Prüfungsprotokoll Theoretische Informatik

Datum:	29.09.2001
Prüfer:	Hromkovic
Gebiete:	Effiziente Algorithmen (Hromkovic) Randomisierte und Approximative Algorithmen (Hromkovic) Compilerbau (Indermark)
Note:	1.3

Allgemeine Anmerkungen:
Im Folgenden beziehe ich mich des öfteren auf das Buch "Algorithms for Hard Problems" von Prof. Hromkovic, daß ich jedem,
der sich auf diese Prüfung vorbereitet, nur wärmstens empfehlen kann, insb. weil es wesentlich kompletter ist, als
jede Mitschrift die ich bisher gesehen habe. Desweiteren empfehle ich das Buch von Cormen Leiserson et al. "Introduction
to Algorithms" Es stellt die ultimative Referenz für alle Standardalgorithmen dar und deckt somit weite Teile der Eff.
Alg. Vorlesung ab. Da wo dieses Buch aufhört setzt dann bündig das Buch von Herrn Hromkovic auf.
Bleibt noch anzumerken, daß Herr Hromkovic auf die 3. VL (die fremd gelesene) generell wenig(er) Wert zu legen scheint. (Oder, um
es mit den Worten von Jakob zu sagen: "Wer für die VL vom Indermark lernt, ist selber schuld.") Ganz so kann ich das zwar
nicht sagen, aber es stimmt schon, daß in der Prüfung nur an der Oberfläche dieses Stoffes gekratzt wird, während Herr Hromkovic
bei seinem eigenen Stoff schon mal gerne ganz genau nachfragt. Aber seht selbst...
****
****
****
****

H: Womite wollen Sie anfangen?
P: Compilerbau wär nett.
H: Gut... Was ist eine LL(k) Grammatik?
P: Eine kontextfreie Grammatik G ist LL(k) gdw. ... (siehe Skript CB)
H: Wozu braucht man das?
P: Zur deterministischen Simulation des Top-Down Analyseautomaten. Mit Hilfe der LL(k) Grammatiken kann man
durch eine k-lookahead eindeutig bestimmen, welche Produktion im nächsten Schritt zur Ableitung eines
Wortes benötigt wird.
H: Kann man eine CFG immer in eine LL(k) Grammatik transformieren?
P: Nein, allein schon aufgrund der Tatsache, daß det. CFL echte Teilmenge von ndet. CFL ist. Das Wortproblem
für allgemeine Sprachen ist mit dem CYK-Alg. in O(n^3) lösbar. Die det. Simulation eines Top-Down-Analyse-
Automaten läßt sich aber in O(n) realisieren.
H: Können Sie so eine Einordnung auch für LR(k) geben?
P: ???Können Sie das vielleicht anders formulieren???
H: Was können sie über die Mächtigkeit von LR(k) für bel. k sagen?

P: Das Argument, daß det CFL echte Teilmenge von ndet. CFL ist gilt auch hier. (Anm.: Es wäre aber interessant,
ob die unendliche Vereinigung über alle k der durch LR(k)-Grammatiken beschriebenen Sprachen genau ndet. CFL
bildet???)

H: O.k. machen wir weiter...Semantische Analyse... Was sind Attributgrammatiken?

P: (Anm.: Autsch! Eigentlich war ich nur auf syntaktische und semantische Analyse vorbereitet, aber es gubt da so
einen netten Satz in der VL von Herrn Indermark...)
Attributgrammatiken dienen zum erfassen kontextsensitiver
Eigenschaften des Quellprogramms, der sog. statischen Semantik, die mit Hilfe der kontextfrei durchgeführten
Syntaxanalyse nicht erkannt werden können. Als Ergebnis erhält man einen attributierten Ableitungsbaum.

****
****

H: O.k. Das reicht. Machen wir weiter mit effizienten Algorithmen... Habe ich in der VL parametrisierte Komplexität
eingeführt?

P: Ja, haben Sie. Par-Kompl. ist eine Designtechnik für Optimierungsalgorithmen, bei der man die Eingabemenge
bzgl eines Parameters in potentiell unendlich viele Teilmengen unterteilt. Dabei ist eine Parametrisierung
def. als P: L-->N und Set_U(k):={...} (Def. 3.3.1.1) Dabei ist zu beachten, daß der Parameter die
"Schwierigkeit des Problems" erfassen sollte.

H: Kennen Sie ein Beispiel?

P: Ja, z.B. ParVC (Alg 3.3.2.4) ... Idee geschildert (Observations 3.3.2.2 + 3.3.2.3) und Arbeitsweise skizziert.
Laufzeit O(n+k^(2k)).

H: Das haben wir verbessert.

P: Mittels eines Divide&Conquer Ansatzes (DCVC) basierend auf der Tatsache, daß für jede Kante e=(u,v) in E gilt:
Entweder u in VC oder v in VC (Obs. 3.3.2.6) Die Laufzeit läßt sich dann drücken auf O(n2^k).

H: Kennen Sie den Satz von Cook-Levin?

P: Ja: SAT ist NP-vollständig.

H: Kennen Sie Beweis?

P: Ja, allerdings muß ich gestehen, daß ich Ihnen nur die Konstruktion nach dem Buch von Michael Sipser anbieten kann.
(Anm.: Ein wie ich finde sehr lesenswertes und vor allem gut lesbares Buch, in dem so mancher Beweis der theo.
Inf. deutlich einfacher nachzuvollziehen ist, als die entsprechende Version aus der VL :-)))

H: Schönes Buch... Glaube ich Ihnen. Beschreiben Sie die Idee der lokalen Optimierung!

