RWTH Aachen **WS 2008/09**
Lehrgebiet Theoretische Informatik Übungsblatt 1
Kneis—Langer—Richter—Rossmanith 23.10.2008

♡ Übung zur Vorlesung Parametrisierte Algorithmen ♡

Tutoraufgabe T1

Rufen Sie sich den Algorithmus für Vertex Cover, der auf einem beschränkten Suchbaum basiert, ins Gedächtnis. Können wir ein ähnliches Verfahren für Independent Set entwerfen? Gefragt ist nur eine informelle Antwort.

Lösungsvorschlag

Nein. Wenn wir bei Independent Set festgestellt haben, daß es keine Lösung gibt, in der der gewählte Knoten *nicht* enthalten ist, dann folgt lediglich, daß mindestens einer seiner Nachbarn im Set enthalten sein muß. Weil der Knotengrad im allgemeinen nicht beschränkt ist, kann der Ausgangsgrad der Knoten im Suchbaum größer als $f(k)$ werden.

Tutoraufgabe T2

Können Sie einen parametrisierten Algorithmus für Independent Set auf planaren Graphen angeben?

Hinweis: Überlegen Sie, ob das Problem aus Aufgabe T1 bei Planarität bestehen bleibt.

Lösungsvorschlag

Zunächst einmal können wir zu der intuitiven Einsicht gelangen, daß ein planarer Graph nicht ausschließlich aus Knoten hohen Grades bestehen kann.

Tatsächlich folgt aus Eulers Formel $|V| - |E| + |F| = 2$ und einigen grundsätzlichen Überlegungen, daß ein endlicher ungerichteter Graph $G = (V, E)$ ohne Schleifen und ohne Multikanten höchstens $3|V| - 6$ Kanten haben kann, wenn $|V| \geq 3$ gilt.

Den Sonderfall, daß es kein inneres *face* gibt, möge man sich getrennt überlegen; man sieht leicht, daß die obige Schranke für alle derartigen Graphen mit mindestens drei Knoten gilt. Ansonsten verwenden wir die folgende Argumentation:

Jedes *face* berührt mindestens drei Kanten. Jede Kante berührt höchstens zwei *faces*. Deshalb gilt $|F| \leq \frac{2}{3}|E|$ und somit $|E| \leq 3|V| - 6$. Daraus folgt, daß jeder planare Graph einen Knoten des Grades höchstens fünf hat.

Da wir also stets auf einem Knoten kleinsten Grades verzweigen können, erhalten wir einen Suchbaum der Größe $O(6^k)$.

Tutoraufgabe T3

Das Problem Clique ist so definiert: Gegeben sind ein Graph und eine Zahl k. Gefragt ist, ob es einen vollständigen Teilgraph der Größe k gibt. Der Parameter ist k.

Beweisen Sie: Clique ist genau dann *fixed parameter tractable*, wenn Independent Set es ist.

Lösungsvorschlag
Wie man sich leicht klarmacht, enthält ein Graph genau dann eine unabhängige Menge der Größe k, wenn es in seinem Komplementärgraph eine Clique der Größe k gibt. Dieser Komplementärgraph lässt sich in polynomieller Zeit berechnen. Damit impliziert ein Algorithmus mit Laufzeit $O(f(k) \cdot n^c)$ für das eine Problem einen Algorithmus mit Laufzeit $O(n^d + f(k) \cdot n^c)$ für das andere Problem (c und d Konstanten).

Tutoraufgabe T4
Entwerfen Sie einen möglichst effizienten Algorithmus, der VC für *Bäume* löst!

Lösungsvorschlag Wir wenden iterativ folgendes Verfahren an, bis der Graph keine Kanten mehr enthält (antworte „JA“) oder wir auf noch nicht kantenfreien Graphen bereits k Knoten gewählt haben (antworte „NEIN“): Wähle ein Blatt v des Baums und betrachte die eingehende Kante $\{u, v\}$. Nehme v in das VC auf und lösche sowohl u als auch v aus dem Graphen. Die Laufzeit dieses Verfahrens ist offensichtlich polynomiell.

Die Korrektheit folgt aus folgendem Argument: Jedes optimale VC für G muß entweder u oder v enthalten, damit die Kante $\{u, v\}$ abgedeckt ist. Weil u ein Blatt in G ist, kann es die beiden Knoten aber nicht gleichzeitig enthalten, da man sonst eine kleinere Lösung konstruieren könnte, die u nicht enthält, aber trotzdem alle Kanten von G abdeckt. Enthält eine optimale Lösung hingeegn u aber nicht v, dann können wir eine andere Lösung gleicher Größe konstruieren, indem wir u und v austauschen.

