Theoretische Informatik

Gedächtnisprotokoll zur mündlichen Diplomprüfung

7. Oktober 2009 (SS09, RWTH Aachen University)

Florian Weingarten

Inhalt:

- Effiziente Algorithmen (nach Vorlesung im SS 2008)
- Netzwerkalgorithmen (nach Skript vom SS 2009)
- Cryptography I+II (nach den Vorlesungen im WS 2007/08 und SS 2008)

Prüfer:

- Prof. Dr. Berthold Vöcking (Lehrstuhl für Informatik 1: Algorithmen und Komplexität)
- Prof. Dr. Rudolf Mathar (Lehrstuhl für Theoretische Informationstechnik)

Prüfungsdauer: 45 Minuten

Prüfungsnote: 1.0

Achtung: Hierbei handelt es sich *nicht* um ein offizielles Prüfungsprotokoll der RWTH sondern um ein privates Gedächtnisprotokoll, das ca. eine Stunde nach der Prüfung angefertigt wurde. Ich habe mit Sicherheit einige Sachen vergessen und gebe natürlich keine Garantie auf Korrektheit.

Effiziente Algorithmen

BV: Fangen mir mal mit Flussalgorithmen an. Wir hatten da ein Flussnetzwerk, das ist ein gerichteter Graph mit einer Quelle und einer Senke und Kantenkapazitäten aus den natürlichen Zahlen. Erzählen Sie mal was ein Fluss ist.
FW: Das ist eine Abbildung auf den Kanten des Netzwerkes in die nicht-negativen reellen Zahlen $f : E \to \mathbb{R}_0^+$, und da gibt es dann ein paar Einschränkungen: Die Flusserhaltung (ausser für Quelle und Senke) und die Kapazitätsbeschrenkung. So, man ist jetzt an dem größtmöglichen Fluss interessiert.
BV: Genau, da hatten wir den Ford Fulkerson Algorithmus. Erzählen Sie mal.
FW: Der sucht sogenannte flussvergrößernde Wege im Restnetzwerk. Das Restnetzwerk ist ein Netzwerk, das die gleichen Knoten wie der ursprüngliche Graph hat. Dann gibt es so eine Hilfsfunktion, $rest_f(u, v)$, die ist gleich $c(u, v) - f(u, v)$, falls die Kante vorher schon da war, $f(v, u)$, falls die entgegengesetzte Kante da war, und sonst 0. Im Restnetz sind jetzt nur die Kanten, deren Restwert positiv ist. Achja, wir hatten angenommen, dass es keine Hin- und Rückkanten gleichzeitig gibt, nur eine von beiden.
BV: Genau, um die Notation einfach zu halten.
FW: Ja, aber das ist eigentlich garkeine Einschränkung, denn ...
BV: Glaub ich Ihnen. Wie schnell geht das?
FW: In $O(mC)$, aber das ist ja nicht so toll. Da kann man ein einfaches Beispiel bauen, und zwar ...
BV: Glaube ich Ihnen. Wie gehts besser?
FW: Mit Breitensuche, dann kriegt man $O(n^5)$, bzw. genauer $O(m^2 n)$.
BV: Glauben Sie, dass Sie das beweisen können?
FW: Ja, krieg ich hin. Das liegt einfach daran, dass die Distanzen von der Quelle größer werden. Allerdings nur wenn man Breitensuche macht. Wenn man eine Flaschenhalskante hat, dann fliegt die raus und die Rückkante kommt rein. Die geht jetzt aber von einem Knoten mit höherer Distanz zurück zu einem mit niedrigerer, kann also nicht besser werden.
BV: Ganz genau. Ich male Ihnen dazu mal ein Bild.
Er zeichnet ein paar Knoten, dann einen Knoten u und einen Knoten v. Daran soll ich das erklären was ich gerade gesagt habe.
Machen Sie mal weiter.
FW: Ja, also, man sieht, dass die Distanz immer um 2 zunimmt. Da die Wege maximal $n - 1$ lang sein können, kann eine Kante höchstens $\frac{1}{2}n$ mal neueingefügt werden. Das schlimmste was passieren kann ist, wenn es $2m$ Kanten im Restnetz gibt, dann hat man $\frac{1}{2}n \cdot 2m$ Iterationen, jede kostet mit Breitensuche $O(m)$.
BV: Genau. So, was soll das denn mit den Rückkanten? Kann ich das nicht weglassen in der Definition von rest_f?
FW: Die sind dafür da, damit der Algorithmus seine Fehler korrigieren kann quasi. Ausserdem funktioniert der Korrektheitsbeweis sonst nicht. Da hat man nämlich...
BV: Jaja, das werden Sie schon können. Sagen Sie mir nur mal die grundliegende Idee.
FW: Man benutzt minimale Schnitte. Das ist quasi das duale Problem zu den Flüssen. Das sind solche Partitionierungen der Knoten in ...
BV: Das *duale* Problem? Haben Sie eine Idee, warum das so heisst?
Verdammt. Ich hätte dieses Wort nicht sagen dürfen :-) Jetzt fragt er Zeug über duale lineare Programme, das wollte ich eigentlich

