Parametrisierte Algorithmen
Wintersemester 2022/2023
Thomas Bläsius & Marcus Wilhelm

Karlsruher Institut für Technologie

Lösung zum Übungsblatt 1

Aufgabe 1: Verschiedenes

Teilaufgabe (a) Ja, denn $2^{\Theta(n)}$ kann beispielsweise für $2^{2n} = 4^n$ stehen, was asymptotisch echt schneller wächst als 2^n. Umgekehrt beinhaltet $2^{\Theta(n)}$ auch $2^{n/2} = (\sqrt{2})^n \approx 1{,}41^n$, was langsamer wächst als 2^n. Man kann also nicht sagen, dass $2^{\Theta(n)}$ langsamer oder schneller wächst als $\Theta(2^n)$. Allerdings ist $2^{\Theta(n)}$ ungenauer.

Teilaufgabe (b) Da T ein Baum ist gilt $m = n - 1 = n_1 + n_3 - 1$. Außerdem ist die Summe der Knotengrade $2m$, also $2m = n_1 + 3n_3$. Ersetzt man m in der zweiten Gleichung durch die erste, so erhält man

$$\begin{aligned} 2(n_1 + n_3 - 1) &= n_1 + 3n_3, \\ 2n_1 + 2n_3 - 2 &= n_1 + 3n_3, \\ n_1 - 2 &= n_3. \end{aligned}$$

Teilaufgabe (c) Angenommen ein Problem P mit Parameter k ist für konstantes k NP-vollständig. Dann würde ein FPT-Algorithmus für dieses parametrisierte Problem P = NP implizieren. Umgekehrt würde P = NP implizieren, dass P in polynomieller Zeit (und damit insbesondere in FPT Zeit) lösbar ist. Man muss also nur parametrisierte Probleme finden, die für einen konstanten Parameter schon NP-schwer sind. Hier ein paar Beispiele:

- k-SAT (jede Klausel enthält k Literale) mit dem Parameter k ist für $k = 3$ NP-vollständig.
- k-FÄRBUNG in Graphen mit dem Parameter k ist für $k = 3$ schon NP-vollständig.
- HAMILTON PFAD in Graphen mit dem Maximalgrad k als Parameter ist für $k = 3$ schon NP-vollständig.

Aufgabe 2: k-DOMINATING SET auf Bäumen

Dieses Problem lässt sich sowohl mit einem Greedy-Algorithmus Lösen, als auch mit einem allgemeinerem Ansatz.

Greedy. Wir betrachten den Baum als an einem beliebigen Knoten gewurzelt und betrachten iterativ ein Blatt l mit maximaler Tiefe und den v Knoten mit Distanz k zu l auf dem Pfad zur Wurzel. Dann ist es immer sicher, v zur Lösung hinzuzufügen, da l abgedeckt werden muss und v unter den Knoten die l abdecken derjeniege ist, der am meisten andere Knoten abdeckt. Diese Reduktionsregel kann angewandt werden, bis ein minimales Dominating Set gefunden wurde.

Dynamisches Programm. Alternativ kann das Problem auch über ein dynamisches Programm gelöst werden. Diese Lösung ist allgemeiner und kann auch verwendet werden um gewichtete Varianten von k-Dominating Set zu lösen. Die grobe Idee ist erneut den Baum als gewurzelt zu betrachten und beginnend bei den Blättern für jeden an einem Knoten gewurzelten Teilbaum Teillösungen zu berechnen.

Die Teillösungen für einen Teilbaum werden durch zwei Parameter h_1 und h_2 unterschieden, wobei h_1 die Tiefe des höchsten Knoten im Dominating Set angibt und h_2 die Tiefe des tiefsten nicht abgedeckten Knotens. Der Parameter h_1 kann die Werte -1 (falls kein Knoten im DS ist) oder $0, \dots, k$ annehmen (falls ein Knoten mit der entsprechenden Distanz zur Wurzel im DS ist) und h_2 kann die Werte -1 (falls alle Knoten im Teilbaum abgedeckt sind) oder $0, \dots, k-1$ (falls ein Knoten mit der entsprechenden Distanz zur Wurzel noch von außerhalb des Teilbaums abgedeckt werden muss) annehmen. Für jede Teillösung wird die Anzahl von Knoten im DS gespeichert.

