Diplomprüfung
Theoretische Informatik

Prüfer:	Prof. Hromkovic
Fächer:	Effiziente Algorithmen
	Kryptographie
	Angewandte Automatentheorie
Datum:	23.09.2005
Note:	1.0

Effiziente Algorithmen

Hro: Was ist parametrisierte Komplexität?
ich: Bei parametrisierter Komplexität versucht man Algorithmen zu finden, die polynomiell in der Länge der Eingabe sind und möglicherweise exponentiell für einen Parameter k. Mann trennt dadurch also die Eingabeinstanzen eines Problems in die mit kleinem Parameter,die gut zu lösen sind und in die mit großem Parameter, die schlecht zu lösen sind.
Dafür kenne ich zwei Algorithmen zur Lösung des VC-Problems. Für den ersten Algorithmus benötigt man zwei Beobachtungen.

1. Hat man im Graphen ein VC der Größe höchstens k, müssen darin die Knoten enthalten sein, deren Grad größer k ist.
2. Hat man in einem Graphen ein VC der Größe höchstens m und der Knotengrad ist beschränkt durch k, dann hat der Graph höchstens $m \cdot (k+1)$ Knoten.

Der Algorithmus geht folgendermaßen: Mit Eingabe (G, k) und $H =$\{Knoten mit Knotengrad $> k$\}

> if $|H| > k$ reject
> else $m = k - |H|$
>
> > bilde Graph $G' = (V', E')$ indem aus G alle Knoten aus H und die dazu inzidenten Kanten gelöscht werden
>
> if $|V'| > m \cdot (k+1)$ reject
> else Suche im Graphen G' nach VertexCover der Größe m

Hro: Welche Komplexität hat der Algorithmus?
ich: $\mathcal{O}(k^{2k})$
Hro: Ist das unabhängig von Eingabe?
ich: Ähm.. natürlich in $\mathcal{O}(k^{2k} \cdot n)$
Hro: Und der andere Algorithmus?

ich: Dafür braucht man die Beobachtung: Für jede Kante $(u, v) \in E$ soll gelten, entweder u ist in VC oder v ist in VC. Der Algorithmus benutzt den Divide and Conquer Ansatz.
Hro: Ok, das reicht mir. In welcher Komplexität?
ich: $\mathcal{O}(n \cdot 2^k)$
Hro: Und wie kann man die Algorithmen verbinden?
ich: Hm... (das muss ich wohl immer überlesen haben), man führt zuerst einen D&C-Schritt durch und dann macht man in beiden Graphen mit dem zweiten Algorithmus weiter.(?)
Hro: Nee, genau umgekehrt. Statt der Suche in G' führt man da den D&C-Schritt ein.
ich: Ja, ok, stimmt.
Hro: Ok, wann nennt man ein Problem stark NP-schwer?
ich: Wenn es ein Polynom p gibt, für das Value(p)-U NP-schwer ist. Value(p)-U ist folgendermaßen definiert: (bin ins Stocken gekommen)... hm, ich verwechsel immer Value(p)-U und Lang$_U$.
Hro: Naja, es ist ja auf einem Integer-Valued-Problem definiert.
ich: Ja, Value(p)-U sind alle die Eingabeinstanzen x von U für die gilt Max-Int(x)$\leq p(|x|)$.
Hro: Was heißt das jetzt?
ich: hm
Hro: Na, in welchem Kapitel hatten wir das denn?
ich: Local Search.
Hro: Ja, auch. (Weiß nicht mehr genau was er gesagt hat, aber bin dann nedlich drauf gekommen, daß ein Problem stark NP-schwer ist, wenn es keinen Pseudo-polynomial-Zeit Algorithmus dafür gibt. Gäbe es einen pseudo-polynomial-Zeit Algorithmus für U könnte man Value(p)-U in polynomialzeit berechnen.)
Hro: Welches Problem ist stark NP-schwer?
ich: TSP.
Hro: Können Sie das beweisen?
ich: (Erinnere mich nicht mehr dran, die zwei letzten Fragen sind mir erst später wieder eingefallen und jetzt paar Wochen nach der Prüfung hab ich keine Lust mehr mir das anzusehen... :))
Hro: Ok, können Sie mir erklären, was ein PTAS ist?
ich: das ist ein Algorithmus, der zu einer Eingabe (x, ε) die Ausgabe $A(x)$ berechnet und es gilt, der relative Fehler von $A(x)$ ist $< \varepsilon$. Der relative Fehler ist so definiert: $\frac{|cost(A(x)) - Opt_U(x)|}{Opt_U(x)}$. Und es muss gelten, daß $Time_A(x, \varepsilon^{-1}) \leq p(|x|)$.
Hro: Wann ist die Zeit auch von ε abhängig?
ich: bei einem FPTAS.
Hro: Können Sie mir ein Beispiel für ein PTAS geben?
ich: Ja, PTAS für SKP. Ich bau mir das mal auf. Also, als Eingabe erhält man n Elemente mit Gewichten $w_1, \ldots, w_n$. Und man hat einen Rucksack, der beschränkt ist durch ein Gewicht b. Jetzt soll die Summe der Gewichte, die im Rucksack sind, maximal sein unter der Bedingung, daß sie kleiner b ist.
Der Algorithmus geht so:

