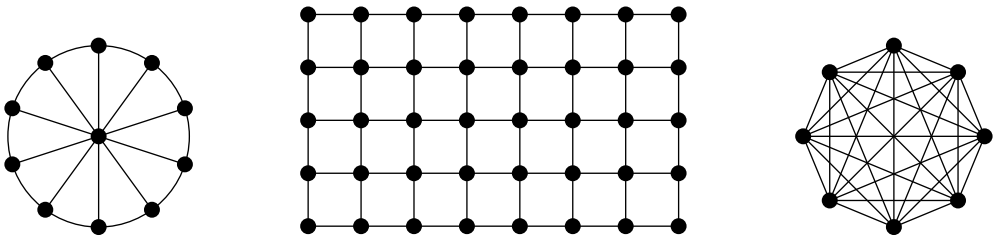
RWTH Aachen **WS 2008/09**
Lehrgebiet Theoretische Informatik Übungsblatt 3
Kneis—Langer—Richter—Rossmanith 6.11.2008

♡ Übung zur Vorlesung Parametrisierte Algorithmen ♡

Tutoraufgabe T9

Wenden Sie die Problemkernreduktion von Buss für Vertex Cover auf die folgenden Graphen an, wobei der Parameter stets k sei und $n \gg k$:

1. Ein Wagenrad mit n Knoten.
2. Ein $n \times m$-Gitter.
3. Eine $2k$-Clique.



Lösungsvorschlag

Im Fall $k = 1$ wird direkt eine einfache Ja- oder Nein-Instanz ausgegeben.

1. Wenn $k = 2$ ist, dann wird wg. $k \ll n$ eine triviale Nein-Instanz ausgegeben, da jeder Knoten mindestens Grad drei hat und es zu viele solcher Knoten gibt. Im Fall $k > 3$ wird nur die Narbe des Rades reduziert, übrig bleibt ein Kreis.
2. Wenn $k \leq 3$, wird wegen $k \ll n$ aus gleichen Gründen wieder eine triviale Nein-Instanz ausgegeben. Bei größeren k passiert nichts.
3. Jeder Knoten der $2k$-Clique hat Grad $2k - 1$ und es gibt mehr als k. Übrig bleibt daher eine triviale Nein-Instanz.

Tutoraufgabe T10

Entwerfen Sie einen Algorithmus, der einen Problemkern für das Problem Max-3-SAT berechnet:

Eingabe:	Eine aussagenlogische Formel F in 3-CNF und eine Zahl k
Parameter:	k
Frage:	Gibt es eine Belegung der Variablen von F, welche k Klauseln erfüllt?

Lösungsvorschlag
Sei m die Anzahl der Klauseln.
Wie man leicht sieht, erfüllt eine der beiden Variablenbelegungen $(0, 0, \ldots, 0)$ oder $(1, 1, \ldots, 1)$ immer mindestens $m/2$ Klauseln.
Wenn $k < m/2$, handelt es sich somit um eine triviale Ja-Instanz, und die Problemkernreduktion kann eine kleinere Trivialinstanz ausgeben, deren Größe jetzt geeignet in k beschränkt ist. Wenn hingegen $k > m/2$ gilt, ist natürlich auch $m < 2k$. Da wir Formeln in 3-CNF betrachten, enthält jede Klausel höchstens drei Variablen, womit die Gesamtzahl der Variablen durch $3m < 6k$ beschränkt ist. Um $3n < 6k$ Variablen zu beschreiben, benötigen wir nur etwa $\log(3n) < \log(6k)$ Bits. Damit ist die ursprüngliche Eingabelänge für die Turingmaschine insgesamt also bereits beschränkt durch $ck \log k$, wobei c eine geeignet gewählte Konstante ist.

Tutoraufgabe T11
Entwerfen Sie einen Problemkernalgorithmus für das Problem Maximum Independent Set auf planaren Graphen.

Lösungsvorschlag
Wie in einer früheren Übung erwähnt, gibt es in jedem planaren Graphen immer einen Knoten, der Grad höchstens fünf hat. Außerdem bleiben Graphen offensichtlich planar, wenn man Kanten oder Knoten entfernt.
Ein Problemkernalgorithmus für das Problem kann bei Eingabe (G, k) mit $G = (V, E)$ so vorgehen:

1. Wenn $k = 0$, gebe eine triviale „Ja“-Instanz Größe aus.
2. Wenn $V = \emptyset$, gebe das *ursprüngliche* (G, k) aus.
3. Wähle einen Knoten v von Grad höchstens fünf.
4. Lösche v und alle seine Nachbarn aus G.
5. Setze $k := k - 1$.
6. Gehe zu 2.

