Diplomprüfung Theoretische Informatik

Datum: 2.6.2008
Prüfer: Vöcking, Unger
Themen:
Effiziente Algorithmen (WS 06/07, SS 07)
Kommunikationsprobleme (WS 07/08)
Compilerbau (Vorlesung nicht gehört, Vorlesungsfolien Noll 06/07)
Dauer: ca 45min
Note: 1.0
Keine Garantie für Korrektheit oder Vollständgkeit (Gedächnissprotokoll)
Alle Rechtschreibfehler sind natürlich Absicht

1 Kommunikationsprobleme

U: Wie schwer ist das allgemeine Broadcast Problem

I: Das Broadcast Problem ist in NP

U: Wie kann man das beweisen

I: Das Broadcast Problem kann auf das SAT Problem reduziert werden. Von dem Ausgangsknoten wird für jede Variable der logischen Formel ein kritischer Weg erzeugt. Dann wird für jede Variable ein Weg für die Negation erzeugt. Der Broadcast Algorithmus muss entscheiden ob er zuerst die negierte oder nicht negierte Variable informert. In dieser Entscheidung liegt die NP-härte. Es gibt 2 hoch n Möglichkeiten. *(Danach hat mich Herr Unger unterbrochen und meinte der Rest wäre dann klar)*

U: Wie lange braucht der Broadcast auf CCC und BF

I: CCC: $\frac{5}{2}n - 2$ und BF: $\frac{3}{2}n$ *(Da hat mich Herr Unger auch schon unterbrochen und meinte das der konstante Faktor nicht so wichtig sei)*

U: Wir hatten ja eine einfache Strategie für den Broadcast Algorithmus in Zeit $\frac{5}{2}n - 1$. Wie konnte man denn noch eine Runde Sparen?

I: Der weitentfernteste Kreis wird in $2n - 1$ informiert. Es gibt zwei Wege dorthin über Kreis bzw Kreuzkanten. Man erreicht damit 2 Benachbarte Knoten in Zeit $2n - 1$ bzw $2n$. *(Da Unterbrach mich Herr Unger und sagte, dass das so nicht stimmt. Nachdem mir der Fehler aber nicht auffiehl erklärte er, dass die Knoten nicht benachbart sind, sondern Entfernung 2 haben.)* Dadurch spart man beim Broadcast auf dem Kreis eine Runde.

U: Kommen wir mal zu Gossip. Wie funktioniert denn der Gossip auf vollständigen Graphen im Einwegmodus

I: Man hat $n = 2m$ Knoten und teilt diese in zwei gleichgroße Teilmengen auf. Inheralb einer Menge ist keine Kommunikation erlaubt. Man hat also einen bipatiten Graphen. In den ersten 2 Runden tauscht jeder Knoten die Information mit dem zugeörigen Knoten aus der anderen Menge aus. Danach benutzt man die Fibonaci Zahlen. *(Da unterbrach mich Herr Unger wieder und wollte dann nur noch die Invariante wissen)* Zum Zeitpunkt t kennt jeder Knoten.... *(Guckt im Script nach ;-))*

U: Wie ist denn die Laufzeit

I: $k = min\{x|fib(x) < m\}$ Damit ist $r(K_n) \leq k + 1$ eine obere Schranke

U: Wie hatten wir denn die untere Schranke gezeigt *(Frage war ein bisschen anders weiß aber nicht mehr so genau.)*

I:
Man formt das Problem leicht um und berechnet eine Lösung für das Network Counting Problem. Jeder Knoten speichert keine Informationen mehr sondern nur wie viele Informationen er erhallten hat. Da er Informationen auch doppelt erhalten kann ist das Problem einfacher als Gossip. Es wird in einem Vektor gespeichert wie viel Informationen z Knoten i zum Zeitpunkt t hatte. $(z_1^t, ...z_n^t)$ Die einzelnen Schritte werden durch Matrizen dargestellt. Die Matrix ist eine Einheistmatrix mit zusätzlichen einsen. Wobei in jeder Spalte und Zeile maximal eine 1 stehen darf da jeder Knoten nur an einen Knoten senden bzw empfangen kann und $a_{ij} \neq a_{ji}$ (Einwegmodus). (Dann wollte Herr Unger gerne die Matrix TAT^{-1} wissen. Ich hab die Matrix definiert und B aufgemalt. *(Von hier an konnte ich nur noch die einzelnen Schritte, ohne sie wirklich verstanden zu haben. Herr Unger hat aber auch nicht gefragt warum man z.B die Spektralnorm als Abschätzung nehmen kann. Er hat auch keine weiteren Fragen zur Linearen Algebra gefragt))* Von der Matrix B berechnet man die Spektralnorm. Es gilt $\|B\| = \|TAT^{-1}\| = \|A\|$. Man berechnet das Minimal Polynom... *(Hier hat Herr Unger mich unterbrochen und meinte wieder, dass der Rest dann klar ist. Er wollte gerne noch die Abschätzung wissen die man am Ende bekommt)* Der Vektor α ist unser Lösungsvektor, den man erhällt wenn man die Matrizen mit dem Startvektor $(1, 1, ..., 1)$ multiplziert. $\|\alpha\| \leq (\prod_{i=1}^{r-2} \|B_i\|) * \|(1, 1...., 1)\| \leq \frac{1}{2}(1 + \sqrt{5})^{r-2} * \sqrt{n}$