> Sortiere zuerst die Gewichte $w_1 \geq w_2 \geq \ldots \geq w_n$.
> $k = \lceil \frac{1}{\varepsilon} \rceil$
> Bilde Mengen $S \subseteq \{1, \ldots, n\}$ mit $|S| \leq k$

im nächsten Schritt bilde aus S Mengen S^* indem man die nächst schweren Gewichte einfügt, für die gilt $\sum_{i \in S} w_i \leq b$

Im letzten Schritt wählt man die beste Lösung aus.

Die Komplexität für das Sortieren ist in $\mathcal{O}(n \log n)$.
Dann muss man sich ansehen wieviele Mengen S gebildet werden: $\sum_{i=0}^{k} \leq \binom{n}{i} \leq \sum_{i=0}^{k} n^k$ in $\mathcal{O}(n^k)$.
Man kann die Mengen in lexikographischer Ordnung aufbauen, daß man von einer Menge zur nächsten in $\mathcal{O}(1)$ kommt.
Jetzt muss man für die gebildeten Mengen jedes Gewicht überprüfen: in $\mathcal{O}(n)$, also Gesamtkomplexität in $\mathcal{O}(n^{k+1})$.
Um zu beweisen, daß das ein PTAS ist, schaut man sich die optimale Lösung $M = \{w_1, \ldots, w_{i_p}\}$ an.

1. Für $p \leq k$ haben wir schon im Mengenaufbau die optimale Lösung gefunden, also $\varepsilon = 0$.

2. Für $p > k$ haben wir im Mengenaufbau eine Lösung $P = \{i_1, \ldots, i_k\}$ gefunden, die die k schwersten Gewichte von M hat.
   Gilt jetzt $P^* = M$ haben wir die optimale Lösung gefunden mit $\varepsilon = 0$.
   Für $P^* \neq M$ muss es i_q geben mit $i_q \in M \backslash P^*$. Es gilt $i_q > i_k \geq k$, da w_{i_q} leichter sein muss als w_{i_k}.
   $cost(P^*) + w_{i_q} > b$
   $w_{i_q} \leq \frac{w_{i_1} + \ldots + w_{i_k} + w_{i_q}}{k+1} \leq \frac{cost(M)}{k+1}$, da $i_1, \ldots, i_k, i_q$ mindestens in M sind.

Hro: Ok, Sie können das. Kommen wir zu Kryptographie.
ich: Gut, mit 5 Sekunden Puse bitte, muss meine Gedanken darauf einstellen :).