vermeiden.
FW: Der Begriff kommt natürlich aus der linearen Programmierung, da betrachtet man zu einem linearen Programm das zugehörige duale LP. Bei einem LP in kanonischer Form $Ax \leq b, x \geq 0$ will man ja immer $c^T x$ maximieren. Man kann jetzt ein zweites LP bauen, wo man versucht eine obere Schranke für den Zielfunktionswert des ersten LPs zu minimieren. Das nennt man dann das duale LP. Das ist dann so ein $A^T y \geq c$ und man soll $y^T b$ minimieren. Das kann man dann wieder in kanonische Form bringen, indem man ...
BV: Jaja. Und was hat das mit Flüssen zutun?
FW: Naja, ich kann ein Flussproblem als LP formulieren, das duale LP ist dann der minimale Schnitt.
BV: Wie geht das?
FW: Hmmm. Für jede Kante eine Variable, und dann ...
BV: Nein, so gerade nicht. Also, das geht schon, aber dann klappt das mit dem dualen LP nicht.
FW: Achja, für jeden *Pfad* eine Variable. Dann hat man in den Nebenbedingungen diese komische Schreibweise $\sum_{P \ni e}$, d.h. man summiert über alle Pfade, die das e enthalten. Aber das krieg ich nichtmehr zusammen :-(
BV: Macht nichts. Ich schreib Ihnen das mal auf... So, wie lautet jetzt das duale LP?
FW: Öööh.. Krieg ich nicht hin.
BV: Ok, macht nichts. *Er schreibt die Lösung hin und erklärt sie mir ganz genau.* Alles verstanden? Ok, dann machen wir mal weiter mit dem SeideLP.
FW: Das ist ein randomisierter Algorithmus zum Lösen von linearen Programmen, und zwar ist das ein Las Vegas Algorithmus. Das ist ganz interessant, der berechnet nämlich *immer* die korrekte Lösung, obwohl er zufällig arbeitet. Die Laufzeit ist allerdings nicht immer gut, sondern nur die erwartete Laufzeit. Die ist $O(m \cdot d!)$.
BV: Genau, und für kleine d, wie bei geometrischen Problemen, ist das jetzt toll. Wie funktioniert der Algorithmus?
FW: Man wählt zufällig eine Hyperebene aus, die entfernt man. Dann löst man das rekursiv und guckt obs geklappt hat, d.h. ob die Lösung die rausgenommene Bedingung verletzt. Falls nicht, ist man fertig. Falls doch, muss man aufpassen. Jetzt schnappt man sich die Gleichung, stellt die nach einer Variable um, setzt die überall ein, auch in den Box Bedingungen (die hat man vorher reingenommen, um sicherzustellen, dass das LP beschränkt ist), und jetzt hat man zwei Nebenbedingungen mehr und eine weniger (also insgesamt eine mehr) und eine Dimension weniger.
BV: Sehr schön. Stellen Sie mal die Rekursionsgleichung für die Laufzeit auf.
FW: Da guckt man sich einfach jeden Schritt einzeln an, nämlich ...
