Grundzüge der Theoretischen Informatik, WS 21/22: Musterlösung zum 13. Präsenzblatt

Julian Dörfler

Aufgabe P13.1 (Sprachen)
Geben Sie für jedes der folgenden Entscheidungsprobleme entweder einen Beweis der NP-Vollständigkeit oder einen Polynomialzeitalgorithmus an:

(a) $(n-2)$-Clique: Gegeben ein Graph G mit n Knoten, hat G eine Clique mit $n-2$ Knoten?

(b) $n/2$-Clique: Gegeben ein Graph G mit n Knoten, hat G eine Clique mit $n/2$ Knoten?

(c) LargeSubsetSum: Gegeben eine Menge S natürlicher Zahlen, gibt es eine Teilmenge $U \subseteq S$, sodass $\frac{1}{4}\sum_{x\in S} x \leq \sum_{x\in U} x \leq \frac{3}{4}\sum_{x\in S} x$?

(d) SubGI: Gegeben zwei Graphen $G = (V_G, E_G)$ und $H = (V_H, E_H)$, ist H zu einem Subgraph von G isomorph? Das heißt: Gibt es eine Knotenmenge $V_S \subseteq V_G$ und eine Bijektion $\varphi : V_H \to V_S$ so, dass $\{u,v\} \in E_H \iff \{\varphi(u), \varphi(v)\} \in E_G$ für alle $u, v \in V_H$?

(e) LinSep: Gegeben eine Menge P mit n Punkten an ganzzahligen Koordinaten in der Ebene, einige rot, einige grün gefärbt, gibt es eine Gerade, die beide Farben trennt?

(f) SAT': Gegeben eine 3-CNF mit m Klauseln, gibt es eine Belegung die $\geq \frac{1}{2}m$ Klauseln erfüllt?

(g) SAT'': Gegeben eine 3-CNF mit m Klauseln, gibt es eine Belegung die $\geq \frac{2}{3}m$ Klauseln erfüllt?

Lösung P13.1 (Sprachen)

(a) $(n-2)$-Clique ist in P. Der folgende Polynomialzeitalgorithmus entscheidet die Sprache:

Auf Eingabe $G = (V, E)$ prüfe für jedes Paar $(u, v) \in V \times V$, ob der Graph ohne u und v eine Clique bildet.

(b) $n/2$-Clique ist NP-vollständig.

$n/2$-*Clique* $\in$ NP : Wir geben eine Polynomialzeit NTM an, die das Problem entscheidet:

Gegeben ein Graph G mit n Knoten, rate $\frac{n}{2}$ paarweise verschiedene Knoten und prüfe ob diese eine Clique formen.

$n/2$-Clique ist NP*-schwer* : Wir verwenden die folgende Reduktion von Clique:

Gegeben ein Graph $G = (V, E)$ und $k \in \mathbb{N}$. Wir geben $G' = (V', E')$ aus, wobei wir G' aus G konstruieren indem wir zuerst $2n - 2k$ Knoten hinzufügen, die jeweils mit allen anderen Knoten verbunden sind und danach n vollständig isolierte Knoten einfügen.

Diese Reduktion ist offensichtlich in Polynomialzeit berechenbar.

Allgemein gilt: $|V'| = 4n - 2k$.

Sei nun $(G, k) \in$ Clique. Dann existieren k paarweise verschiedene Knoten $v_1, \ldots, v_k \in V$, so dass diese eine Clique formen. Nun formen diese Knoten, zusammen mit den $2n - 2k$ zuerst eingefügten Knoten eine Clique der Größe $2n - k$. Da G' genau $4n - 2k$ Knoten hat, gilt also $G' \in n/2$-Clique.

Sei nun $G' \in n/2$-Clique. Dann existieren $2n-k$ paarweise verschiedene Knoten $v_1, \ldots, v_{2n-k} \in V$, so dass diese eine Clique formen. Da die isolierten Knoten können niemals Teil dieser Clique sein können, sind maximal $2n - 2k$ Knoten zusätzlich eingefügt. Umgekehrt müssen also mindestens k dieser Knoten ebenfalls in G vorkommen. Die Kanten zwischen diesen sind die selben wie in G', somit formen diese eine Clique der Größe mindestens k. Nun formen diese Knoten, zusammen mit den $2n - 2k$ zuerst eingefügten Knoten eine Clique der Größe $2n - k$, es gilt also $(G, k) \in$ Clique.

Da Clique NP-schwer ist, ist also auch $n/2$-Clique NP-schwer.

(c) LargeSubsetSum ist in P. Der folgende Polynomialzeitalgorithmus entscheidet die Sprache:

Bei Eingabe S finde das maximale Element $y \in S$. Falls $y \leq \frac{3}{4} \sum_{x \in S} x$, gib "existiert" aus, ansonsten "existiert nicht".

