Grundzüge der Theoretischen Informatik, WS 21/22: Musterlösung zum 12. Übungsblatt

Julian Dörfler

Aufgabe A12.1 (Feedback Vertex Set) *(4 Punkte)*
Sei $G = (V, E)$ ein *gerichteter* Graph. Dann ist ein k-*Feedback Vertex Set* (kurz k-FVS) eine Menge $F \subseteq V$ von k Knoten, so dass G nach Entfernen aller Knoten aus F azyklisch[1] ist.

Wir definieren nun

$$\mathsf{FVS} = \{(G, k) \mid G \text{ enthält ein } k\text{-FVS}\}$$

Zeigen Sie, dass FVS NP-vollständig ist.
Hinweis: Reduzieren Sie von VertexCover.

Lösung A12.1 (Feedback Vertex Set) Wir beweisen zuerst $\mathsf{FVS} \in \mathsf{NP}$, indem wir einen Polynomialzeit-Verifizierer angeben.

> Bei Eingabe (G, k) mit Zertifikat c prüfe, ob c eine Menge F von k Knoten aus G kodiert. Danach entferne alle Knoten aus F aus G und prüfe ob der resultierende Graph azyklisch ist. Wenn all dies erfüllt ist akzeptiere, ansonsten verwerfe.

All dies ist offensichtlich in Polynomialzeit möglich, also ist $\mathsf{FVS} \in \mathsf{NP}$.

Um nun zu zeigen, dass FVS NP-schwer ist, reduzieren wir VertexCover auf FVS.

> Bei Eingabe (G, k) mit $G = (V, E)$ konstruiere einen Graphen $G' = (V, E')$ mit
>
> $$E' = \{(u, v), (v, u) \mid \{u, v\} \in E\}$$
>
> Danach gib (G', k) aus.

Diese Reduktion ist offensichtlich in Polynomialzeit möglich.

Falls nun $(G, k) \in \mathsf{VertexCover}$, dann enthält G ein Vertexcover C der Größe k. Wir behaupten nun, dass C ebenfalls ein k-Feedback Vertex Set für G' ist. Angenommen G' ohne C enthielte noch einen Zyklus, dann enthält G' ohne C insbesondere noch eine Kante $(u, v) \in E'$. Dies würde aber implizieren, dass $\{u, v\} \in E$, dies ist aber ein Widerspruch dazu, dass C ein Vertexcover für G ist. Also ist $(G', k) \in \mathsf{FVS}$.

Falls nun $(G', k) \in \mathsf{FVS}$, dann existiert ein k-FVS F für G'. Wir behaupten nun, dass F ebenfalls ein Vertexcover für G ist: Sei $\{u, v\} \in E$ beliebig. Dann ist $\{(u, v), (v, u)\}$ ein gerichteter Kreis in G'. Daher enthält F entweder u oder v. F ist also ein k-Vertexcover für G. Also ist $(G, k) \in \mathsf{VertexCover}$.

Da VertexCover NP-schwer ist, ist also auch FVS NP-schwer.

Da FVS sowohl in NP, als auch NP-schwer ist, ist FVS NP-vollständig.

---

[1] d.h. G enthält keinen gerichteten Kreis mehr

Aufgabe A12.2 (Rucksackproblem) *(4 Punkte)*
Beim *Rucksackproblem* KNAPSACK ist das Ziel, gegeben ein Tupel von Elementen $M = (m_1, \dots, m_n)$ mit Gewichten $W = (w_1, \dots, w_n)$ und Werten $C = (c_1, \dots, c_n)$, sowie zwei natürlichen Zahlen U und L, herauszufinden, ob es eine Teilmenge $M' = (m_{i_1}, \dots, m_{i_\ell})$ von M gibt, so dass

$$\sum_{j=1}^{\ell} w_{i_j} \leq U \text{ und } \sum_{j=1}^{\ell} c_{i_j} \geq L\,.$$

In diesem Fall ist die Instanz (M, W, C, U, L) lösbar.

