Prof. T. Nipkow
M. Eberl, J. Rädle, S. Wimmer
Technische Universität München
KEINE ABGABE

Einführung in die theoretische Informatik

Sommersemester 2019 – Übungsblatt Lösungsskizze 11

AUFGABE 11.1. (*Wichtige Begriffe*) Stufe A

Überprüfen Sie, dass Sie die folgenden Begriffe korrekt definieren können.

- Polynomielle Reduktion $\leq_p$
- SAT
- 3-KNF-SAT

AUFGABE 11.2. (*Quiz*) Stufe B

Diskutieren Sie die folgenden Aussagen:

(a) Für jede Sprache, die von einer NTM in polynomieller Zeit akzeptiert werden kann, existiert eine DTM, die ebenfalls die Sprache in polynomieller Zeit akzeptiert.

(b) A ist **NP**-schwer $\rightarrow A \in \mathbf{P}$.

Lösungsskizze

(a) Vielleicht. Wenn P=NP, dann ja, weil dann alle Probleme in NP (von NTM in polynomieller Zeit lösbar) auch in P liegen (von DTM in polynomieller Zeit lösbar). Wenn P≠NP, dann nein, weil sonst folgender Widerspruch auftritt: ein Problem in NP \ P ist von einer NTM, nicht aber von einer DTM in polynomieller Zeit lösbar. Könnte eine DTM die NTM in polynomieller Zeit simulieren, wäre das Problem aber auch P.

(b) Falsch. Es gibt NP-schwere Probleme, die nicht in NP liegen. Diese liegen folglich auch definitiv nicht in P. NP-schwer ist eine untere Schranke, $\in NP$ eine obere, und wenn beide da sind, ist es NP-vollständig.

AUFGABE 11.3. (*Einführung polynomielle Reduktion*) Stufe C

Sie kennen bereits das Problem HAMILTON := $\{G \mid G$ enthält einen Hamiltonkreis$\}$. Dieses Problem ist **NP**-vollständig.
Zeigen Sie nun, dass das Travelling-Salesman-Problem (TSP) ebenfalls **NP**-vollständig ist.
TSP:

- Eingabe: $n \times n$ Matrix M mit $M_{i,j} \in \mathbb{N}$ und eine Zahl $k \in \mathbb{N}$.
- Frage: Gibt es eine "Rundreise" der Länge $\leq k$.

Eine Rundreise ist eine Permutation π von $[1; n]$, also ein einfacher Kreis, der jede Insel einmal enthält. Die Länge der Rundreise ist die Summe der Kosten der einzelnen Reisen, also $\Sigma_{i=1}^{n} M_{\pi_i, \pi_j}$ (dabei ist π_i der i-te Eintrag der Permutation, beginnend bei 0, und π_j ist der Eintrag für die nächste Insel, also $j = (i+1)\%n$).

Lösungsskizze
Vergleiche Folie 377.
Hier sehr ausführlich:
Um zu zeigen, dass TSP **NP**-vollständig ist, müssen wir zeigen, dass TSP $\in$ **NP** und dass TSP **NP**-schwer ist.

- $\in$ **NP**: Wir können eine korrekte Rundreise raten, und dann in polynomieller Zeit verifizieren, dass sie eine Länge $\leq k$ hat, indem wir einmal den Pfad entlang gehen und die Kosten ausrechnen. Das entspricht $|\pi| = n$ Zugriffen auf die Matrix, geht also in linearer Zeit.
- **NP**-schwer: Durch Reduktion HAMILTON $\leq_p$ TSP. Gegeben das Encoding eines Graphen $G = (V, E)$ konstruieren wir eine Matrix wie folgt: Wir nummerieren die Knoten in G von 1 bis n durch. Dann ist

$$M_{i,j} := \begin{cases} 1 & \text{falls } \{v_i, v_j\} \in E \\ 2 & \text{sonst} \end{cases}$$

