12. Aufgabenblatt

(Besprechung in den Tutorien 23.01.2023–27.01.2023)

Aufgabe 1. Clique and Half Clique

Eine Clique der Größe k in einem ungerichteten Graphen $G = (V, E)$ ist eine Knotenmenge $V' \subseteq V$ mit $|V'| = k$ und $\{u, v\} \in E$ für alle $u, v \in V'$ mit $u \neq v$.

Beweisen Sie, dass das Problem Half Clique NP-schwer ist.

Half Clique
Eingabe: Ein ungerichteter Graph $G = (V, E)$.
Frage: Gibt es eine Clique der Größe $|V|/2$ in G?

——————Lösungsskizze——————

Wir zeigen eine Reduktion Clique $\leq_m^p$ Half CLique.

Sei $G = (V, E)$ ein Graph mit $|V| = n$ und $k \in \mathbb{N}$. Wir konstruieren den Graph $H := (V', E')$ wobei

$$V' := V \dot\cup \{1, \ldots, n\}$$

und

$$E' := E \cup \{\{v, i\} \mid v \in V, 1 \leq i \leq n - k\} \cup \{\{i, j\} \mid 1 \leq i, j \leq n - k \wedge i \neq j\}.$$

Dann ist $(G, k) \mapsto H$ total und in Polynomzeit berechenbar. (Streng genommen $\langle (G, k) \rangle \mapsto \langle H \rangle$. Wir ignorieren hier die Kodierung der Instanzen.)

Wir zeigen nun, dass $(G, k) \in$ Clique $\iff H \in$ Half Clique.

"$\Rightarrow$": Sei $S \subseteq V$ mit $|S| = k$ eine Clique in G. Dann besteht H aus $2n$ Knoten und $S \cup \{1, \ldots, n - k\}$ ist eine Clique der Größe n in H.

"$\Leftarrow$": Sei $S \subseteq V'$ mit $|S| = n$ eine Clique in H. Da alle Knoten $i > n - k$ in H isoliert sind, enthält S mindestens k Knoten aus V. Diese bilden dann eine Clique in G.

Da Clique NP-vollständig ist, folgt, dass Half Clique NP-schwer ist.

Aufgabe 2. Polynomzeitreduktion (Klausuraufgabe 2012)

Betrachten Sie die beiden folgenden Probleme:

Clique
Eingabe: Ein ungerichteter Graph $G = (V, E)$ und $k \in \mathbb{N}$.
Frage: Gibt es eine Clique der Größe k in G

Multicolored Clique
Eingabe: Ein ungerichteter Graph $G = (V, E)$, ein $k \in \mathbb{N}$ und eine Funktion $c \colon V \to \{1, 2, \ldots, k\}$.
Frage: Gibt es eine Clique V' der Größe k in G, sodass für alle $i \in \{1, 2, \ldots, k\}$ ein $v \in V'$ existiert mit $c(v) = i$?

Hinweis: Intuitiv ist Multicolored Clique die Aufgabe, eine Clique V' der Größe k zu finden, wobei es für jede „Farbe“ $i \in \{1, 2, \ldots, k\}$ *genau* einen Knoten mit Farbe i in V' gibt.

Betrachten Sie die folgende Reduktion von Clique auf Multicolored Clique.

Reduktion: Sei der Graph $G = (V, E)$ und $k \in \mathbb{N}$ eine Eingabe für Clique. Wir konstruieren einen Graph $G' = (V', E')$ zusammen mit einer Färbung $c : V' \to \{1, 2, \ldots, k\}$ in 3 Schritten:

1. Für jeden Knoten $v \in V$ führe k Knoten $v^1, v^2, \ldots, v^k$ in G' ein. Setze $c(v^i) = i$ für alle $i \in \{1, 2, \ldots, k\}$.

2. Verbinde für jede Kante $\{u, v\} \in E$ und für alle $1 \leq i, j \leq k$ die Knoten v^i und u^j in G' durch eine Kante.