Bleibt die Korrektheit zu zeigen: Sei $S \in$ LargeSubsetSum. Dann existiert ein U mit

$$\frac{1}{4} \sum_{x \in S} x \leq \sum_{x \in U} x \leq \frac{3}{4} \sum_{x \in S} x \,.$$

Daraus folgt ebenfalls

$$\frac{1}{4} \sum_{x \in S} x \leq \sum_{x \in S \setminus U} x \leq \frac{3}{4} \sum_{x \in S} x \,.$$

Insbesondere kann also weder U, noch $S \setminus U$ ein Element y enthalten, so dass $y > \frac{3}{4} \sum_{x \in S} x$. Der Algorithmus gibt also das korrekte Ergebnis aus.

Wir nehmen nun umgekehrt an, für das maximale Element $y \in S$ gilt $y \leq \frac{3}{4} \sum_{x \in S} x$. Hierbei unterscheiden wir zwei Fälle:

$y \geq \frac{1}{4}\sum_{x\in S} x$: Dann zeigt $U = \{y\}$ direkt $S \in \mathsf{LargeSubsetSum}$.

$y < \frac{1}{4}\sum_{x\in S} x$: Da alle Elemente in S kleiner als $\frac{1}{4}\sum_{x\in S} x$ sind können wir beliebige Elemente zu U hinzufügen, bis die Summe groß genug ist ohne die obere Schranke zu überschreiten. Es gilt also ebenfalls $S \in \mathsf{LargeSubsetSum}$.

(d) SubGI ist NP-vollständig.

SubGI $\in$ NP : Wir geben eine Polynomialzeit NTM an, die das Problem entscheidet:

Gegeben ein Graph G mit n Knoten, rate $\frac{n}{2}$ paarweise verschiedene Knoten und prüfe ob diese eine Clique formen.

SubGI *ist* NP-*schwer* : Wir verwenden die folgende Reduktion von Clique:

Gegeben (G, k). Wir geben (G, C_k) aus, wobei C_k der vollständige Graph mit k Knoten ist.

Diese Reduktion ist offensichtlich in Polynomialzeit berechenbar.

Sei nun $(G, k) \in \mathsf{Clique}$ mit $G = (V_G, E_G)$. Dann existiert eine Clique $C \subseteq V_G$, so dass für alle $\{u, v\} \subseteq C$ gilt $\{u, v\} \in E_G$. Jede Bijektion $\varphi : C \to C_k$ erfüllt nun $\{u, v\} \in E_G \Leftrightarrow \{\varphi(u), \varphi(v)\} \in E_{C_k}$, da C_k der vollständige Graph mit k Knoten ist und somit alle Kanten existieren. Somit ist $(G, C_!) \in \mathsf{SubGI}$.

Sei nun $(G, C_k) \in \mathsf{SubGI}$. Dann existiert eine Teilmenge $V_S \in V_{C_k}$ und eine Bijektion $\varphi : V_S \to V_{C_k}$, so dass $\{u, v\} \in E_G \Leftrightarrow \{\varphi(u), \varphi(v)\} \in E_{C_k}$. Da für alle $\{u, v\} \subseteq V_{C_k}$ gilt und φ bijektiv ist $\{u, v\} \in E_{C_k}$, gilt ebenfalls $\{\varphi^{-1}(u), \varphi^{-1}(v)\} \in E_G$. Da alle diese $\varphi^{-1}(u), \varphi^{-1}(v)$ paarweise verschieden sind, bildet die Menge $\varphi^{-1}(C_k)$ also eine k-Clique in G und somit $(G, k) \in \mathsf{Clique}$.

Da Clique NP-schwer ist, ist also auch SubGI NP-schwer.

(e) LinSep ist in P. Wir interpretieren folgendermaßen: eine Gerade trennt die beiden Farben, wenn zwei Punkte unterschiedlicher Farbe, die nicht auf der Geraden liegen, niemals auf der gleichen Seite der Gerade liegen.

Der folgende Polynomialzeitalgorithmus entscheidet nun die Sprache:

Bei Eingabe P, bezeichne die Menge aller roten Punkte als R und die aller grünen Punkte als G. Falls eine dieser Mengen leer ist, existiert eine trennende Gerade offensichtlich. Ansonsten teste für alle Paare $(r, g) \in R \times G$, ob die Gerade durch r und g die Farben trennt.

Wir geben hier keinen vollständigen Beweis der Korrektheit, sondern lediglich eine Intuition: Wenn es eine Gerade g gibt, die die beiden Farben trennt, so gibt es einen Punkt in R, der am nächsten an g liegt. Wir können nun g so verschieben, dass g diesen Punkt enthält und jeder Punkt auf derselben Seite der Gerade bleibt auf der er war. Schließlich können wir diese Gerade nun so lange rotieren, bis ebenfalls ein Punkt aus G auf g liegt, hierbei kann sich der verwendete Punkt in R aber ändern.

(f) SAT' ist in P. Wir zeigen $\mathsf{SAT}' = \{\varphi \mid \varphi \text{ ist in 3-CNF}\}$, woraus $\mathsf{SAT}' \in \mathsf{P}$ direkt folgt.