  Also sind die Kosten, von i nach j zu kommen 1, wenn eine Kante in G existiert, und 2, wenn nicht. Dann ist (M, n) eine Instanz des TSP und es gilt $G \in$ HAMILTON $\Leftrightarrow f(G) = (M, n) \in$ TSP. (noch zu beweisen)
  Gegeben etwas, das keinen Graphen encodiert, geben wir etwas zurück, das nicht $\in$ TSP ist, also z.B. etwas, das kein Tupel (M, k) encodiert oder eine Matrix mit positiven Kosten und $k = 0$.
  Außerdem noch ein Randfall: Gegeben einen Graphen, der weniger als 3 Knoten hat, geben wir etwas zurück, das nicht $\in$ TSP ist, weil ein Graph mit weniger als 3 Knoten niemals einen Hamilton-Kreis enthalten kann; unsere Reduktion könnte aber eventuell ein Element aus TSP zurück geben. Beispiel: $G = (\{1, 2\}, \{\{1, 2\}\})$.
  Korrektheit:
  - Wenn $G \in$ HAMILTON, dann gibt es einen Kreis, der nur Kanten von G verwendet und alle Knoten genau einmal besucht. Folglich gibt es auch eine Rundreise mit Kosten n in M.

– Wenn $G \notin$ HAMILTON, dann gibt es keinen Kreis, der nur Kanten von G verwendet und alle Knoten genau einmal besucht. Folglich gibt es keine Rundreise mit Kosten $\leq n$ in M, da mindestens einmal eine Kante mit Kosten 2 verwendet werden muss.
– Gegeben etwas, das keinen Graph encodiert, also nicht in HAMILTON ist, geben wir etwas zurück, das nicht in TSP ist.

Berechenbarkeit in polynomieller Zeit:
Die Reduktion definiert eine $n \times n$ Matrix, wobei n die Anzahl der Knoten von G ist. Folglich braucht sie quadratische Zeit in der Anzahl der Knoten.

AUFGABE 11.4. (*Abschlusseigenschaften von NP*) Stufe C
Seien $A, B \subseteq \Sigma^*$ Sprachen in NP. Zeigen Sie, dass $A \cap B$ ebenfalls in NP liegt. Geben Sie hierfür an, wie geeignete polynomiell verifizierbare Zertifikate hierfür aussehen.

Lösungsskizze
Da A und B in NP liegen, können wir annehmen, dass es polynomiell beschränkte Verifikatoren M_A und M_B für A bzw. B gibt. Für ein $w \in A$ gibt es also ein Zertifikat c, das "beweist", dass $w \in A$ liegt und dies in Polynomialzeit überprüft werden kann. M_A akzeptiert dann das Wort $w\#c$.
Ein Zertifikat für $w \in A \cap B$ besteht dann aus einem Paar von Zertifikaten, eines für $w \in A$ und eines für $w \in B$. Dabei benutzen wir wieder # als Trennzeichen. Hierbei ergibt sich noch eine Subtilität. Laut Definition 6.7 könnte ein polynomieller Verifikator auch "lange" Zertifikate (bspw. exponentiell lange in $|w|$) akzeptieren, obwohl er nur ein polynomiell langes Präfix lesen kann. Das Suchen nach einem Trennzeichen bzw. kopieren der Eingabe würde aber für solche langen Zertifikat nicht mehr in polynomieller Zeit funktionieren. Wir können aber das Lesen eines zusammengesetzten Zertifikats immer abbrechen und die Eingabe ablehnen, wenn wir mehr als $p(|w|)$ Zeichen gelesen haben (für ein gewisses Polynom p).
Für einen Verifikator für $A \cap B$ gilt damit:

$$w\#c \in L(M_\cap) \iff \exists c_A, c_B.\ c = c_A\#c_B \wedge w\#c_A \in L(M_A) \wedge w\#c_B \in L(M_B) \wedge |c_A\#c_B| \leq p(|w|)$$

Wiederum ist der Verifikator polynomiell beschränkt, da M_A und M_B es auch sind.

AUFGABE 11.5. (*Wizard of ZOLP*) Stufe D
Zeigen Sie, dass das Entscheidungsproblem *Zero-One-Linear-Program (ZOLP)* NP-schwer ist. Geben Sie hierzu eine geeignete Reduktion von 3-KNF-SAT auf ZOLP an und beschreiben Sie **zusätzlich** das Vorgehen anhand folgender Formel:

$$(\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_3 \vee x_4)$$

Definition (ZOLP-Entscheidungsproblem)

Eingabe: Ein System von linearen Ungleichungen

$$\begin{aligned} b_1 &\leq a_{1,1}y_1 + \ldots + a_{1,n}y_n \\ b_2 &\leq a_{2,1}y_1 + \ldots + a_{2,n}y_n \\ &\vdots \\ b_m &\leq a_{m,1}y_1 + \ldots + a_{m,n}y_n \end{aligned}$$

mit $a_{i,j}, b_i \in \mathbb{Z}$ und $m, n > 0$.
Frage: Gibt es für die Variablen $y_1, \ldots, y_n$ Werte aus $\{0, 1\}$, so dass alle Ungleichungen gleichzeitig erfüllt sind?