Zeigen Sie, dass das folgende Problem NP-vollständig ist: Wir definieren nun das Problem:

$$\mathsf{KNAPSACK} = \{(M, W, C, U, L) \mid (M, W, C, U, L) \text{ ist lösbar}\}\,.$$

(a) Zeigen Sie, dass KNAPSACK in Zeit polynomiell in L und n von einer DTM entscheidbar ist.

(b) Zeigen Sie, dass KNAPSACK NP-vollständig ist.
Hinweis: Reduzieren Sie von SubsetSum.

(c) Wieso widersprechen sich die beiden vorhergehenden Teilaufgaben nicht?

Lösung A12.2 (Rucksackproblem)

(a) Wir benutzen dynamische Programmierung:

Gegeben (M, W, C, U, L) erstelle eine Tabelle T, mit $L + 1$ Zeilen und $n + 1$ Spalten (jeweils 0-indiziert). Der Eintrag $T_{i,j}$ in der i-ten Zeile und j-ten Spalte soll hierbei den minimalen Wert enthalten, den $\sum_{k \in M'} w_k$ annehmen kann für Teilmengen $M' \subseteq \{1, \dots, j\}$ mit $\sum_{k \in M'} c_k \geq i$. Die Einträge $T_{i,j}$ werden wie folgt berechnet:

$$T_{i,j} = \begin{cases} 0 & \text{falls } j = 0 \text{ und } i = 0 \\ \infty & \text{falls } j = 0 \text{ und } i > 0 \\ \min\{T_{i,j-1}, T_{i-c_j,j-1} + w_j\} & \text{falls } j > 0 \text{ und } c_j \leq i \\ \min\{T_{i,j-1}, w_j\} & \text{sonst} \end{cases}$$

Nun ist $T_{L,n} \leq U$, genau dann, wenn $(M, W, C, U, L) \in \mathsf{KNAPSACK}$.

Wir verwenden in diesem Algorithmus ∞ symbolisch. Dies wird entweder extra behandelt in den arithmetischen Operationen oder durch eine große Zahl (mindestens $U + 1$) ausgetauscht. Da zur Berechnung von $T_{i,j}$ nur auf $T_{i',j'}$ mit $i' \leq i$ und $j' \leq j$ zugegriffen werden muss, ist die Laufzeit polynomiell[2] in $L \cdot n$. Für die Korrektheit des Algorithmus gibt es nun zwei Fälle für eine Menge $M' \subseteq \{1, \dots, j\}$:

[2]In der Praxis wäre dies direkt $O(L \cdot n)$, jedoch müssen wir bei einer DTM um die Kodierung der Tabelle Gedanken machen. Dies ist aber maximal ein polynomieller Overhead.

$j \in M'$: Für $M'' = M' \setminus \{j\}$ gilt $\sum_{k \in M''} c_k = \sum_{k \in M'} c_k - c_j \geq i - c_j$ und $\sum_{k \in M'} w_k = \sum_{k \in M''} w_k + w_j$, somit ist M'' enthalten in den Mengen über die $T_{i-c_j,j-1}$ definiert wurde.

$j \notin M'$: M' selbst ist nun schon in den Mengen enthalten über denen $T_{i,j-1}$ definiert wurde.

Es gibt auch eine alternative Lösung mit Laufzeit polynomiell in U und n: Wir benutzen erneut dynamische Programmierung:

Gegeben (M, W, C, U, L) erstelle eine Tabelle T, mit $U+1$ Zeilen und $n+1$ Spalten (jeweils 0-indiziert). Der Eintrag $T_{i,j}$ in der i-ten Zeile und j-ten Spalte soll hierbei den maximalen Wert enthalten, den $\sum_{k \in M'} c_k$ annehmen kann für Teilmengen $M' \subseteq \{1, \ldots, j\}$ mit $\sum_{k \in M'} w_k \leq i$. Die Einträge $T_{i,j}$ werden wie folgt berechnet:

$$T_{i,j} = \begin{cases} 0 & \text{falls } j = 0 \\ \max\{T_{i,j-1}, T_{i-w_j,j-1} + c_j\} & \text{falls } j > 0 \text{ und } c_j \leq i \\ T_{i,j-1} & \text{sonst} \end{cases}$$

Nun ist $T_{U,n} \geq L$, genau dann, wenn $(M, W, C, U, L) \in \mathsf{KNAPSACK}$.

