Institut für Informatik
Prof. Dr. J. Rothe

Sommersemester 2024
Übungsblatt 10

Lösungsvorschläge
Kryptokomplexität II

Bearbeitungszeit: 11. Juni bis 21. Juni
Verantwortlich: Roman Zorn

Aufgabe 1: Multiprocessor Job Scheduling

Betrachten Sie das in der Vorlesung definierte Entscheidungsproblem Multiprocessor Job Scheduling.

	Multiprocessor Job Scheduling
Gegeben:	Eine Liste von Jobs $J = (j_1, j_2, \ldots, j_k)$, wobei Job j_i die Länge l_i hat, m Prozessoren und eine Schranke t.
Frage:	Ist es möglich, alle Jobs ohne Überlappungen auf die Prozessoren aufzuteilen, so dass die Gesamtlaufzeit alle Jobs abzuarbeiten höchstens t ist?

(a) ▶ Betrachten Sie die Liste von Jobs $J = (j_1, j_2, j_3, j_4, j_5, j_6, j_7)$ mit den Joblängen $l_1 = 2,\ l_2 = 1,\ l_3 = 4,\ l_4 = 4,\ l_5 = 1, l_6 = 4$ und $l_7 = 5$. Es gibt $m = 3$ Prozessoren. Entscheiden Sie jeweils für die Schranken $t_1 = 6, t_2 = 7$ und $t_3 = 8$, ob die Instanzen $\langle J, m, t_i \rangle$ in Multiprocessor Job Scheduling liegen. Falls ja, geben Sie eine Belegung der Prozessoren an, falls nein, begründen Sie.

(b) ▶ Zeigen Sie, dass das Problem Multiprocessor Job Scheduling in NP liegt.

(c) ▶ Nehmen Sie an, man könnte die Jobs beliebig zerteilen, das heißt, ein Job der Länge 7 könnte zum Beispiel in zwei neue Jobs der Länge 1 und 6 aufgeteilt werden. Was ändert sich bezüglich der Komplexität des Problems?

Lösungsvorschlag:

(a) Für $t_1 = 6$ ist es eine NEIN-Instanz. Die Gesamtlaufzeit aller Jobs ist mit 21 schon mehr als 3 Prozessoren in $t_1 = 6$ Zeiteinheiten schaffen könnten.

Für $t_2 = 7$ ist es eine NEIN-Instanz. Zwar haben die Jobs insgesamt eine Laufzeit von 21, was mit 3 Prozessoren prinzipiell in $t_2 = 7$ Zeiteinheiten möglich ist, aber wir haben 4 Jobs, die jeweils mehr als $7/2$ Zeiteinheiten benötigen. Man kann nie zwei von diesen Jobs auf einen Prozessor legen, da dieser dann schon über 7 Zeiteinheiten braucht. Bei 3 Prozessoren und 4 solchen Jobs ist es aber nicht möglich keine zwei Jobs auf den selben

Prozessor zu legen.

Für $t_3 = 8$ gibt es beispielsweise folgenden Schedule:

Prozessor 1	j_1	j_1	j_2	j_7	j_7	j_7	j_7	j_7
Prozessor 2	j_3	j_3	j_3	j_3	j_4	j_4	j_4	j_4
Prozessor 3	j_5	j_6	j_6	j_6	j_6			

(b) Es genügt einen nichtdeterministischen Polynomialzeit-Algorithmus anzugeben:

- Rate (nichtdeterministisch) eine Belegung der Prozessoren.
- Prüfe ob ein Prozessor mehr als t Zeiteinheiten benötigt.
- Wenn nicht, haben wir eine Lösung gefunden und akzeptieren.

(c) Eine Instanz ist trivialerweise genau dann eine JA-Instanz, wenn die Gesamtlaufzeit aller Jobs maximal $m \cdot t$ (also der Kapazität der Prozessoren) ist. Man kann die Jobs nämlich immer so zerstückeln, dass alle Prozessoren gleichzeitig fertig werden.

Aufgabe 2: Dominierende Menge und Knotenüberdeckung

Sei G ein ungerichteter Graph.

(a) ▶ Zeigen Sie, dass eine Knotenüberdeckung von G im Allgemeinen keine dominierende Menge in G sein muss.

(b) ▶ Ist das Komplement einer minimalen dominierenden Menge in G immer auch eine dominierende Menge?

(c) ▶ Zeigen Sie, dass eine dominierende Menge in G im Allgemeinen keine Knotenüberdeckung von G sein muss.

(d) ▶ Geben Sie eine VERTEXCOVER-Instanz (H, k') an, bei der H zusammenhängend ist und $|V(H)| \geq 3$ gilt, und begründen Sie mit ihr, weshalb die folgende Reduktion h von VERTEXCOVER auf DOMINATINGSET fehlschlägt:

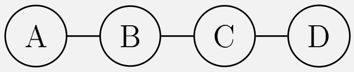
$$h(G, k) = (G, k).$$

Lösungsvorschlag:

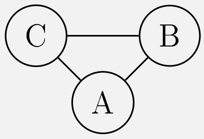
(a) Der Graph, der aus einem Knoten besteht, hat die leere Menge als Knotenüberdeckung. Jedoch ist die leere Menge keine dominierende Menge, da in diesem Fall für jeden Knoten ein benachbarter Knoten in der dominierenden Menge existieren muss. Somit müssen isolierte Knoten immer in einer dominierenden Menge sein.