Lösungsskizze
Möglichkeit 1: Für jede Variable im Ausgangsproblem zwei Variablen im Zielproblem. Sei $F := \bigwedge_{i=1}^{k} C_i$ und $C_i := \bigvee_{j=1}^{3} L_{i,j}$ mit z Variablen die Eingabe für das 3KNF-SAT Problem. Die Reduktion f modelliert jede Variable des SAT-Problems mit zwei Variablen des ZOLP-Problems. Sei x_i eine Variable dann encodiert y_{2i}, ob das Literal x_i wahr ist, und y_{2i-1}, ob das Literal $\neg x_i$ wahr ist. Somit haben wir

$$f(x_i) \mapsto y_{2i} \qquad f(\neg x_i) \mapsto y_{2i-1}$$

Für das ZOLP wählen wir $m = k + 2z$ und $n = 2z$. Die Zeile $i \in [1, k]$ ist dann definiert in Abhängigkeit von C_i als

$$f(L_{i,1} \vee L_{i,2} \vee L_{i,3}) \qquad \mapsto \qquad 1 \leq f(L_{i,1}) + f(L_{i,2}) + f(L_{i,3})$$

Zusätzlich fügen wir folgende Konsistenzgleichungen ein, damit y_{2i} und $y_{2i} - 1$ unterschiedliche Werte erhalten für alle $i \in [1, z]$

$$\begin{aligned} 1 &\leq y_{2i} + y_{2i-1} \\ -1 &\leq -y_{2i} - y_{2i-1} \end{aligned}$$

Die Reduktion ist polynomiell in der Eingabegröße, da für jede Klausel eine Ungleichung und für jede Variable 2 Ungleichungen erstellt werden.
Anwendung auf Beispiel:

$$\begin{aligned} 1 &\leq y_1 + y_4 + y_6 \\ 1 &\leq y_2 + y_3 + y_8 \\ 1 &\leq y_1 + y_2 \\ -1 &\leq -y_1 - y_2 \\ 1 &\leq y_3 + y_4 \\ -1 &\leq -y_3 - y_4 \\ 1 &\leq y_5 + y_6 \\ -1 &\leq -y_5 - y_6 \\ 1 &\leq y_7 + y_8 \\ -1 &\leq -y_7 - y_8 \end{aligned}$$

Möglichkeit 2: Für jede Variable im Ausgangsproblem eine Variable im Zielproblem. Skizze: Wir kodieren jede Klausel als eine Ungleichung im Zielproblem. Disjunktionen ersetzen wir mit Addition. Literale x_i werden in einen Summanden y_i übersetzt und Literale $\neg x_i$ in einen Summanden $1 - y_i$. Damit ist ein Literal l erfüllt genau dann wenn der entsprechende Summand den Wert 1 annimmt. Um sicherzustellen, dass jede Klausel erfüllt ist, steht auf der linken Seite jeder Ungleichung 1. Offensichtlich kann eine jede Ungleichung dieser Form in die von ZOLP geforderte Form gebracht werden.

$$(\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_3 \vee x_4)$$

Anwendung auf Beispiel:

$$\begin{aligned} 1 &\leq (1 - y_1) + y_2 + y_3 \\ 1 &\leq y_1 + (1 - y_3) + y_4 \end{aligned}$$

Umformen:

$$\begin{aligned} 0 &\leq -y_1 + y_2 + y_3 \\ 0 &\leq y_1 - y_3 + y_4 \end{aligned}$$

AUFGABE 11.6. (*co-NP*) Stufe E

Wir definieren co-NP über die Komplemente von Sprachen aus NP.

$$\text{co-NP} = \{\mathsf{L} \mid \overline{\mathsf{L}} \in \text{NP}\}$$

Eine Sprache L heißt co-NP-vollständig, falls $\mathsf{L} \in \text{co-NP}$ und für jedes $\mathsf{L}' \in \text{co-NP}$ gilt: $\mathsf{L}' \leq_p \mathsf{L}$.

(a) Beweisen Sie die folgende Behauptung:

Wenn die Sprache L NP-vollständig ist, so ist $\overline{\mathsf{L}}$ co-NP-vollständig.