BV: Machen Sie mal nur die interessanten Schritte.
FW: Das sind natürlich die rekursiven. Da hat man dann $T(m,d) = ... + T(m-1,d) + T(m+1,d-1) + ...$ So, da fehlt jetzt noch ein $\frac{d}{m}$ vor dem letzten Term, weil der wird nicht immer ausgeführt. Könnte ja sein, dass man beim zweiten Schritt schon fertig ist.
BV: Genau, sonst wär das auch garnichtmehr linear! Warum denn ausgerechnet $\frac{d}{m}$?
FW: Naja, weil wir angenommen hatten, dass das LP nicht degeniert ist. Man kann dann nämlich die optimalen Lösungen eindeutig durch den Schnitt von d Nebenbedingungen charakterisieren und dann ...
BV: Jaja, genau, ich seh schon, sie haben das verstanden. Gehen wir mal weiter zu den Approximationsalgorithmen. Da gibt es einen sehr klassischen ...
FW: Christofides!
BV: Genau :-)
FW: Da geht es um das Traveling Salesman Problem. Man kann leicht zeigen, dass für polynomialzeit-berechenbares α kein $\alpha(n)$-Approximationsalgorithmus existieren kann, deswegen schränkt man sich erstmal auf metrisches TSP ein, d.h. die Distanzmatrix muss die Dreiecksungleichung erfüllen.
BV: Genau, das ist in der Praxis ja eh immer erfüllt.
FW: Wir hatten zuerst eine 2-Approximation ...
BV: Jaja, das will ich garnicht sehen.
FW: Okay, also Christofides macht jetzt einen minimalen Spannbaum. Der kann schonmal nicht teurer sein als die TSP Tour, denn wenn man aus der Tour eine Kante rauswirft, bekommt man einen Spannbaum. Jetzt sucht man in diesem Baum die Knoten mit ungeradem Grad V' und bestimmt darauf ein min-cost Matching. Das ist hier sogar ein perfektes Matching. Das Matching packt man dann in den Baum mit rein. Da muss man aufpassen, falls die Kanten vorher schon da waren, muss man die doppelt reinnehmen. Jetzt gibt es einen tollen Satz aus der Graphentheorie, das sogenannte *Handschlag Lemma*, das sagt $\sum \deg(v) = 2 \cdot |E|$ und damit kann man sofort folgern, dass die Kardinalität von V' gerade ist. Jetzt baut man eine Eulertour, das klappt immer, wenn alle Knoten geraden Grad haben. Und jetzt kommt der entscheidende Schritt, der bei normalem TSP nicht funktioniert. Man überspringt Knoten, die man schon gesehen hat. Das ganze ist jetzt eine 1.5 Approximation. Die 1 kommt von dem Baum, hab ich ja schon gesagt, und für die 0.5 überlegt man sich einfach, dass man eine TSP Tour in zwei Matchings zerlegen kann, und das günstigere der beiden kostet höchstens $\frac{1}{2} opt$.
BV: Genau, machen wir mal weiter mit Netzwerkalgorithmen.