Für jedes Blatt gibt es zwei Teillösungen: eine in der das Blatt zum DS hinzugefügt wird ($h_1 = 0$, $h_2 = -1$) und eine in der das Blatt nicht zum DS hinzugefügt wird ($h_1 = -1$, $h_2 = 1$). Falls die Teillösungen aller Kinder eines Knotens v bereits berechnet wurden, dann ist es möglich in die Teillösungen für v zu berechnen. Wir skizzieren im folgenden wie dis in $O(\deg v)$ Zeit möglich ist. Hierzu iterieren wir über alle möglichen Wert x_1, x_2 für h_1 und h_2. Für jedes Kind c suchen wir die günstigste Teillösung für die beiden Fälle, dass c den einen Knoten in Ebene höchstens $x_1 - 1$ zum DS hinzufügt, oder dass ein anderes Kind von v dies tut. Hierbei muss natürlich sichergestellt werden, dass die h_1 und h_2 Werte der verschiedenen Teillösungen zu x_1 und x_2 passen. Die beste Teillösung für v und die x_1, x_2 kann dann gefunden werden, indem die Lösungsgrößen verglichen werden für die $O(\deg v)$ Varianten welches Kind von v einen DS Knoten auf ebene $x_1 - 1$ hat.

Da pro Knoten also $O(\deg v)$ Zeit benötigt wird, ist die Gesamtlaufzeit in $O(m) = O(n)$.

Aufgabe 3: IS und Clique auf regulären Graphen

Independent Set. Für einen Knoten mit Grad d branchen wir auf den $d + 1$ möglichen Entscheidungen den Knoten selbst oder einen seiner Nachbarn zur Lösung hinzuzufügen. In jedem Branch löschen wir den gewählten Knoten und verringern k um 1. Dies liefert einen Rekursionsbaum mit $O((d+1)^k$ Blättern und somit einen FPT-Algorithmus für Parameter $k + r$.

Clique. Falls $k > r + 1$, liegt eine Nein-Instanz vor. Andernfalls können wir für einen Knoten v überprüfen ob seine Nachbarschaft eine $(k-1)$-Clique bildet, indem wir über die weniger als $\binom{r}{k-1} \in O(r^k)$ $k-1$-elementigen Teilmengen von $N(v)$ iterieren und in $O(k^2)$ Zeit prüfen, ob diese eine Clique bilden. Wenn dies für alle Knoten $v \in V$ überprüft wird, ergibt sich eine Gesamtlaufzeit in $O(r^k k^2 n)$.

Aufgabe 4: Punkte und Geraden

Das Problem zu entscheiden, ob es eine k-Geradenüberdeckung gibt, lässt sich ähnlich wie Vertex Cover mit Kernbildung lösen.

Zuerst stellen wir fest, dass nur Geraden betrachtet werden müssen, die durch mindestenz zwei Punkte gehen, da Geraden durch nur einen Punkt ersetzt werden können ohne die Lösung größer zu machen. Außerdem gilt, dass falls eine Gerade mit p Punkten nicht in der Überdeckung ist, dann müssen stattdessen p andere Geraden (eine durch jeden Punkt) in der Überdeckung enthalten sein.

Somit können wir folgende Reduktionsregeln formulieren:

Reduktionsregel 1: Falls es eine Kante mit mehr als k Punkten gibt, dann muss diese in der Lösung enthalten sein. Lösche die Kante und die Punkte und verringere k um 1.

Reduktionsregel 2: Gibt es mehr als k^2 Punkte, aber nur noch Geraden, die höchstens k Punkte enthalten, dann kann es keine Geradenüberdeckung mit k Geraden geben. Gib in diesem Fall eine konstant Große Nein-Instanz zurück.

Falls beide Reduktionsregeln nicht anwendbar sind, gibt es also höchstens k^2 Punkte und k^4 Geraden, also einen Problemkern dessen Größe nur von k abhängt. Ein FPT-Algorithmus für Geradenüberdeckung kann diese Kernbildung anwenden und anschließend das Problem per brute-force lösen.