3. Verbinde für alle $1 \leq i < j \leq k$ und Knoten $v \in V$ die Knoten v^i und v^j mit einer Kante.

Wir definieren nun die Polynomzeitreduktion f durch $f(G, k) = (G', c, k)$.

Überprüfen Sie die obige Reduktion auf Korrektheit und korrigieren Sie diese gegebenenfalls. Beweisen Sie anschließend die Korrektheit der (eventuell korrigierten) Reduktion, d. h. zeigen Sie

$$\forall (G, k) : (G, k) \in \textsc{Clique} \Leftrightarrow f(G, k) \in \textsc{Multicolored Clique}.$$

——————Lösungsskizze——————

Fehler: Die Knoten $v^1, v^2, \ldots, v^k$ bilden immer eine multicolored Clique für beliebigen Knoten v. Der Fehler kann behoben werden, indem der 3. Schritt der Konstruktion weggelassen wird.

Beweis der Korrektheit: Sei $(G = (V, E), k)$ eine Instanz für $\textsc{Clique}$ und sei $G' = (V', E')$ mit Färbung $c : V' \to \{1, \ldots, k\}$ der Graph, welcher durch die obige korrigierte Reduktion aus (G, k) konstruiert wird.

Wir zeigen $(G, k) \in \textsc{Clique} \iff (G', k, c) \in \textsc{Multicolored Clique}$.

"$\Rightarrow$": Sei $H \subseteq V$ eine Clique der Größe k in G mit $H = \{v_1, \ldots, v_k\}$. Wir zeigen, dass die Knotenmenge $\{v_1^1, \ldots, v_k^k\} \subseteq V'$ eine multicolored Clique in G' ist. Nach Konstruktion (Schritt 1) gilt $c(v_i^i) = i$ für alle $i \in \{1, \ldots, k\}$. Da H eine Clique ist, gilt für alle $i, j \in \{1, \ldots, k\}, i \neq j$, dass $\{v_i, v_j\} \in E$ und damit (nach Schritt 2 der Konstruktion) auch $\{v_i^i, v_j^j\} \in E'$.

"$\Leftarrow$": Sei $\{v_1', \ldots, v_k'\}$ eine multicolored Clique der Größe k in G' mit $c(v_i') = i$ für alle $i \in \{1, \ldots, k\}$. Für jeden Knoten v_i' sei $t(v_i') \in V$ der korrespondierende Knoten aus G, d. h. $t(v_i')$ ist der Knoten, für den v_i' in Schritt 1 konstruiert wurde. Wir zeigen, dass die Menge $H := \{t(v_1'), \ldots, t(v_k')\}$ eine Clique der Größe k in G bildet: Da $\{v_i', v_j'\} \in E'$ gilt nach Schritt 2 auch $\{t(v_i'), t(v_j')\} \in E$ für alle $i, j \in \{1, \ldots, k\}, i \neq j$ und damit ist H eine Clique in G.

————————————————

Aufgabe 3. Transitivität von Reduktionen (Klausuraufgabe SoSe 2017)

Im Folgenden seien Σ und Π zwei endliche Alphabete. Betrachten Sie die folgenden beiden Reduktionstypen.

Definition 1. Eine Sprache $A \subseteq \Sigma^*$ heißt *linearzeit-reduzierbar* bzw. *quadratzeit-reduzierbar* auf eine Sprache $B \subseteq \Pi^*$ (in Zeichen $A \leq_m^\ell B$ bzw. $A \leq_m^q B$) genau dann, wenn es eine totale, in *linearer* Zeit ($O(|x|)$ für jedes $x \in \Sigma^*$) bzw. *quadratischer* Zeit ($O(|x|^2)$ für jedes $x \in \Sigma^*$) berechenbare Funktion $f : \Sigma^* \to \Pi^*$ gibt, sodass gilt:

$$\forall x \in \Sigma^* : x \in A \Leftrightarrow f(x) \in B.$$

1. Begründen Sie die Transitivität für einen der beiden Reduktionstypen.

2. Argumentieren Sie kurz (in 2-3 Sätzen), warum Transitivität im Kontext des Vollständigkeitskonzepts eine sinnvolle Eigenschaft für Reduktionen ist.