Kryptographie

Hro: Fangen wir mit Rabin an.
ich: Rabin ist ein Public-Key Verfahren, da braucht man zwei Schlüssel einen öffentlichen und einen geheimen. Rabin beruht auf Quadrieren und Wurzel Ziehen.
Zur Schlüsselgenerierung wählt man zwei große Primzahlen p, q aus, die äquivalent zu $3 \bmod 4$ sein sollen aus Grünen der Effizienz. Man setzt $n = p \cdot q$.
Der secret key ist (p, q), der public key ist n.
Um jetzt eine Nachricht m, die kleiner n sein soll, zu verschlüsseln, setzt man $m \longrightarrow m^2$.
Hro: mod n
ich: Ja genau, und um einen Krypttext zu entschlüsseln, benutzt man den chinesischen Restsatz. Statt $\sqrt{c} \bmod n$ zu rechnen, berechnet man die Wurzeln für $[c \bmod p], [c \bmod q]$. Und zwar erhält man die Wurzel von $c \bmod p$, indem man $c^{\frac{p+1}{4}} \bmod p$ rechnet, wir hatten ja $p, q = 3 \bmod 4$ gewählt. Man erhält eine Wurzel $[x]$ und die andere ist $[-x]$. Das Gleiche für q. Dann erhält man für jeden Wert $[c \bmod n]$ bzw. $[c \bmod q]$ zwei Wurzel, insgesamt vier Wurzeln.
Wenn wir als Ergebnis haben $[x_1], [x_2]$ (x_1 ist Wurzel von $c \bmod p$, x_2 ist Wurzel von $c \bmod q$), dann

erhält man als Wurzel mod n den Wert: $[x_2 \cdot p \cdot d + x_1 \cdot q \cdot e]$, wobei $p \cdot d + q \cdot e = 1$ gilt; die Zahlen erhält man mit Hilfe des erweiterten euklidischen Algorithmus.
Um zu Vereinfachen aus den vier Wurzeln die Richtige rauszusuchen, kann man vorne an die Nachricht noch das Jacobi Symbol anhängen.
Hro: Gut, und wie schwer ist das?
ich: So schwer wie die Faktorisierung.
Hro: Können Sie das auch beweisen?
ich: Ja, das hab ich mir noch vor ein paar Tagen angesehen.
Hro: Also, in die eine Richtung ist klar.
ich: Ja, das hab ich ja grade gezeigt. Wenn ich p, q kenne, kann ich die Wurzel ziehen mit Hilfe des chinesischen Restsatzes.
Angenommen ich könnte Wurzeln berechnen, dann wähle ich ein x zufällig aus und berechne die Wurzel von x^2 mod n mit Hilfe meines Algorithmus also $A(n, x^2) = y$.
wenn gilt: $x = \pm y$, wähle ich erneut x aus.
wenn gilt: $x \neq \pm y$, dann gilt doch $n|x^2 - y^2$ aber $n \nmid (x - y)$ und $n \nmid (x + y)$. Also kann ich mit Hilfe des euklidischen Algorithmus den $gcd(n, x + y)$ berechnen und erhalte so einen Faktor von n.
(Steht im Appendix, ein Freund hatte mich drauf hingewiesen das zu lernen, es ist wohl auch schon paarmal drangekommen...)
Hro: Erklären Sie mal RSA.
ich: Zur Schlüsselgenerierung wählt man zwei große Primzahlen p, q. $n = p \cdot q$.
Man wählt ein e, sodaß $gcd(e, \varphi(n)) = 1$, wobei $\varphi(n)$ die Anzahl der Einheiten in $\mathbb{Z}_n$ ist und es gilt $\varphi(n) = (p - 1)(q - 1)$. Dann wählt man das inverse Element zu e also $ed = 1$ mod $\varphi(n)$, mit Hilfe des erweiterten Euklidischen Algorithmus und zwar berechnet der zu zwei Zahlen a, b nicht nur den $gcd(a, b)$ sondern auch e, d für die gilt $a \cdot e + b \cdot d = gcd(a, b)$.
Um eine Nachricht $m < n$ zu verschlüsseln, setzt man $m \longrightarrow m^e$ und um den Krypttext zu entschlüsseln setzt man $c \longrightarrow c^d$. Es gilt ja $m^{ed} = m$.
Hro: Sie meinen doch äquivalent.
ich: Ja natürlich, $m^{ed} \equiv m$.
Um das zu zeigen betrachtet man drei Fälle.