(b) Wie es oben erklärt wurde, müssen isolierte Knoten immer in einer dominierenden Menge sein. Das Komplement von einer dominierenden Menge, die alle isolierte Knoten enthält, enthält keine der isolierten Knoten von G, also kann es keine dominierende Menge sein.

(c) Betrachte den folgenden Pfad mit 4 Knoten. Dann bilden die beiden Knoten A, D eine dominierende Menge, aber keine Knotenüberdeckung, da keine Knoten der mittleren Kante $\{B, C\}$ in der Menge enthalten sind.



(d) Betrachte einen Kreis mit 3 Knoten und $k' = 1$. Diese Instanz ist nicht in VertexCover, da mindestens eine Kante nicht abgedeckt wird. Jedoch ist sie eine Ja-Instanz für DominatingSet.



Aufgabe 3: VertexCover $\leq_m^p$ DominatingSet

► Zeigen Sie mit einer direkten Reduktion

VertexCover $\leq_m^p$ DominatingSet.

Lösungsvorschlag: Sei $G = (V, E)$ ein Graph und sei (G, k) eine VertexCover-Instanz. Wir bilden einen neuen Graph $G' = (V', E')$, indem G' den Graph G als Teilgraphen und für jede Kante $\{x, y\} \in E$ einen neuen Knoten enthält, der mit x und y verbunden ist. Sei i die Anzahl der isolierten Knoten in G'. Wir setzen $k' = k + i$. Wir zeigen, dass eine dominierende Menge der Größe höchstens k' in G' genau dann existiert, wenn eine Knotenüberdeckung der Größe höchstens k in G existiert.

$\Leftarrow$: Sei C eine Knotenüberdeckung der Größe höchstens k und sei I die Menge der isolierten Knoten in G'. Dann ist $C \cup I$ eine dominierende Menge der Größe höchstens k'. Sei $x \in V' \setminus (C \cup I)$. Wenn $x \in V$, dann gibt es eine Kante $\{x, y\} \in E$. Da C eine Knotenüberdeckung ist, gilt $y \in C$. Wenn x ein neuer Knoten ist, dann gibt es in G eine dazugehörige Kante, die von C überdeckt wird und somit wird x dominiert. Also ist $C \cup I$ eine dominierende Menge.

$\Rightarrow$: Sei D eine dominierende Menge der Größe höchstens k'. Wir können $D \subseteq V$ annehmen. Falls ein neu erzeugter Knoten in D liegt, können wir ihn durch einen Nachbarn ersetzen. D bleibt weiterhin eine dominierende Menge der Größe höchstens k'. Sei $\{x, y\} \in E$. Dann muss x oder y in D sein, andernfalls würde der für diese Kante neu erzeugte Knoten nicht dominiert werden, da dieser nur mit x und y verbunden ist. Folglich ist $D \setminus I$ eine Knotenüberdeckung.

Aufgabe 4: Knotenfärbungen

(a) ► Sei $d \geq 3$ ein maximaler Knotengrad von dem zusammenhängenden Graph G. Kann G immer mit d oder weniger Farben gefärbt werden?

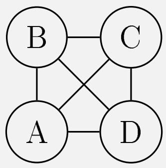
(b) ► Geben Sie ein Beispiel von einem Graph an, der vierfärbbar, aber nicht dreifärbbar ist.

(c) ► Zeigen Sie mit einer Many-One-Reduktion

$$3\text{-Colorability} \leq_m^p 4\text{-Colorability}.$$

Lösungsvorschlag:

(a) Nein – vollständige Graphen brauchen mindestens $d+1$ Farben, um gefärbt zu werden.

(b) 

(c) Für die Reduktion zeigen wir, dass jede Instanz I von 3-Colorability in eine Instanz I' für 4-Colorability überführt werden kann, sodass

$$I \in 3\text{-Colorability} \iff I' \in 4\text{-Colorability}.$$

Sei $I = (V, E)$ eine Instanz für 3-Colorability. Wir erzeugen in Polynomialzeit eine Instanz $I' = (V', E')$ für 4-Colorability, wobei $V' = V \cup \{x\}$ einen zusätzlichen Knoten x enthält und $E' = E \cup \{\{x, v\} \mid v \in V\}$ enthält zusätzlich je eine Kante von jedem Knoten aus V zu dem neuen Knoten x.

$\Rightarrow$: Sei I dreifärbbar. Dann ist V in I' mit der gleichen Färbung und wir verwenden die vierte Farbe für den neuen Knoten x. Also ist I' vierfärbbar.

$\Leftarrow$: Sei I' vierfärbbar. Die Farbe von dem neuen Knoten x kann nur einmal für ihn verwendet werden, da er mit jedem anderen Knoten verbunden ist. Somit ist I dreifärbbar.