Da zur Berechnung von $T_{i,j}$ nur auf $T_{i',j'}$ mit $i' \leq i$ und $j' \leq j$ zugegriffen werden muss, ist die Laufzeit polynomiell[3] in $U \cdot n$. Für die Korrektheit des Algorithmus gibt es nun zwei Fälle für eine Menge $M' \subseteq \{1, \ldots, j\}$:

$j \in M'$: Für $M'' = M' \setminus \{j\}$ gilt $\sum_{k \in M''} w_k = \sum_{k \in M'} w_k - w_j \leq i - w_j$ und $\sum_{k \in M'} c_k = \sum_{k \in M''} c_k + c_j$, somit ist M'' enthalten in den Mengen über die $T_{i-w_j,j-1}$ definiert wurde.

$j \notin M'$: M' selbst ist nun schon in den Mengen enthalten über denen $T_{i,j-1}$ definiert wurde.

(b) Wir reduzieren von $\mathsf{SubsetSum}$: Sei $(x_1, \ldots, x_n, b)$ eine Instanz von $\mathsf{SubsetSum}$. Wir definieren $m_i = w_i = c_i = x_i$ für alle $i \in \{1, \ldots, n\}$ und setzen $U = L = b$.

Für eine Teilmenge $M' = (m_{i_1}, \ldots, m_{i_\ell})$ von M sind Bedingungen

$$\sum_{j=1}^{\ell} w_{i_j} \leq U \text{ und } \sum_{j=1}^{\ell} c_{i_j} \geq L\,.$$

mit dieser Wahl nun direkt äquivalent zu

$$\sum_{j=1}^{\ell} x_{i_j} = b$$

[3] In der Praxis wäre dies direkt $O(U \cdot n)$, jedoch müssen wir bei einer DTM um die Kodierung der Tabelle Gedanken machen. Dies ist aber maximal ein polynomieller Overhead.

Es folgt also sofort, dass

$$(x_1, \dots, x_n, b) \in \mathsf{SubsetSum} \Leftrightarrow (M, W, C, U, L) \in \mathsf{KNAPSACK}\,.$$

Die Reduktion ist offensichtlich in polynomieller Zeit durchführbar und da $\mathsf{SubsetSum}$ NP-schwer ist, ist also $\mathsf{KNAPSACK}$ auch NP-schwer. Um Inklusion in NP zu zeigen, kann eine NTM einfach in Polynomialzeit eine Teilmenge $M' = (m_{i_1}, \dots, m_{i_\ell}$ von M raten und dann verifizieren ob

$$\sum_{j=1}^{\ell} w_{i_j} \leq U \text{ und } \sum_{j=1}^{\ell} c_{i_j} \geq L\,.$$

gelten.

Da $\mathsf{KNAPSACK}$ nun also NP-schwer und in NP enthalten ist, ist es ebenfalls NP-vollständig.

(c) Die beiden Teilaufgaben widersprechen sich nicht, da L (und auch alle anderen Zahlen der Eingabe) in Binärdarstellung gegeben werden. Somit kann L exponentiell in der Eingabelänge sein, womit der Algorithmus aus Teilaufgabe (a) exponentielle Laufzeit in der Eingabelänge hat.

Aufgabe A12.3 (3-Färbung) *(4 Punkte)*
Eine k-Knotenfärbung eines Graphen $G = (V, E)$ ist eine Funktion $c : V \to \{1, 2, \dots, k\}$. Wir nennen eine solche Färbung gültig, wenn für alle $\{u, v\} \in E$ gilt $c(u) \neq c(v)$, also keine zwei benachbarten Knoten die gleiche Farbe zugewiesen bekommen. Wir sagen ein Graph G ist k-knotenfärbbar, wenn eine gültige k-Knotenfärbung c für G existiert.

Wir definieren

$$k\text{-}\mathsf{Col} = \{G \mid G \text{ ist } k\text{-knotenfärbbar}\}$$

wobei wir im weiteren Verlauf der Aufgabe aber nur noch 3-Col betrachten.