1. $m \in \mathbb{Z}_n^*$, d.h. m ist prim zu n.
   also gilt ja $m^{\varphi(n)} = 1$ mod n und dann kann man sagen $m^{ed} = m^{ed \bmod \varphi(n)} = m$, womit das gezeigt wäre.

2. $p|m$ und q teilt m nicht.

Hro: Ok, das reicht mir. Jetzt interessiert mich das Thema elektronische Wahlen. Da gab es doch die Möglichkeit einen Schlüssel auf k Autoritäten zu verteilen. Wie ging das?
ich: Hm, das kann ich nicht so gut. (Das hatte ich mir fast gar nicht angesehen, weil ich gar nicht mit so einer Frage gerechnet hatte) Also das beruht auf Polynomen, aber mehr weiß ich nicht.
Hro: Ach kommen Sie, wie kann man denn ein Polynom darstellen?
ich: $ax^k + bx^{k-1} + \ldots + z$
Hro: Die x-Werte kann man ja weglassen.

ich: Ja, man muss nur die $k+1$ Koeffizienten speichern.
Hro: Ok, und wie kann man ein Polynom noch darstellen? ... Denken Sie mal an die Fourrier-Transformation.
ich: Puh, daran erinnere ich mich nicht mehr.
Hro: Also, man kann ein Polynom mit Hilfe der Nullstellen darstellen. ... (hat noch mehr erklärt, weiß ich aber jetzt nicht mehr) Ok gut, dann erklären Sie mir aber doch noch was Zero-Knowledge ist.
ich: Man nennt ein interaktives Beweissystem mit Prover udn Verifier Zero-Knowledge, wenn es einen probabilistischen Simulator gibt, der in polynomialzeit abeitet und zu jedem Verifier V^* und einer Eingabe x ein akzeptierendes Transkript erzeut. Ein akzeptierendes Transkript ist eine Kommunikation in der im letzten Schritt der Verifier akzeptiert. Zusätzlich gilt, daß die Verteilung der Transkripte mit dem Simulator gleich der Verteilung der Transkripte ist, die durch einen richtigen Prover und den Verifier erzeugt werden.
Hro: Man kriegt also nach einer Kommunikation keine Information.
ich: ja.
Hro: Dann geben Sie mir mal ein Beispiel.
ich: Ich kenne da das Simplified Fiat Shamir Identification Scheme. Es gilt $p \cdot q = n$, $x = y^2$ und der Prover hat hierbei das Geheimnis y. Er kennt eine Wurzel von x und wird den Verifier davon überzeugen, ohne die Wurzel zu nennen.
Hro: Ok gut, kommen wir zur Automatentheorie.

Angewandte Automatentheorie

Hro: Deterministische endliche Automaten. Es gibt ja die Möglichkeit einen DEA zu minimieren. Auf welcher Kongruenz basiert die Minimierung?
ich: Auf der Nerode Kongruenz, die über einer Sprache L definiert ist.
$u \sim_L v$ wenn $\forall w \in \Sigma^*$, $uw \in L \Leftrightarrow vw \in L$
Hro: Wie geht man bei der Minimierung vor?
ich: Man geht über die Länge der Wörter und guckt für ein Paar von Zuständen, ob es ein Wort gibt, für welches man aus einem Zustand in einen Endzustand kommt, und aus dem anderen Zustand nicht. Diese Zustände trennt man.
Beim Blockverfeinerungsalgorithmus erhält man zum Schluss Blöcke von Zuständen; für die Minimierung setzt man einen Block gleich einen Zustand.
Hro: Wie sieht der erste Schritt vom Blockverfeinerungsalgorithmus aus?
ich: Man bildet zwei Blöcke $B_1 = Q \backslash F$ und $B_2 = F$.
Hro: Terminiert das?
ich: Ja nach $|Q|$ Schritten.
Hro: höchstens $|Q|$. Wie kann man feststellen, ob es nicht schon vorher abbricht?
ich: wenn im letzten Schritt keine Blockaufteilung stattgefunden hat, kann man aufhören.
Hro: Wie geht man bei den NEA's vor? Worauf beruht die NEA-Minimierung?
ich: Auf der Bisimulationsäquivalenz.
Hro: Ist der NEA optimal?
ich: Nein.