P: Um das Prinzip der lokalen Suche anwenden zu können, stellt man zunächst jede zul. Lsg. zu einer

Eingabe x als Vektor dar. Auf dieser Darstellung def. man eine Nachbarschaft (Def. 3.6.1.1). Der Std.Alg. zur lokalen Suche (LSS(Neigh) S.175 unten) generiert nun zunächst eine gültige Lsg. und verbessert diese dann sukkzessive durch lokale Transformationen, bis keine Verbesserung mehr möglich ist. Dabei treten die folgenden beiden Probleme auf: 1) Exaktheit einer Nachbarschaft 2) p-zeit Durchsuchbarkeit (Def. 3.6.3.1) ->entweder man braucht zu lange oder man bleibt in einem lok. Opt. stecken.

H: Können Sie mir das am Beispiel TSP zeigen (Anm.: Wenn Euch diese Frage gestellt wird, dann geht es aller Wahrscheinlichkeit nach um eine 1.x)

P: Ja. ->Skizze mit "Diamanten"-Formation S.191 oben (Beweis zu Theorem 3.6.3.10). Erläuterung der Skizze (Er wollte zwar nicht alle Kanten exakt beschriftet haben, dafür aber die Anzahl der sub-Opt. Lsg. und deren Güte ganz genau wissen.)

****

H: O.K. Kommen wir zum letzten Teil. Nennen Sie mir ein Beispiel für einen LasVegas Algorithmus.

P: Da wäre z.B. der Alg. zur Bestimmung von quadratischen Nichtresten (Alg.5.3.2.4) zu nennen. Def. von q.R. und q.n.R. Kriterien (Theorem 5.3.2.2+5.3.2.3) und Idee des Alg.

H: Können Sie 5.3.2.3 beweisen?

P: Ja klar, und zwar.... ähhh ... ähh.... (Und da verließen sie ihn... Beweis ist aber eigentlich nicht weiter schwer wie man im Buch auf S.333 oben sieht.)

H: Egal, kommen wir vielleicht gleich noch mal drauf zu sprechen. Wir haben da noch einen Alg. für den Vergleich von Polynomen gemacht.

P: Ja. Erstmal ist grundsätzlich für das Problem zu sagen, daß für dieses Problem nicht bekannt ist, ob es in P ist bzw. ob NP-schwer (Anm.: oder vielleicht beides, wenn P=NP :-))) Die Idee des Alg. ist die beiden Polynome für zufällige a_1 ... a_m aus Z_p auszuwerten und dann zu vergleichen. Wir haben dann gezeigt, daß mit hinreichender W' ein Zeuge für die Ungleichheit gefunden wird.

H: Und wie hoch ist diese W' genau?

P: Das hängt davon ab, wie p für Z_p gewählt wird. Die Anzahl der Nullstellen eines Polynoms in m Var vom Grad <d ist md/2 (MÄÄÄÄP: Schnitzer Nr.2 md/p wäre die richtige Zahl gewesen) und somit muß man p>2dm wählen, um eine W' von mindestens 1/2 garantieren zu können (wieder richtig).

H: O.k. Das reicht mir. Skizzieren sie mir die Idee von Christofides (Alg 4.3.5.4)

P: Kurz die Schritte des Alg. aufgeschrieben und die Arbeitsweise erklärt Laufzeit O(n^4) Darauf hingewiesen, daß die Konstruktion des MST in O(n^2) keinesfalls mit dem allgemein bekannten Kruskal-Alg. möglich ist, sondern nur

mit einer "gewieften" Implementierung des Prim-Alg (Vgl. Cormen Leiserson). Approximationsgüte 3/2.

H: Beweisen Sie das!

P: Beweis skizziert (2.Teil des Beweises zu Theorem 4.3.5.5) incl. WC-Beispiel.

H: Gut. Skizzieren sie ein FPTAS für KP

P: Basiert auf dem Alg. DPKP (Alg 3.2.2.2) mit O(|I|^2 MAXINT(I)) und auf der Idee MAXINT ausreichend klein zu machen.

H: Wie klein?

P: (Setzte an, um die Formel aus dem Alg (4.3.4.9) hinzuschreiben)

H: Brauchen Sie nicht genau zu wissen. Die Größenordnung reicht...

P: MAXINT muß ploynomiell beschränkt werden d.h. der Faktor enthält 2^-t für ausreichend großes t.

H: Das reicht. Am Ende der VL hat Herr Seibert noch einige Techniken für den Beweis der nicht-Approximierbarkeit
von Problemen gemacht.

P: Jepp. 1)AP-Reduktion 2)GAP-Reduktion 3)PCP

H: Was ist PCP? (Auch hier: Wenn Euch diese Frage gestellt wird, dann geht es fast sicher nur noch um das x in 1.x)

P: Idee des probabilistichen Verifizierers beschrieben + die Idee des PCP-Satzes, nämlich NP anders zu beschreiben.
NP=PCP(log_2 n, O(1))

H: Was heißt das genau?

P: (Mußte einen Moment überlegen) Pr. Ver. ließt log_2 n Random Bits, überprüft danach O(1) Bits des Proofs und
entscheidet dann ob w in L oder nicht.

H: Hat Herr Seibert noch ein Beispiel für die Anwendung geschafft?

P: Ja, der letzte Satz der VL lautet: (Theorem 4.4.4.6) Für x=(9*2^12)^-1 ist Gap_(1-x,1)-3SAT NP-hart.

H: (Grinst) Können sie das beweisen?

P: Nein, das nun wirklich nicht...

H: (Grinst noch mehr) Nicht so schlimm... Danke...

P: Erleichtertes Aufatmen

****
****
****
****