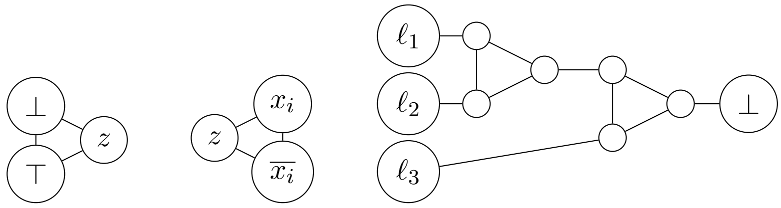


Abbildung 1: Die Gadgetgraphen G_1, G_2 und G_3

(a) Zeigen Sie 3-$\mathsf{Col} \in \mathsf{NP}$.

(b) *(0 Punkte)* Machen Sie sich klar, dass in einem Dreieck, sobald zwei Knoten eine Farbe haben, der dritte Knoten eindeutig bestimmt ist. Somit können wir Farben eindeutig nach den drei Knoten aus G_1 als $\top, \perp$ und z benennen.

(c) Zeigen Sie, wie man mehrere Kopien von G_2 mit einer Kopie von G_1 verbinden kann, so dass alle Knoten mit Beschriftung x_i und $\overline{x_i}$ in allen gültigen Färbungen jeweils genau einen Knoten mit Farbe $\perp$ und einen mit Farbe $\top$ erhalten. Weiterhin sollte es für jede mögliche Kombination $\perp$ und $\top$ auf die x_i und $\overline{x_i}$ zu verteilen eine gültige Färbung geben.

(d) Zeigen Sie, dass G_3 genau dann eine gültige 3-Knotenfärbung besitzt, wenn mindestens einer der Knoten ℓ_1, ℓ_2 und ℓ_3 eine andere Farbe zugewiesen bekommt als der Knoten $\perp$.

(e) Zeigen Sie hiermit nun, dass 3-Col NP-schwer ist.

Hinweis: Reduzieren Sie von 3-SAT.

Lösung A12.3 (3-Färbung)

(a) Wir zeigen dies mit einer NTM:

Bei Eingabe $G = (V, E)$ rate eine 3-Knotenfärbung c für G. Dies können wir zum Beispiel durch Raten einer Liste der Länge $|V|$ mit Einträgen aus $\{1, 2, 3\}$ machen. Nun akzeptiere, wenn für alle Kanten $\{u, v\} \in E$ gilt $c(u) \neq c(v)$ und verwerfe sonst.

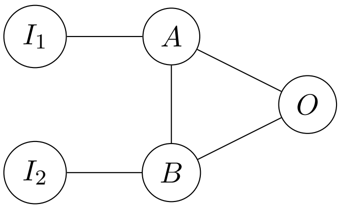
Diese NTM ist offensichtlich Polynomialzeit beschränkt , es gilt also 3-Col $\in$ NP.

(b) Allgemein: um eine k-Clique gültig zu k-knotenfärben müssen alle Knoten paarweise verschiedene Farben zugewiesen bekommen, Sobald also $k-1$ Knoten schon eine Farbe haben, so ist die Farbe eindeutig bestimmt.

Ein Dreieck ist nun einfach eine 3-Clique.

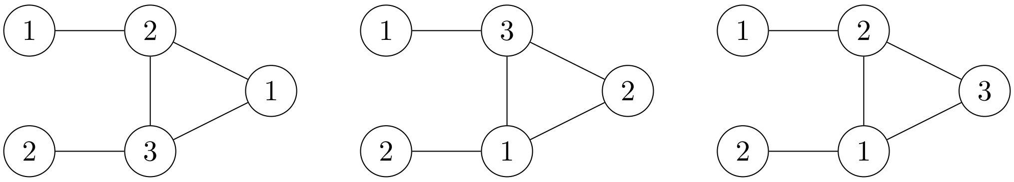
(c) Wir verbinden G_1 mit n Kopien von G_2 indem wir alle Knoten die mit z beschriftet sind vereinigen. Da z immer die Farbe z hat, müssen x_i und $\overline{x_i}$ nun beide jeweils eine unterschiedliche der Farben $\perp$ und $\top$ zugewiesen bekommen. Insbesondere haben die Knoten x_i und $\overline{x_i}$ keine weiteren Kanten ausser dem Dreieck aus G_2, d.h. beide Optionen sind möglich.