Netzwerkalgorithmen

BV: Machen wir mal etwas Spieltheorie.
Juhu, das einzige Thema in dieser Vorlesung, das mir wirklich gut gefallen hat!
BV: Erzählen Sie mal etwas zu Congestion Spielen.
FW: Ja, das ist so ein Tupel $\Gamma = (N, R, (\Sigma_i)_{i \in N}, (d_r)_{r \in R})$ von Spielern, Resourcen, dann solchen $\Sigma_i \subseteq 2^R$, das sind sogenannte Strategienmengen, und eine Abbildung $d_r : \{1, ..., n\} \to \mathbb{Z}$ für jede Resource, die bildet von der Anzahl der Spieler, die eine Resource belegen ab. Das ist das sogenannte Delay der Resource. Wir haben uns jetzt gefragt, ob man immer solche Gleichgewichte finden kann. D.h. ein Zustand in dem sich kein Spieler mehr selbstständig verbessern kann, also, ohne die Hilfe der andern. Das sind sogenannte Nash Gleichgewichte.
BV: Und? Gibt es die in jedem Spiel immer?
FW: Hmmm.. Also bei Congestion Spielen Ja, im allgemeinen müsste ich raten und würde Nein sagen.
BV: Okay, für *pure* Nash Gleichgewichte stimmt das.
FW: Ja, also, man guckt sich dafür diesen Transitionsgraphen an, da hat man für jeden State einen Knoten und Kanten, wenn sich ein Spieler verbessert. Der Graph ist natürlich endlich, also kann es da eh keine unendlichen einfachen Wege geben, aber es könnte ja sein, dass man im Kreis läuft. Aber Rosenthal hat sich dann überlegt, dass das nicht geht. Der hat nämlich gezeigt, dass jede Folge von verbessernden Zügen immer endlich ist. Damit folgt dann, dass es immer Nash Gleichgewichte gibt.
BV: Wieso das?
FW: Also er hat das mit einer Potentialfunktion gezeigt: $\Phi : \Sigma_1 \times ... \times \Sigma_n \to \mathbb{Z}, S \mapsto \sum_{r \in R} \sum_{i=1}^{n_r(S)} d_r(i)$. Ach, das hatte ich eben vergessen. Das n_r ist die Anzahl der Spieler, die in diesem State die Resource r belegen. So, man summiert hier jetzt quasi das Delay für einen Spieler, zwei Spieler, drei Spieler, etc. auf. Rosenthal hat jetzt gezeigt, wenn ein Spieler sich um $\triangle$ verbessert, dass dann die Potentialfunktion auch genau um $\triangle$ abnimmt. Der Wertebereich von Φ ist natürlich beschränkt, das sieht man ja sofort, und dann hats man schon.
BV: Wie ging das genauer? Das war ein netter Trick.
FW: Ja, man ändert einfach die Summationsreihenfolge und guckt sich den Spieler, der sich verbessert, als letztes an. Dann sieht mans.
BV: Genau. Wie schnell findet man ein Nash Gleichgewicht?
FW: Dafür hatten wir eine eigene Komplexitätsklasse gebaut. Also, es gibt da einen Spezialfall, symmetrische Netzwerkspiele, da ist das trivial und geht mit Reduktion auf Min-Cost-Flow. Aber im allgemeinen ist das PLS-vollständig. PLS heißt Polynomial Local Search. Da muss man wieder aufpassen, das sieht so aus, als hiesse das, dass es polynomiell geht. Das heisst aber nur, dass man sich polynomiell verbessern kann, aber man muss das eventuell exponentiell oft tun.
BV: Und wie schnell geht das in Potentialspielen?
FW: Hm. Das Wort habe ich noch nie gehört.
BV: Oh, Mist, das habe ich dieses Jahr auch garnicht in der Vorlesung gehabt. Mein Fehler. Ok, wie haben wir gezeigt, dass das PLS-vollständig ist?
FW: Über eine PLS-Reduktion von Max-Cut auf unser Problem.
BV: Genau.
Ich habe jetzt noch genauer die Definition von PLS und PLS-Reduktion erklärt. Danach hat Herr Vöcking mir noch ausführlich ein paar Details zu dem Reduktionsbeweis erzählt ohne weitere Fragen dazu zu stellen.
BV: So, das reicht mir jetzt. Ich bin fertig.

Cryptography I+II

RM: Fangen wir mal mit dem Begriff der perfekten Sicherheit an. Erzählen Sie mal etwas.
FW: Ja, man modelliert den Klartext, den Schlüssel und den Ciphertext als Zufallsvariablen. Dann definiert man Entropie und ...
RM: Nicht so schnell. Erklären Sie das mal genauer mit der Zufallsvariable.
Habe jetzt irgendwas zusammengestammelt, die Frage kam recht unerwartet.
RM: Was ist Entropie?
FW: Das ist definiert als ...
RM: Nein, halt, die Definition interessiert mich garnicht. Was ist das anschaulich!
FW: Hm, also, das ist quasi der durchschnittliche Informationsgehalt. Dann hatten wir noch die bedingte Entropie ...
RM: Nicht so schnell! Schreiben Sie erstmal hin was perfekte Sicherheit ist.
FW: Ein Kryptosystem heisst perfekt sicher, wenn $H(\hat{M}) = H(\hat{M}|\hat{C})$ ist. Wobei jetzt ...
RM: Jetzt warten Sie doch mal. Message und Key sind ja stochastisch unabhängig, das wissen Sie bestimmt. Wie ist das mit dem Ciphertext und der Message?