——————Lösungsskizze——————

1. Behauptung: Linearzeit-Reduzierbarkeit ist transitiv.

   Beweis. Seien $\Sigma_1, \Sigma_2, \Sigma_3$ endliche Alphabete. Seien $A \subseteq \Sigma_1^*, B \subseteq \Sigma_2^*, C \subseteq \Sigma_3^*$. Angenommen, $A \leq_m^\ell B$ und $B \leq_m^\ell C$, das heißt es gibt Abbildungen f_1 und f_2, so dass
$$\forall x \in \Sigma_1^* : x \in A \Leftrightarrow f_1(x) \in B \tag{1}$$
   und
$$\forall x \in \Sigma_2^* : x \in B \Leftrightarrow f_2(x) \in C. \tag{2}$$
   Wir zeigen, dass dann $A \leq_m^\ell C$ gilt. Genauer: Die Komposition $f := f_2 \circ f_1$ reduziert A in linearer Zeit auf C. Aus den Korrektheitseigenschaften (1) und (2) folgt direkt die Korrektheit von f:
$$\forall x \in \Sigma_1^* : x \in A \Leftrightarrow f_1(x) \in B \Leftrightarrow f(x) = f_2(f_1(x)) \in C \tag{3}$$

   Es bleibt zu zeigen, dass f in Linearzeit berechnet werden kann. Wir wissen, dass f_1 und f_2 jeweils in linearer Zeit berechnet werden können, das heißt es existieren Konstanten $c_1, c_2 \in \mathbb{N}$, so dass für alle $x \in \Sigma_1^*$ in höchstens $c_1 \cdot |x|$ Berechnungsschritten die Ausgabe $f_1(x)$ berechnet werden kann. Analog kann für alle $x \in \Sigma_2^*$ in höchstens $c_2 \cdot |x|$ Berechnungsschritten die Ausgabe $f_2(x)$ berechnet werden.

   Da eine Turingmaschine pro Berechnungsschritt höchstens ein Zeichen schreiben kann, ist die Laufzeit eine direkte Obergrenze für die Ausgabelänge. Wir wissen also, dass $|f_1(x)| \leq c_1 \cdot |x|$. Somit kann die Komposition $f = f_2 \circ f_1$ in höchstens $c_2 \cdot c_1 \cdot |x| + c_1 \cdot |x| = (c_2+1) \cdot c_1 \cdot |x|$ Schritten berechnet werden, also insbesondere in linearer Zeit ($(c_2+1) \cdot c_1$ ist ein konstanter Faktor für Konstanten c_1, c_2).

2. Eine Sprache A ist *vollständig* für eine Komplexitätsklasse, wenn sie selbst in der Klasse enthalten ist und alle Sprachen aus dieser Klasse auf A reduzierbar sind. Mit Transitivität genügt es nun, eine vollständige Sprache B zu kennen und diese auf A zu reduzieren, um zu zeigen, dass A vollständig ist. Denn wenn alle Sprachen der Klasse auf B reduzierbar sind und außerdem B auf A reduziert wird, dann ist gezeigt, dass alle Sprachen auf A reduzierbar sind. Anstatt also direkt zu zeigen, dass *alle* Sprachen aus der Klasse auf A reduziert werden können, genügt eine Reduktion von *einer einzigen* vollständigen Sprache auf A. Das spart Arbeit.

Aufgabe 4. Erfüllende Belegung Finden

Aus der Vorlesung ist folgendes NP-vollständiges Problem bekannt:

> **KNF-SAT**
> **Eingabe:** Eine aussagenlogische Formel F in konjunktiver Normalform.
> **Frage:** Ist F erfüllbar?