(b) Sei $\mathcal{F}$ die Menge aller boolschen Formeln und $\mathcal{V}$ die Menge aller Variablen. Sei VALID die Menge aller Tautologien:

$$\text{VALID} \coloneqq \{F \in \mathcal{F} \mid \forall \sigma : \mathcal{V} \to \{0,1\}.\ \sigma(F) = 1\}$$

Beweisen Sie, dass VALID co-NP-vollständig ist

(c) Betrachten Sie Definition 6.7 sowie Satz 6.9 der Vorlesung erneut (Zertifikat und polynomiell beschränkter Verifikator). Diskutieren Sie, wie eine analoge Version von Definition 6.7 und Satz 6.9 für co-NP aussehen kann. Beweisen Sie dann Ihre Analogie zu Satz 6.9.

Lösungsskizze

(a) Offensichtlich folgt aus $\mathsf{L} \in \mathsf{NP}$: $\overline{\mathsf{L}} \in \mathsf{co\text{-}NP}$. Sei nun $\mathsf{L}' \in \mathsf{co\text{-}NP}$ eine beliebige Sprache. Weil L NP-vollständig ist, haben wir $\overline{\mathsf{L}'} \leq_p \mathsf{L}$ und somit auch $\mathsf{L}' \leq_p \overline{\mathsf{L}}$. Damit ist $\overline{\mathsf{L}}$ co-NP-vollständig.

(b) Um (b) zu zeigen, verwenden wir (a) und zeigen, dass $\overline{\mathsf{VALID}} = \{F \in \mathcal{F} \mid \exists \sigma : \mathcal{V} \to \{0,1\}^*.\ \sigma(F) = 0\}$ NP-vollständig ist. $\overline{\mathsf{VALID}} \in \mathsf{NP}$ folgt daraus, dass σ mit $\sigma(F) = 0$ ein polynomielles Zertifikat ist, das wir in polynomieller Zeit verifizieren können. Aus der Reduktion $f(F) = \neg F$ erhalten wir $\mathsf{SAT} \leq_p \overline{\mathsf{VALID}}$ und somit is VALID co-NP-vollständig.

(c) Analog zu Definition 6.7 behaupten wir dass $\mathsf{L} \in \mathsf{co\text{-}NP}$ gdw. es eine DTM M und ein Polynom p gibt, so dass $\{w \in \Sigma^* \mid \forall c \in \Delta^{p(|w|)}.\ w\#c \in \mathsf{L}(\mathsf{M})\} = \mathsf{L}$ und $\mathsf{time}_{\mathsf{M}}(w\#c) \leq p(|w|)$.

Beweis:
$\mathsf{A} \in \mathsf{co\text{-}NP} \Leftrightarrow \overline{\mathsf{A}} \in \mathsf{NP} \Leftrightarrow$ Es gibt einen polynomiell beschränkten Verifikator M für $\overline{\mathsf{A}}$
Damit folgt

$$\begin{aligned} & w \in \overline{\mathsf{A}} \Leftrightarrow \exists c \in \Delta^{p(|w|)}.\ w\#c \in \mathsf{L}(\mathsf{M}) \\ \iff & w \in \mathsf{A} \Leftrightarrow \nexists c \in \Delta^{p(|w|)}.\ w\#c \in \mathsf{L}(\mathsf{M}) \\ \iff & w \in \mathsf{A} \Leftrightarrow \forall c \in \Delta^{p(|w|)}.\ w\#c \notin \mathsf{L}(\mathsf{M}) \\ \iff & w \in \mathsf{A} \Leftrightarrow \forall c \in \Delta^{p(|w|)}.\ w\#c \in \mathsf{L}(\mathsf{M}') \end{aligned}$$

Wobei die DTM M' so konstruiert ist, dass sie $w\#c$ genau dann akzeptiert wenn $w\#c$ von M nach $p(|w|)$ Schritten nicht akzeptiert wird. Dazu beschränken wir die Laufzeit von M ähnlich wie auf Folie 328 und vertauschen die Endzustände und Nichtendzustände von M. Damit gilt $\{w \in \Sigma^* \mid \forall c \in \Delta^{p(|w|)}.\ w\#c \in \mathsf{L}(\mathsf{M}')\} = \mathsf{A}$ und $\mathsf{time}_{\mathsf{M}'}(w\#c) \leq p(|w|)$.