FW: Die sind es, falls das System perfekte Sicherheit hat und auch umgekehrt.
RM: Können Sie das beweisen?
FW: Öh... Ämm.. Nein, aber das folgt vermutlich sofort mit der Definition der bedingten Entropie, die ist nämlich ...
RM: Jaja, das will ich nicht sehen. Bei dieser Gleichheit die Sie da hingeschrieben haben. Eine Richtung gilt immer. Welche?
FW: Hmm. $\leq$ gilt immer. *War natürlich genau die falsche :-).*
RM: Ja genau. Ähm. Nein, das ist quatsch. Die andere. Okay, was ist Redundanz?
FW: Da ging es um Kodierungen. Das hängt mit der Entropie zusammen, das ist nämlich gerade $1 - \frac{H(X)}{\log(m)}$, wobei m die Kardinalität des Quellalphabets ist. Je höher die Entropie ist, desto geringer ist also die Redundanz.
RM: Was ist das Noiseless Coding Theorem von Shannon und was hat das mit Redundanz zutun?
FW: Öööhhh. Das macht eine Aussage darüber, wie gut man eine Quelle komprimieren kann, also, verlustfrei. Äähh.. Da steckt irgendwie die Redundanz drin.
Prof. Mathar hat jetzt langsam gemerkt, dass ich Stochastik ziemlich doof finde und da nicht besonders viel Ahnung von habe. Er erklärt mir genau, was er hören wollte.
RM: Gehen wir mal zum One-Time-Pad.
FW: Ja, das ist so das Standard-Beispiel für perfekte ...
RM: Jetzt hören Sie doch erstmal zu!
Prof. Mathar malt ein Bild auf für ein Cryptosystem, ähnlich dem One-Time-Pad, nur dass jedes Keyzeichen immer doppelt vorkommt. Während er zeichnet, quatsche ich Ihn voll und erzähle, dass das nicht sicher sein kann.
RM: Sie sollen doch abwarten und nicht einfach das Zeug aus den Prüfungsprotokollen nachplappern! Also, was ist denn jetzt die Wahrscheinlichkeitsverteilung der Zufallsvariable für die Keys?
FW: Ööööh. Gleichverteilt?
RM: Das ist ja totaler Blödsinn.
FW: Öööh...
RM: Wieviele Messages und wieviele Keys gibt es jetzt hier bei Länge $2t$ über Alphabet A?
FW: Naja, $|A|^t$ und $|A|^{2t}$. Also keine Gleichverteilung.
RM: Genau, gehen wir mal zum Diffie-Hellman-Schlüsselaustausch. Erstmal: Ich sage Ihnen mal ein paar natürliche Zahlen: a, x, n. Wann ist sowas hier lösbar: $y = a^x \mod n$.
FW: Das ist der diskrete Logarithmus, und zwar ...
RM: Denken Sie erstmal nach!
FW: Oh, sie wollen nach y lösen. Das geht natürlich immer.
Er streicht das erste x durch und schreibt y hin.
FW: So, jetzt ist es der diskrete Logarithmus.
RM: Ist das immer lösbar?
FW: Nein, im allgemeinen natürlich nicht. Aber wenn das y teilerfremd zu n ist und das a genau Ordnung $\varphi(n)$ hat, also ein primitives Element ist, dann ja.
RM: Okay, schön. So, jetzt erzählen Sie mal wie Diffie-Hellman funktioniert.
FW: Also, Ich möchte mit Ihnen einen Schlüssel austauschen. Dazu einigen wir uns erstmal auf eine Primzahl p und einen Generator modulo p.
RM: Wo kriegen wir das p her? Und den Generator? Wie findet man den?
FW: Also das p ist einfach, da picken wir uns einfach zufällig eine große Zahl raus und gucken mit Miller-Rabin ob die prim ist. Wenn nicht, nehmen wir die übernächste Zahl und machen das gleiche, und so weiter.
RM: Und den Generator? Gibt es den immer?
FW: Ja, geben tuts den für Primzahlen immer. Aber damit wir den einfach finden, wählen wir das p so, dass wir die Faktorisierung von $p-1$ kennen. Also, eigentlich macht man das anders. Man wählt natürlich eine Primzahl q so, dass $2q+1$ auch prim ist, und das nehmen wir dann als p. Dann hatten wir nämlich einen Satz, der sagt, dass a genau dann primitiv ist modulo p, wenn $a^{\frac{p-1}{p_i}} \neq 1$ ist modulo p für alle Primfaktoren von $p-1$, also hier nur 2 und q.
RM: Und wie kriegen wir das a jetzt?
FW: Das raten wir. Es gibt $\varphi(\varphi(p))$ viele davon, dann kann man ganz leicht ausrechnen, dass die Wahrscheinlichkeit, dass man falsch rät nur ungefähr 50 Prozent ist.
RM: Okay, schön. Weiter?
FW: Ich rate ein $x \in \{2, ..., p-2\}$. $p-1$ wäre doof. Dann schicke ich Ihnen a^x. Sie machen das gleiche und schicken mir Ihr a^y. Jetzt kennen wir beide a^{xy}.
RM: Genau. Da gibt es einen schönen Man-In-The-Middle Angriff.
FW: Ja, Herr Vöcking setzt sich jetzt zwischen uns und fängt das a^x ab. Er potenziert das mit q. Das q kennt er, denn p kennt ja jeder. Wir sind hier in einem endlichen Körper, also hat diese quadratische Gleichung nur 2 Lösungen, nämlich ± 1. Die probiert er aus.
Die beiden fanden es glaube ich ganz nett, dass ich das Protokoll nicht auf dem Papier mit A und B erklärt habe, sondern mit mir und Ihnen als Personen :-)

