Theoretische Informatik HS24

Nicolas Wehrli

Übungsstunde 11

3. Dezember 2024

ETH Zürich
nwehrl@ethz.ch

Heute

1. Feedback zur Serie
2. NP-Vollständigkeit
3. How To P-Reduktion

Feedback zur Serie

Feedback

Aufgabe 25: off by one errors, $0 \leq j \leq s(n)$ gives $(s(n) + 1)$ possibilities for j.
Aufgabe 26: forgot to copy input for (a)/ walk back the input band pointer for (b) before 2nd simulation.

NP-Vollständigkeit

Verfizierer

Sei $L \subseteq \Sigma^*$ eine Sprache und sei $p : \mathbb{N} \to \mathbb{N}$ eine Funktion.
Ein Algorithmus A (MTM) ist ein **p-Verfizierer für L** mit $\mathbf{V}(\mathbf{A}) = \mathbf{L}$, falls A mit folgenden Eigenschaften auf allen Eingaben aus $\Sigma^* \times (\Sigma_{\text{bool}})^*$ arbeitet:

(i) $\text{Time}_A(w, x) \leq p(|w|)$ für jede Eingabe $(w, x) \in \Sigma^* \times (\Sigma_{\text{bool}})^*$.

(ii) Für jedes $w \in L$ existiert ein $x \in (\Sigma_{\text{bool}})^*$, so dass $|x| \leq p(|w|)$ und $(w, x) \in L(A)$. Das Wort x nennt man einen **Beweis** oder einen **Zeugen** der Behauptung $w \in L$.

(iii) Für jedes $y \notin L$ gilt $(y, z) \notin L(A)$ für alle $z \in (\Sigma_{\text{bool}})^*$.

Polynomialzeitverifikation

Falls $p(n) \in \mathcal{O}(n^k)$ für ein $k \in \mathbb{N}$, so sagen wir, dass A ein **Polynomialzeit-Verfizierer** ist.
Wir definieren die **Klasse der in Polynomialzeit verifzierbaren Sprachen** als

$$\mathbf{VP} = \{V(A) \mid A \text{ ist ein Polynomialzeit-Verfizierer}\}.$$

Bezug zu NP

Satz 6.8

VP = NP

Die Klasse NP ist demnach die Klasse aller Sprachen L, die für jedes $x \in L$ einen in $|x|$ polynomiell langen Beweis von "$x \in L$" haben, welchen man deterministisch in polynomieller Zeit verifizieren bekann.

Dies können wir benutzen, um $L \in \mathrm{NP}$ zu beweisen!

P-Reduktion

Seien $L_1 \subseteq \Sigma_1^*, L_2 \subseteq \Sigma_2^*$ zwei Sprachen. Wir sagen, dass $\mathbf{L_1}$ **polynomiell auf** $\mathbf{L_2}$ **reduzierbar ist,** $\mathbf{L_1 \leq_p L_2}$, falls eine polynomieller Algorithmus A existiert, der für jedes Wort $x \in \Sigma_1^*$ ein Wort $A(x) \in \Sigma_2^*$ berechnet, so dass

$$x \in L_1 \iff A(x) \in L_2$$

A wird eine **polynomielle Reduktion** von L_1 auf L_2 genannt.

Bemerkung: Analog zur EE-Reduktion, nur das A jetzt noch polynomiell laufen muss.

Eine Sprache L ist **NP-schwer**, falls für alle Sprachen $L' \in$NP gilt $L' \leq_p L$.
Eine Sprache L ist **NP-vollständig**, falls

(i) $L \in$NP und
(ii) L ist NP-schwer.

Lemma 6.7

Falls $L \in P$ und L ist NP-schwer, dann gilt P = NP.

Wir haben

$$\mathrm{SAT} = \{x \in (\Sigma_{\text{logic}})^* \mid x \text{ kodiert eine erfüllbare Formel in KNF}\}$$

Satz 6.9

SAT ist NP-vollständig.