Beweisen Sie folgende Aussage: Wenn P = NP, dann gibt es einen Polynomzeitalgorithmus, der für eine gegebene erfüllbare Formel in KNF eine erfüllende Belegung findet.

——————Lösungsskizze——————

Anmerkung: Die Aussage ist nicht so trivial, wie sie zunächst erscheinen mag. Wenn P = NP, dann gibt es zwar einen Polynomzeitalgorithmus, der entscheidet, ob eine gegebene Formel erfüllbar ist, daraus folgt aber nicht unmittelbar, dass dieser Algorithmus auch eine erfüllende Belegung liefert.

Wir nehmen an, dass P = NP. Da KNF-SAT in NP ist, folgt, dass KNF-SAT polynomzeitlösbar ist, also, dass ein Algorithmus (eine DTM) $\mathcal{A}$ existiert, der das Problem in Zeit $O(p(n))$ entscheidet, wobei p ein Polynom ist und n die Länge der Kodierung einer

Eingabeformel bezeichnet. Wir geben nun einen Algorithmus an, der für eine erfüllbare Formel F in KNF eine erfüllende Belegung findet.

Seien $x_1, \ldots, x_k$ die in F auftretenden Variablen. Sei $F_0 \coloneqq F$. Für $i = 0, \ldots, k-1$ macht der Algorithmus nun Folgendes:

Sei F_i^1 die Formel, die man erhält, indem man in F_i jede Klausel, die das Literal x_{i+1} enthält, löscht, und das Literal $\overline{x_{i+1}}$ aus jeder Klauser löscht. Sei F_i^0 die Formel, die man erhält, indem man in F_i das Literal x_{i+1} aus jeder Klausel löscht und jede Klausel, die $\overline{x_{i+1}}$ enthält, löscht. (Die Formel F_i^1 entpricht dem Fall, dass man x_{i+1} mit 'wahr' belegt, und F_i^0 dem Fall, dass man x_{i+1} mit 'falsch' belegt.) Benutze nun $\mathcal{A}$, um zu bestimmen, ob F_i^1 erfüllbar ist. Wenn ja, dann setze $F_{i+1} \coloneqq F_i^1$ und belege x_{i+1} mit 'wahr'. Wenn nicht, dann setze $F_{i+1} \coloneqq F_i^0$ und belege x_{i+1} mit 'falsch'. Am Ende wird die Belegung der Variablen ausgegeben.

Die Laufzeit dieses Algorithmus ist in $O(k \cdot (n + p(n)) + k)$, also polynomiell in der Eingabegröße n.

Es bleibt zu zeigen, dass der Algorithmus korrekt ist. Wir beweisen zunächst folgende Behauptung: Die Formel F_i is erfüllbar für alle $i \in \{0, \ldots, k\}$. Für $i = 0$ gilt die Behauptung nach Voraussetzung. Sei nun F_i erfüllbar. Fall 1: Es gibt eine erfüllende Belegung für F_i, die x_{i+1} mit 'wahr' belegt. Dann ist $F_{i+1} = F_i^1$ und diese Belegung erfüllt dann auch F_{i+1}. Fall 2: Jede erfüllende Belegung für F_i setzt x_{i+1} auf 'falsch'. Dann ist $F_{i+1} = F_i^0$ auch erfüllbar.

Wir beweisen nun, dass die ausgegebene Belegung α jede Formel F_i mit $i \in \{0, \ldots, k\}$ erfüllt. Da $F = F_0$, folgt daraus die Korrektheit des Algorithmus. Die Formel F_k enthält keine Klauseln mehr und ist somit immer trivial erfüllt. Angenommen, α erfüllt die Formel F_i. Fall 1: $\alpha(x_i)$ ist 'wahr'. Dann erfüllt α jede Klausel in F_{i-1}, die x_i enthält. Da α nach Voraussetzung F_i erfüllt, sind auch alle Klauseln, die nicht x_i enthalten von α erfüllt. Daher ist α eine erfüllende Belegung für F_{i-1}. Fall 2: analog...
