tauben. Ich sollte mir dann überlegen, wie ein ZKP aussieht, mit dem man zeigen kann, dass man alle Tauben am Himmel mit k Schüssen treffen kann.*
I: Ich war im ersten Moment ziemlich geschockt und wusste nicht wirklich, was ich machen sollte. Nach ein wenig Nachdenken habe ich dann gesagt, dass man versuchen könnte, das Problem auf ein NP-vollständiges Graph-Problem zu reduzieren. Dr. Unger meinte, dass das genau der richtige Weg sei und ermutigte mich, weiter "laut zu denken". Ich habe dann noch ein bisschen erzählt und dieses und jenes gesagt, bin aber im Endeffekt nicht wirklich weiter gekommen. Schließlich meinte Dr. Unger dann, dass man die Geschichte auf das Set-Cover-Problem reduzieren kann. Im Nachhinein ist das völlig offensichtlich, aber da ich Set-Cover bereits in Effiziente Algorithmen nicht gelernt hatte und dort schon auf die Nase gefallen war, bin ich natürlich in der Prüfung nicht auf den Trichter gekommen. War aber scheinbar nicht weiter tragisch, da wohl entscheidend war, dass ich das Prinzip der NP-Reduktion erkennen sollte.

U: Was haben wir denn mit ZKPs alles gemacht?
I: Identifikation (Shamir, Fiat-Shamir, Schnorr), Wahlen, elektronisches Geld.

U: Kannst du etwas mehr zur Anwendung bei elektronischem Geld erzählen?
I: Ich habe dann erläutert, dass die Bank und ihre Kunden bei der Auszahlung von Geld einen ZKP (Schnorr, Kenntnis eines disk. Logarithmus) durchführen und das Transskript dann als Basis für die Münze verwenden. Als ich dann die Modifikation der Münze durch den Kunden erläutern wollte, meinte Dr. Unger, dass ich das nicht erklären brauche.

U: *Schreibt die φ-Funktion auf ein Blatt.* Woher kennst du die und was macht man damit?
I: Wahlen, Mischer vermischen die Stimmen, Wähler kennen φ und $g^{\cdots}$ und raten solange eine Zahl, bis sie eine gefunden haben, die ihrem Kandidaten entspricht, usw. usw.

U: Ganz genau. Nehmen wir mal an, dass wir ein Wahlprotokoll haben

wollen, bei dem niemand weiss, wie die Wahl ausgegangen ist. Trotzdem soll der Sieger (und nur er) beweisen können, dass er gewonnen hat. Wie würdest du das machen?
I: Hmm, das steht definitiv nicht im Skript. Ich bin aber relativ schnell drauf gekommen, dass der Sieger der Wahl mit einem ZKP die Kenntnis eines Geheimnisses beweist, das er durch Lagrange-Interpolation aus einem (t, n)-Threshold-Schema ermittelt hat. Die für die Interpolation notwendigen Stützstellen erhält er von den Wählern. Wenn also der Sieger zum Beispiel 50% der Stimmen benötigt um zu gewinnen und es k Wähler gibt, wählt man sich ein Polynom vom Grad $\frac{k}{2} - 1$ und bastelt daraus dann das (t, n)-Schema.

V: Dann lass uns doch bitte mal kurz alleine.