U: Was gibt es denn für eine untere Schranke für α

I: $\|\alpha\| \geq \sum_{i=1}^{n} \alpha_i^2$ die normale Euklidische Norm. Die kann man abschätzen mit $\|\alpha\| \geq \frac{n}{2} * \sqrt{n}$

U: Wie kommt man denn auf die Abschätzung

I: *(Da wusste ich erstmal keine Antwort drauf. Herr Unger hat mir dann geholfen und die c_1, c_2, c_3 erwähnt)* Darauf hin viel mir wieder ein, dass man die Knoten zählt, die mehr als n Informationen haben und die die weniger als n haben. $c_1 + c_2 + c_3 \geq \frac{n}{2}$ *(Mehr konnte ich dazu nicht sagen er hat den Rest dann selber erklärt)*

U: Und wie ist jetzt die Abschätzung

I: Es gilt jetzt $\frac{n}{2} * \sqrt{n} \leq \|\alpha\| \leq \frac{1}{2}(1 + \sqrt{5}^{r-2}) * \sqrt{n}$. Daraus kann man r ausrechnen. Die Abschätzung ist dann $r \geq 2 + log_{\frac{1}{2}(1+\sqrt{5})} \frac{n}{2}$

U: Wie gut ist die Abschätzung denn im Gegensatz zu den Fibonaci Zahlen von unserem Algorithmus

I: *(Davon hatte ich keine Ahnung. Herr Unger hat dann nur kurz gesagt, dass sie sehr nah beieinander sind und dass sie sich nur um den Wert 1 oder 2 unterscheiden)*
U: Kommen wir mal zum Kapitel Wellenlängenzuweisung. Wie funktioniert das denn auf einem Stern

I: Man verdoppelt die äußeren Knoten und erstellt einen bipartiten Graph. Zwei Knoten sind genau dann verbunden wenn sie Nachrichten austauschen wollen. Auf bipartiten Graphen ist das Knotenfärbungsproblem nicht in NP. Also kann man das einfach ausrechnen. Nachrichten von oder zu dem mittleren Knoten können mit Greedy Nachgefärbt werden.
U: Herr Unger hat in meinem Beispielgraphen eine zusätzliche Kante reingemalt, so dass 2 Knoten sich gegenseitig Nachrichten schicken und wollte wissen was man denn dann macht
I: Musste kurz überlegen bis mir einfiel, dass entgegengesetzt laufende Nachrichten sich nicht behindern. Damit stellt das kein Problem da.

U: Was ist denn eine obere Schranke für k beim k-Knotenfärbungsproblem

I: $\Delta(G) + 1$

U: Was hat denn das Kantenfärbungsproblem mit dem Knotenfärbungsproblem zu tun

I: Linegraph

U: Sind die Probleme gleich schwer

I: Ich hab gesagt dass beide in NP sind (Das hat ihm allerdings nicht gereicht. Mehr wusste ich dazu allerdings nicht. Er hat dann noch einige Sachen dazu erklärt. Ich weiß aber nicht mehr so genau was

2 Vorbereitung Kommunikationsprobleme

Die Kapitel Broadcast, Gossip und Wellenlängenzuweisung sollte man auf jedenfall perfekt können
Bei Approximation ist Clustering und Poise wichtig. Die Heuristiken fragt er nicht ab
Systolischer Gossip kann dran kommen. Hier hat Herr Unger speziell sys Gossip auf Gittern erwähnt
Das Kapitel Netzwerkdesign fragt er eher nicht. Vieleicht bei 1.0 Kandidaten
Das Kapitel Fehlertollerranz findet er ‘langweilig‘. Die Aussage müsst iher selber bewerten ;)
Die beiden fehlenden Kapitel hat er garnicht erwähnt.
Am besten selber nochmal in die Sprechstunde gehen.
Die Informationen stammen aus 2 unabhängigen Gesprächen mit Herrn Unger (von mir und Kommilitone)

3 Effiziente Algorithmen

Das Kapitel werd ich nicht so ausführlich bearbeiten, da ja schon genug Protokolle dazu existieren
V: Wie funktioniert denn Ford Fulkerson

I: fv-Wege

V: Was ist denn der Resgraph

I: Definition $rest_f = ...$ aufgeschrieben

V: Wieso fügt man denn in entgegengesetzer Richtung eine Kante ein

I: Damit man einen Fluss in entgegengestzer Richtung fließen lassen kann und die gelöschte Kante wieder neu einfügen kann

V: Ja, aber wo braucht man das genau im Beweis. Es würde ja auch ohne funktionieren

I: (Konnte ich ihm auch nach weiteren hinweisen keine Antwort drauf geben)
V: Beweisen sie doch mal MINCUT = MAXFLOW den Schritt von 2 nach 3