Sei φ eine 3-CNF Formel in m Klauseln. Betrachte eine beliebige Belegung x. Falls x schon $m/2$ Klauseln erfüllt, so sind wir fertig. Falls aber x nun $< m/2$ Klauseln erfüllt, definierien wir $\overline{x}$ als die komplementäre Belegung zu x, d.h. für jede Variable $\overline{x}_i$ gilt $\overline{x}_i = 1 - x_i$. Nun erfüllt $\overline{x}$ aber mindestens alle Klauseln die von x nicht erfüllt wurden, also mindestens $m/2$. Somit erfüllt also entweder x oder $\overline{x}$ mindestens $m/2$ Klauseln.

(g) SAT'' ist NP-vollständig.

$\mathsf{SAT}'' \in \mathsf{NP}$: Wir geben eine Polynomialzeit NTM an, die das Problem entscheidet:

> Gegeben eine 3-CNF φ mit m Klauseln, rate eine Belegung der Variablen und prüfe ob diese $\geq \frac{2}{3}m$ Klauseln erfüllt.

SAT'' *ist* NP*-schwer* : Wir verwenden die folgende Reduktion von SAT:

> Gegeben eine 3-CNF φ mit m Klauseln. Gebe φ' zurück, wobei wir φ' aus φ konstruieren, indem wir die Klauseln (x_1) und $(\neg x_1)$ jeweils m mal hinzufügen.

Diese Reduktion ist offensichtlich in Polynomialzeit berechenbar.

Zuerst bemerken wir: unabhänging von der gewählten Belegung werden genau m der $2m$ hinzugefügten Klauseln erfüllt sein. Falls eine Belegung x also genau k Klauseln in φ erfüllt, so erfüllt x genau $k + m$ Klauseln in φ'.

Es gilt nun für eine 3-CNF φ mit m Klauseln $\varphi \in \mathsf{SAT}$ genau dann, wenn eine Belegung x existiert, so dass x genau m Klauseln erfüllt. Dies ist genau dann der Fall, wenn x genau $2m$ Klauseln in φ' erfüllt, also $\varphi' \in \mathsf{SAT}''$ ist, da φ' $3m$ Klauseln enthält.

Da SAT NP-schwer ist, ist also auch SAT'' NP-schwer.

Aufgabe P13.2 (Reguläre Graphen)

Für einen ungerichteten Graphen $G = (V, E)$ definieren wir den Grad $\delta(v)$ eines Knoten $v \in V$ als die Anzahl an Nachbarn von v in G. Formal:

$$\delta(v) := |\{u \in V \mid \{u, v\} \in E\}|$$

Wir nennen einen ungerichteten Graphen G nun regulär, wenn alle Knoten in G den gleichen Grad haben.

Zeigen Sie, dass das Problem

$$\mathsf{RegClique} = \{(G, k) \mid G \text{ ist ein regulärer Graph und enthält eine } k\text{-Clique}\}$$

NP-vollständig ist.

Hinweis: Reduzieren Sie von Clique und kopieren Sie den Graphen in der Reduktion mehrfach.

Lösung P13.2 (Reguläre Graphen) Wie im Beweis, dass Clique $\in$ NP raten wir auch hier die Knoten der k-Clique und prüfen, ob diese eine Clique bilden. Zusätzlich prüfen wir noch, ob der Eingabe-Graph G regulär ist. All dies ist in Polynomialzeit möglich, also ist RegClique $\in$ NP.

Nun zeigen wir noch die NP-Schwere, indem wir von Clique reduzieren. Hierzu reduzieren wir den Graphen $G = (V, E)$ und Ganzzahl k auf (G', k), wobei G' wie folgt konstruiert wird. Wir schreiben dafür $V = \{v_1, \ldots, v_n\}$.

- Sei d der maximal Grad eines Knoten in G. Wir fügen in G' nun d Kopien von G ein.
- Für jeden Knoten v_i vom Grad d' fügen wir nun $d - d'$ Knoten $w_{i,1}, w_{i,2}, \ldots, w_{i,d-d'}$ ein und verbinden diese jeweils mit allen d Kopien von v_i in G'.

Nun ist G' regulär, da zu jeder Kopie eines Knoten v_i vom Grad d' genau $d - d'$ Kanten hinzugefügt wurden. Diese haben also nun genau Grad d. Alle eingefügten Knoten $w_{i,j}$ werden direkt mit d Kopien von v_i verbunden, haben also auch Grad d.

Bleibt also zu zeigen, dass G' genau dann eine k-Clique enthält, wenn G eine k-Clique enthält.

$k = 1$ Eine 1-Clique ist nur ein einzelner Knoten. Da G' nur Knoten enthält, wenn G Knoten enthält, ist dieser Fall trivial.

$k = 2$ Die Existenz einer 2-Clique ist äquivalent zur Existenz einer Kante. Da G' nur Kanten enthält, wenn G Kanten enthält, ist dieser Fall ebenfalls trivial.

$k \geq 3$ Hier enthält jede k-Clique nur Knoten, die Teil von Dreiecken sind. Alle Knoten $w_{i,j}$ sind nun aber niemals Teil eines Dreiecks, also können diese niemals Teil einer k-Clique sein. Somit muss jede k-Clique in G' vollständig in einer der Kopien von G in G' liegen, womit die Aussage gezeigt ist.

Da Clique NP-vollständig ist, ist also auch RegClique NP-vollständig.