Hro: (Hier erinnere ich mich nicht mehr genau an die Frage, aber Herr Hromkovic wollte wissen wieso das NEA-Nicht-Universalitätsproblem PSPACE schwer ist)
ich:(wußte nicht genau wie ich anfangen sollte, erstmal Gedanken sortieren) Also, um zu zeigen, daß das NEA-NUP PSPACE schwer ist, zeigt man, daß sich jede Sprache $L \in$ PSPACE auf das NEA-NUP reduzieren läßt $L \leq_p$ NEA-NUP. Dazu konstruiert man für ein Wort w einen NEA, der genau dann nicht universell ist, wenn $w \in L$. Jetzt guckt man sich die Turingmaschine M an, die L in PSPACE erkennt. Eine Konfiguration von M hat die Form uqv. Und eine akzeptierende M, w-Berechnung ist eine Folge von Konfigurationen $u_0q_0v_0\#u_1q_1v_1\# \ldots \#u_sq_sv_s\#$, wobei $u_0q_0v_0$ Anfangskonfiguration ist. Jedes $u_{i+1}q_{i+1}v_{i+1}$ ist Folgekonfiguration von $u_iq_iv_i$ und $u_sq_sv_s$ ist Endkonfiguration. Hierbei gilt immer $|u_i| + |v_i| \leq p(n)$, da M in PSPACE.
Der Nea wird jetzt so konstruiert, daß er alles akzeptiert nur eben nicht die akzeptierende M, w-Berechnung. Also wenn $w \in L$, ist er nicht universell.
Dazu überprüft er ob $u_0q_0v_0 \ldots$
Hro: Ich will das jetzt mal etwas beschleunigen. Wie kommt der Nicht-Determinismus zustanden, denn den braucht man?
ich: Der NEA rät drei Positionen $u_iq_iv_i$, wofür der Folgewert $u_{i+1}q_{i+1}v_{i+1}$ keine Folgekonfiguration ist.
Hro: Gut danke, dann warten Sie bitte kurz draußen.

Ich fand die Prüfung super, Herr Hromkovic ist in der Joggingjacke gekommen, ganz lässig und cool. Er strahlt auch so eine Ruhe aus, da ist man direkt viel entspannter, sehr angenehm. Letztendlich hab ich mir viel mehr Stress gemacht als eigentlich nötig, hat ja alles gut geklappt. Herr Hromkovic meinte in der Nachbesprechung, daß er jedem Studenten einen Joker geben würde, ich hätte meinen gut bei den elektronischen Wahlen verspielt und sonst hätte ich alles gewußt (gut, zwei- dreimal mußte er nachhaken, bei Value(p)-U zum Beispiel, aber das ist mir auch später eingefallen. Er hilft einem auf die Sprünge, wenn man die Antwort nicht gleich parat hat. Ich kann Herrn Hromkovic als Prüfer nur weiterempfehlen auch wenn er ab jetzt nur noch in Zürich prüft.
Habs ziemlich ausführlich gemacht, so ziemlich alles genau aufgeschrieben wie er und ich es gesagt haben, weil mir in der Vorbereitungszeit eigentlich nur die ausführlichen Protokolle geholfen haben.
Also euch noch viel Erfolg!