Die Korrektheit ist leicht einzusehen: Zunächst einmal ist klar, daß der Algorithmus nach spätestens $\min\{|V|, k\}$ Schleifendurchläufen terminiert, da jedesmal k minimiert wird oder wenigstens ein Knoten entfernt wird. Wenn der Algorithmus unter 2. abbricht, ist $k = 0$ und wir haben offensichtlich bereits k mal ein v gefunden. Diese so gewählten v bilden ein IS der Größe k. Ansonsten bricht der Algorithmus im dritten Schritt ab. Zu diesem Zeitpunkt ist die Schleife weniger als k mal durchlaufen worden, und in jedem Schritt wurden höchstens sechs Knoten entfernt. Damit ist $V \leq 6k$, und schon die Originalinstanz ist klein genug.

Hausaufgabe H5
Entwerfen Sie einen Algorithmus, der einen Problemkern für das Problem Maximum Satisfiability berechnet:

Eingabe:	Eine aussagenlogische Formel F in CNF und eine Zahl k
Parameter:	k
Frage:	Gibt es eine Belegung der Variablen von F, welche k Klauseln erfüllt?

Lösungsvorschlag

Wie schon in Tutoraufgabe T10 gesehen, können wir davon ausgehen, daß die Eingabeformel nur $2k$ Klauseln besitzt.Wir müssen nun nur noch zeigen, daß wir auch die Anzahl der Variablen beschränken können.

Zuerst wenden wir Vereinfachungsregeln an: Zwei Variablen x, y sind aquivalent, falls eine Klausel x genau dann enthält, wenn sie auch y enthält und eine Klausel $\overline{x}$ genau dann enthält, wenn sie auch $\overline{y}$ enthält. In diesem Fall können wir offensichtlich x auf true und y auf false setzen und erfüllen alle Klauseln die x oder y enthalten. Durch diese Vereinfachungsregel wird k reduziert und wir müssen nur noch einen Kernel für die übrigbleibende Formel angeben.

Somit erhalten wir eine Formel mit höchstens $2k$ Klauseln, die keine aquivalenten Variablen enthält. Da jede Variable in jeder der $2k$ Klauseln entweder positiv, negativ oder gar nicht vorkommen kann, gibt es somit höchstens 3^{2k} viele nichtaquivalente Variablen.

Unser Kernel besteht also aus einer Formel der Länge $2k$ mit höchstens 3^{2k} vielen Variablen. Die Eingabelänge ist damit natürlich auch durch eine Funktion von k beschränkt.

Hausaufgabe H6

Beweisen Sie, daß das folgende Problem in FPT liegt:

Eingabe:	Ein Graph G und eine Zahl k
Parameter:	k
Frage:	Kann man bis zu k Kanten löschen und hinzufügen, so daß ein Graph entsteht, der keinen induzierten Pfad mit drei Knoten enthält?

Lösungsvorschlag

Ein Algorithmus gehe bei Eingabe (G, k) so vor:

1. Betrachte alle $\binom{n}{3}$ Teilmengen von drei Knoten und teste, ob diese in G einen induzierten Pfad bilden.
2. Gibt es keinen solchen Pfad, antworte „Ja“.
3. Sei sonst $p = v_1v_2v_3$ solch ein induzierter Pfad, $e_1 = \{v_1, v_2\}$ und $e_2\{v_2, v_3\}$ die beiden Kanten und $e_3 = \{v_1, v_3\}$ eine neue Kante zwischen v_1 und v_3.
4. Wenn $k = 0$ ist, antworte „Nein“.
5. Teste rekursiv, ob $(G - e_1, k - 1)$, $(G - e_2, k - 1)$, oder $(G + e_3, k - 1)$ eine Lösung bilden.

Die Korrektheit des Algorithmus läßt sich per Induktion mit Hilfe des folgenden Arguments leicht zeigen: Jeder induzierte Pfad der Länge drei muß in einer Lösung durch eine Modifikation des Graphens „zerstört“ werden, sei es durch das Entfernen einer der beiden Pfadkanten oder durch das Hinzufügen einer weiteren Kante, so daß der Pfad kein Pfad mehr ist (sondern ein Dreieck). Wenn es eine Lösung gibt, dann führt also einer dieser drei Möglichkeiten im Branching auch zu einer Lösung.

Da die Höhe des Suchbaums durch k beschränkt ist, und jeder seiner Knoten nur drei Kinder hat, hat er Größe höchstens $O(3^k)$. In jedem Knoten des Suchbaums wird nun nur polynomieller Aufwand getrieben: So werden z.B. alle $\binom{n}{3} \leq n^3$ Teilmengen betrachtet und der Graph leicht modifiziert. Die Gesamtlaufzeit des Algorithmus ist also $O(3^k \cdot poly(n))$ und damit das obige Problem in FPT.