RM: Genau. Sehr schön. Jetzt möchte ich das gerne auf einer elliptischen Kurve machen. Wie geht das?
FW: Das geht ganz genauso. Das funktioniert in jeder zyklischen Gruppe. Das einzige was sich ändert ist die Notation... Naja, und man muss aufpassen, weil die Gruppensturktur auf einer elliptischen Kurven nur in Spezialfällen zyklisch ist, daher nehme ich einfach irgendeinen Punkt P der Kurve und nehme die von ihm erzeugte Untergruppe, die ist natürlich zyklisch. Jetzt das gleiche Spiel, das schreibt man dann halt additiv, also schicke ich Ihnen ein $x \cdot P$ und ...
RM: Was ist das denn? Ist das x auch ein Punkt der Kurve? Wie kann man denn hier multiplizieren??
FW: Nein, natürlich nicht, das x ist eine ganze Zahl. Da kann man sich natürlich auf die offensichtliche Art eine Skalarmultiplikation bauen. Ich addiere einfach x-mal das P zu sich selbst.
RM: Genau, aber ist das denn schnell genug?
FW: Ja, da nehme ich Square-and-Multiply, das klappt in jeder Gruppe und ...
RM: Jaja, ist klar. Und wie addiert man jetzt?
FW: Naja, bei reellen Kurven kann man das schön erklären, man nimmt eine Sekante und dann ...
RM: Wie soll das denn gehen?
FW: Hier garnicht, hier nimmt man eine Tangente, wenn man P zu sich selbst addieren will und ...
RM: Ja, reicht. Ich bin zufrieden. Gehen Sie bitte raus.

Fazit

Alles in allem war es eigentlich eine nette Prüfung, auch wenn Herr Mathar mich oft unterbrochen hat. Zu seiner Verteidigung muss ich sagen, dass ich auch sehr hektisch war und einfach Zeug erzählt habe obwohl er die Frage nicht zuende gestellt hat. Ich versuche normalerweise immer alles zu erzählen, was mir einfällt. Herr Vöcking fand das sehr gut hatte ich das Gefühl, Herr Mathar überhaupt nicht. Also, alles in allem: Prof. Mathar ist meiner Einschätzung nach ein sehr netter Mensch und auch als Prüfer zu empfehlen (alleine schon weil seine Vorlesung eine der besten ist, die angeboten wird), allerdings sollte man ihm nicht auf die Nerven gehen!

Herr Vöcking ist der freundlichste Prüfer, den ich bisher gesehen habe. Er hilft sehr viel und erzählt einleitende Sachen und erklärt einem sogar das, was man nicht konnte, damit man noch etwas lernt. Er ist sehr geduldig, lässt immer ausreden und wird nicht böse, wenn man ihm mal ins Wort fällt, weil man unbedingt antworten will :-) Nach der Prüfung hat Herr Vöcking mich noch gefragt, wo ich denn vertiefen möchte, und mir gesagt, dass ich mich doch melden sollte, falls ich eine Diplomarbeit suche.

Dass ich eine 1.0 bekommen habe, hat mich sehr überrascht. Ich hätte mir selbst allerhöchstens eine 1.3 gegeben, eher eine 1.7.

Zum Stoff: Wie bereits gesagt, hat mir die Vorlesung von Prof. Mathar (Krypto) extrem gut gefallen und ist mir sehr leicht gefallen, da ich Mathematik als Nebenfach habe und daher bereits bestes Vorwissen in Zahlentheorie und Algebra hatte (Stochastik leider nicht so sehr :-)). Effiziente Algorithmen kann ich auch nur empfehlen. Netzwerkalgorithmen würde ich allerdings nicht noch einmal in eine Prüfung nehmen, da mir der Stoff überhaupt nicht gefallen hat (ausser das einzige Kapitel, was glücklicherweise in der Prüfung dran kam :-)).