(d) Zum analysieren von G_3 schauen wir uns den folgenden Teilgraphen H an:

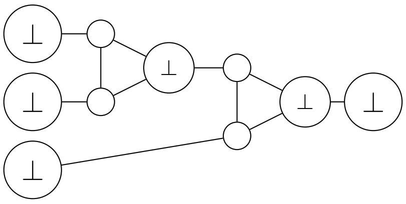


Nehmen wir an I_1 und I_2 sind mit der gleichen Farbe (oBdA mit der Farbe 1) gefärbt. Dann haben A and B die Farben 2 und 3 in irgendeiner Reihenfolge. Somit muss O ebenfalls die Farbe 1, also die gleiche wie I_1 und I_2 haben.

Nehmen wir nun an, dass I_1 und I_2 unterschiedliche Farben haben (oBdA 1 und 2), dann gibt es gültige Färbungen für beliebige Farben am Knoten O. Dies beweisen wir durch explizites angeben der Färbungen:



Wenden wir dies nun doppelt auf G_3 an, wenn ℓ_1, ℓ_2 und ℓ_3 alle die Farbe $\perp$ haben, sehen wir dass die folgenden Farben erzwungen sind:



Nun haben aber zwei benachbarte Knoten die gleiche Farbe, es kann also keine gültige Färbung geben. Falls nun aber mindestens einer von ℓ_1, ℓ_2 und ℓ_3 eine andere Farbe als $\perp$ erhält, können wir die mittleren Knoten passend wählen um eine gültige Färbung zu erhalten.

(e) Wir reduzieren $\mathsf{3\text{-}SAT}$ auf $\mathsf{3\text{-}Col}$.

Gegeben eine 3-CNF ϕ mit n Variablen und m Klauseln. Wir vereinigen nun eine Kopie von G_1 mit n Kopien von G_2 und m Kopien von G_3, alle Knoten die mit $\perp$ beschriftet sind, werden hierzu zu einem vereinigt, das gleiche gilt für alle Knoten die mit z beschriftet sind. Die Knoten mit Namen ℓ_1, ℓ_2 und ℓ_3 in der Kopie von G_3, die zur Klausel $\ell_1 \vee \ell_2 \vee \ell_3$ in ϕ gehört, werden mit den entsprechenden Knoten aus G_2 vereinigt. Letztlich geben wir diesen kombinierten Graph G aus.

Diese Reduktion ist offensichtlich in Polynomialzeit möglich.

Sei nun $\phi \in \mathsf{3\text{-}SAT}$. Dann gibt es eine erfüllende Belegung $x_1, \dots, x_n$ für ϕ. Färbe nun alle Knoten x_i mit $\top$ und Knoten $\overline{x_i}$ mit $\perp$ für Variablen x_i die mit 1 belegt werden und färbe nun alle Knoten x_i mit $\perp$ und Knoten $\overline{x_i}$ mit $\top$ für Variablen x_i die mit 0 belegt werden.

Es gibt nun nach Teilaufgabe (c) eine Möglichkeit die Kopien der Teilgraphen G_1 und G_2 zu färben und nach Teilaufgabe (d) einen Weg all die Teilgraphen G_3 korrekt zu 3-färben, der gesamte Graph ist also 3-färbbar.

Sei nun G 3-färbbar. Dann wähle als Belegung $x_i = 1$ wenn der Knoten x_i die gleiche Farbe wie der Knoten $\top$ bekommt und $x_i = 0$ sonst. Nun muss in jeder Kopie des Teilgraphen G_3 nach Teilaufgabe (d) mindestens einer der Knoten ℓ_1, ℓ_2, ℓ_3 entweder mit z oder mit $\top$ gefärbt sein. Nach Teilaufgabe (c) ist hierbei die einzige Möglichkeit $\top$, somit erfüllt die Belegung alle Klauseln, es gilt also $\phi \in \textsf{3-SAT}$.