Daß das Verfahren insgesamt BC auf Bäumen korrekt löst, folgt nun mit einer einfach Induktion z.B. über die Größe des Graphens, wobei mit Hilfe obigen Argumentes im Induktionsschritt ein entsprechendes Austauschargument immer angewendet werden kann.

Tutoraufgabe T5 Wir betrachten jetzt die *gewichtete* Version von Vertex Cover. Gegeben ist ein Graph mit Knotengewichten (welche Zahlen sind). Außerdem ist wieder eine Zahl k gegeben. Die Frage ist, ob es ein Vertex Cover gibt, dessen *Gewicht* höchstens k ist.

Paßt folgende Anwendung zu diesem Problem? Sponsoren wollen spenden, aber einige Firmen sind gegenseitige Konkurrenten und würden nicht gleichzeitig mitmachen. Auf welche Sponsoren muß ich verzichten?

Ist die gewichtete Version von Vertex Cover immer noch *fixed parameter tractable*? Unterscheiden Sie, ob die Gewichte ganzzahlig oder rational sind.

Lösungsvorschlag
Zunächst sieht man leicht, daß man die Höhe des Suchbaums nicht mehr in k beschränken kann, wenn rationale oder negative Gewichte zugelassen werden (betrachte z.B. das Gewicht $1/n$ oder -1 pro Knoten).

Man kann allerdings ganzzahlige Gewichte kleiner oder gleich 0 einfach oBdA ausschließen, da es offensichtlich immer sinnvoll ist, Knoten ohne oder mit negativem Gewicht direkt in das VC aufzunehmen. Seien darum nun alle Gewichte ganzzahlig und oBdA größer 0. Wir können nun das klassische Verzweigungsverfahren über Kanten verwenden. Der Grad jedes Knotens im Suchbaum ist immer noch zwei, während die Höhe durch das Gesamtgewicht, also dem Parameter, beschränkt ist. Damit ist das Problem *fixed parameter tractable*.

Die Anwendung passt auf das Problem: Gebe jedem Sponsor einen Knoten in einem Graphen, dessen Gewicht dem Betrag seiner Förderung entspricht. Konkurrenten werden nun über eine Kante verbunden.

Ein VC C dieses Graphens entspricht nun einer Menge von Sponsoren, auf die wir verzichten müssen, damit der Graph $G[V \setminus C]$ kantenfrei, d.h. konfliktfrei, ist. Wenn wir das Gewicht dieses VC minimieren, dann minimieren wir auch gleichzeitig den Verlust, der durch den Verzicht auf diese Sponsoren entsteht.

Hausaufgabe H1

Entwerfen Sie einen möglichst effizienten Algorithmus, der VC für *tree connected cycles* löst! Dies sind Graphen, die nur aus disjunkten Kreisen bestehen, die durch Pfade baumförmig miteinander verbunden sind.

Hausaufgabe H2

Zeigen Sie, daß das folgende Problem *fixed parameter tractable* ist: Gegeben ist ein Graph. Gesucht ist ein minimales Vertex Cover. Parameter ist die *Größe des minimalen Vertex Covers*.

Was ist der Unterschied zum Problem aus der Vorlesung?

Was ist die Laufzeit Ihres Algorithmus?

Lösungsvorschlag

Der Unterschied besteht natürlich darin, daß der Parameter hier nicht Bestandteil der Eingabe, sondern vielmehr eine Eigenschaft des Eingabegraphen ist. Insbesondere ist seine Berechnung nicht trivial.

Wir können diese Widrigkeit jedoch leicht überwinden, indem wir zum Beispiel den Suchbaum-Algorithmus mit Laufzeit $O(2^k \cdot n)$ für $k = 1, 2, 3, \ldots$ laufen lassen, bis er schließlich eine Lösung findet. Aufgrund der Suchreihenfolge ist diese dann auch offensichtlich minimal.

Wenn k' die minimale Größe einer Lösung ist, dann ergibt sich durch die Aufrufe eine Gesamtlaufzeit von

$$O\big(\sum_{k=0}^{k'}(2^k \cdot n)\big) = O\big(2^{k'} \cdot n\big).$$