I: Den Cut definiert. f(u,v) = c(u,v) und f(v,u) = 0 erklärt. Mir viel auf, dass man die Kanten in entgegengesetzter Richtung genau hier braucht, da sonst der Beweis nicht klappt. *(Mehr wollte Prof. Vöcking von dem Beweis auch nicht mehr wissen)*

V: Welche Laufzeit hat Ford Fulkerson

I: Pseudopolynominelle Laufzeit und normale aufgeschrieben und kurz erklärt (*Musste nichts beweisen*)

V: Wie kann man denn einen MinCut mit einem nicht randomisierten Algorithmus in n^4 berechnen *(**Achtung:** Die Lösung dazu steht nicht im Script. Ich konnte die Frage auch nicht beantworten)*
Man hat ja bei MinCuts keine Quelle bzw Senke. Also ruft Dinic n^2 mal auf mit allen möglichen Quelle-Senke paaren. Dann hat man n^5. Es reicht allerdings auch wenn man nur die Quelle Variable macht und die Senke fest läßt. dann hat man nur n Aufrufe und kommt auf n^4.

V: Kommen wir mal zu den Randomisierten Algorithmen. Wie funktioniert denn Contract
I: Kante uniform zufällig auswählen und kontrahieren. Am Ende bleiben 2 Knoten übrig. Die Kanten zwischen den Knoten definieren Cut. Der Cut muss nicht ein MinCut sein, falls zufällig eine Kante des MinCut kontrahiert wird

V: Wie zeigt man denn die Warscheinlichkeit

I: P[C=K] = ... hab ihm jede einzelne Umformung bis $\geq \frac{1}{n-i-+1}$ erklärt. Stochastische Abhängigkeit, Abschätzung für m_{i-1} usw. *(Weiter wollte er es nicht wissen)*

V: Wie funktionierte denn Makespan Scheduling auf allgemeinen Maschinen

I: Man definiert ILP, relaxiert dieses und rundet die Werte wieder. ILP ist in NP. Man muss allerdings einen Trick anwenden. Ein Orakel veräht einem opt. Man läßt dann nur noch zulässige Zuweisungen zu (S_T definiert).

V: Warum funktioniert das denn sonst nicht

I: *(Ich wusste es nicht. Prof Vöcking gab dann die Antwort. Irgendwas mit einem Gap zwischen gerundeter und ungerundeter Zahl. Steht im Script)*

4 Compilerbau

Ich hatte von Compilerbau nicht wirklich viel Ahnung weil ich nur 4 Tage dafür gelernt hatte. Die Fragen dazu waren allerdings nur sehr oberflächlich.
V: Wo braucht man denn CFG

I: In der Syntaxphase

V: Wie ist denn die Laufzeit zum lösen des Wortproblems bei CFG

I: Im Allgemeinen n^3. Durch die spezielle Struktur eines Programms gehts aber in linearer Zeit.

V: Was gabs denn für spezielle Grammatiken

I: LL(K) bzw LL(1) für den Top Down Ansatz und LR(K) bzw LR(0) und LR(1) für Bottom Up
V: Malen sie doch mal ein Diagramm auf, welche Menge in welcher enthalten ist

I: LL(1) $<$ LL(K) $<$ LR(K)

V: Wie funktiniert denn das Parsen mit LL(1) Grammatiken

I: Kompletten Automat mit act funktion definiert

V: Und was benutzt man bei LR(K)
I: Shift und Reduce *(Mehr wollte er nicht hören)*

V: Was gibt es denn für Attribute in der Semantischen Analyse

I: Inherite für TopDown und Synthetische für Botto Up

V: Wo benutzt man denn welche in einem Programm

I: *(Ich wusste es nicht. Prof Vöcking hat mir mehrere Minuten versucht durch Tips und Umformulierungen zu helfen, bin aber nicht drauf gekommen)* Kann euch die Antwort leider nicht verraten weil ich sie nicht mehr weiß ;-)

5 Zusammenfassung

Obwohl ich doch einige Sachen nicht wusste habe ich eine 1.0 bekommen, weil ich die 2 schweren Beweise zu Contract und r(K(n)) gut erklären konnte. Die Prüfungsatmosphäre war sehr entspannt. Sowohl Prof Vöcking und Herr Unger haben einem geholfen wenn man nicht weiter wusste. Notizen und Zeichnung wurde nicht an der Tafel sondern auf einem Blatt Papier gemacht. Außerdem kann man sich während der Prüfung ein Bild über seine Leistung machen, da der Beisitzer deine Antworten auf dem Protokoll bewertet.

Wenn man die ersten Fragen gut beantwortet werden die Fragen relativ schnell schwerer was ebenfalls ein gutes Zeichen ist. Sowohl Prof Vöcking als auch Herr Unger fragen einige Zusammenhangsfragen (wie z.B Wo benutzt man denn welche Attribute in einem Programm), die nicht unbedingt im Script stehen. Nach der Prüfung hat mich Prof Vöcking darauf hingewiesen, dass ich bei meiner nächsten Prüfung darauf acht

geben sollte auch solche Fragen beantworten zu können.