Da $\textsf{3-SAT}$ $\textsf{NP}$-schwer ist, ist also $\textsf{3-Col}$ ebenfalls $\textsf{NP}$-schwer.

Aufgabe A12.4 (co-NP) *(4 Punkte)*
Wir definieren

$$\textsf{coNP} = \{L \mid \overline{L} \in \textsf{NP}\}\,.$$

(a) Zeigen Sie: $\textsf{co-NP} \neq \textsf{NP} \Rightarrow \textsf{P} \neq \textsf{NP}$.

(b) Eine Sprache L ist co-NP-*vollständig* wenn

$$L \in \textsf{co-NP} \wedge \forall L' \in \textsf{co-NP} : L' \leq_P L\,.$$

Zeigen Sie, dass die folgende Sprache co-NP-vollständig ist:

$$\textsf{TAUT} = \{F \mid F \text{ ist eine tautologische Formel in disjunktiver Normalform}\}\,.$$

Lösung A12.4 (co-NP)

(a) Wir zeigen die Kontraposition. Sei also $\textsf{P} = \textsf{NP}$. Aus $L \in \textsf{P} \Leftrightarrow \overline{L} \in \textsf{P}$ folgt

$$\textsf{co-NP} = \{L \mid \overline{L} \in \textsf{NP}\} = \{L \mid \overline{L} \in \textsf{P}\} = \{L \mid L \in \textsf{P}\} = \{L \mid L \in \textsf{NP}\} = \textsf{NP}\,.$$

(b) Wir zeigen zunächst, dass $\overline{\textsf{SAT}}$ co-NP-vollständig ist:
Sei $L \in \textsf{co-NP}$. Dann ist $\overline{L} \in \textsf{NP}$. Aus der NP-Vollständigkeit von SAT folgt, dass $\overline{L} \leq_P \textsf{SAT}$, und daher auch $L \leq_P \overline{\textsf{SAT}}$. Außerdem ist $\overline{\textsf{SAT}} \in \textsf{co-NP}$, da $\textsf{SAT} \in \textsf{NP}$. Nun muss gezeigt werden, dass (1) $\overline{\textsf{SAT}} \leq_P \textsf{TAUT}$ und dass (2) $\textsf{TAUT} \leq_P \overline{\textsf{SAT}}$. Damit folgt aus der co-NP-Vollständigkeit von $\overline{\textsf{SAT}}$, die co-NP-Vollständigkeit von TAUT.

(1) Wir geben die Reduktion an: Gegeben ein x, testen wir ob x eine CNF-Formel kodiert. Wenn nein, geben wir die trivial wahre DNF-Formel $(x_1 \vee \neg x_1)$ zurück. Ansonsten sei Φ die von x kodierte Formel. Wir negieren die Formel Φ und erhalten durch mehrmaliges benutzen des Satzes von de Morgan eine DNF-Formel (tausche $\wedge$ und $\vee$ und negiere alle Literale), die wir zurückgeben.

Sei nun $x \in \overline{\textsf{SAT}}$. Wenn x keine Formel kodiert, gibt die Reduktion $(x_1 \vee \neg x_1) \in \textsf{TAUT}$ zurück. Ansonsten kodiert x eine unerfüllbare Formel F. In diesem Fall ist das Komplement von F eine Tautologie, welche wir als DNF-Formel zurückgeben.

Sei $x \notin \overline{\textsf{SAT}}$, d.h. $x \in \textsf{SAT}$. Dann kodiert x eine erfüllbare CNF-Formel F. Dann ist die Negation von F aber keine Tautologie.

Da die Konstruktion (tausche $\wedge$ und $\vee$ und negiere alle Literale) in polynomieller Zeit durchführbar ist, folgt $\overline{\textsf{SAT}} \leq_P \textsf{TAUT}$.

(2) Analog zu (1): Wir negieren eine tautologische DNF-Formel und erhalten eine unerfüllbare CNF-Formel.