Da

$$L_1 \leq_p L_2 \implies (L_2 \in \mathrm{P} \implies L_1 \in \mathrm{P})$$

können wir mit diesem Resultat die NP-Schwere anderer Probleme einfacher beweisen.

Klassische Probleme

$$
\begin{aligned}
\text{SAT} &= \{\phi \mid \phi \text{ ist eine erfüllbare Formel in KNF}\} \\
\text{CLIQUE} &= \{(G, k) \mid G \text{ ist ein ungerichteter Graph, der eine } k\text{-Clique enthält}\} \\
\text{VC} &= \{(G, k) \mid G \text{ ist ein ungerichteter Graph mit einer Knotenüberdeckung} \\
&\qquad \text{(vertex cover) der Mächtigkeit höchstens } k\}
\end{aligned}
$$

Higher-Level of Abstraction.

Wir müssen uns nicht mehr überlegen, wie die Probleminstanzen als endliche Wörter kodiert sind!

Das heisst auch, dass ihr nicht mehr explizit auf falsche Form überprüfen müsst.

Ihr könnt annehmen, dass die Eingabe jeweils schon eine wohlgeformte Instanz des Problems ist.

Aufgabe 6.22

Beweise

$$\text{VC} \leq_p \text{CLIQUE}$$

Zur Erinnerung:

$$\begin{aligned}\text{CLIQUE} &= \{(G,k) \mid G \text{ ist ein ungerichteter Graph, der eine } k\text{-Clique enthält}\}\\ \text{VC} &= \{(G,k) \mid G \text{ ist ein ungerichteter Graph mit einer Knotenüberdeckung}\\ &\qquad \text{(vertex cover) der Mächtigkeit höchstens } k\}\end{aligned}$$

Ein VC ist eine Knotenmenge $C \subseteq V$, so dass

$$\forall \{u,v\}. v \in C \vee u \in C.$$

Aufgabe 6.22

Wir beschreiben einen polynomiellen Algorithmus A:

Eingabe $(G = (V, E), k)$ für VC

1. Findet $\overline{G} = (V, \overline{E})$ mit $\overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$
2. Gibt $(\overline{G}, |V| - k)$ aus.

Es sollte klar sein, dass A polynomiell läuft.

Korrektheit

Wir beweisen nun

$$S \subseteq V \text{ ist ein Vertex Cover von } G \iff V \setminus S \text{ ist eine Clique von } \overline{G}$$

Aufgabe 6.22

$(\implies)$:

Sei $S \subseteq V$ ein Vertex Cover von G.

$\implies$ Per Definition gilt für jede Kante $\{u, v\} \in E$ mindestens $u \in S$ oder $v \in S$.

$\implies$ Also existiert keine Kante $\{u, v\} \in E$ mit $u, v \in V \setminus S$.

$\implies$ Deshalb gilt für alle $u, v \in V \setminus S, u \neq v$, dass $\{u, v\} \in \overline{E}$.

$\implies$ $V \setminus S$ ist eine Clique in $\overline{G}$.

Aufgabe 6.22

$(\Longleftarrow)$:

Sei $V \setminus S$ eine Clique in $\overline{G}$.

$\implies$ Per Definition gilt für alle Knotenpaare $u, v \in V \setminus S, u \neq v$ jeweils $\{u, v\} \in \overline{E}$.

$\implies$ Also existiert keine Kante $\{u, v\} \in E$ mit $u, v \in V \setminus S$.

$\implies$ Deshalb gilt für alle $\{u, v\} \in E$, dass $u \in S$ oder $v \in S$.

$\implies$ S ist ein Vertex Cover in G.

Aufgabe 6.22

Mit der Aussage

$$S \subseteq V \text{ ist ein Vertex Cover von } G \iff V \setminus S \text{ ist eine Clique von } \overline{G} \tag{1}$$

können wir nun die Korrektheit beweisen.

$$\begin{aligned}
(G, k) \in \mathrm{VC} &\iff \exists S \subseteq V : S \text{ ist ein VC von } G \text{ und } |S| \leq k \\
&\iff V \setminus S \text{ ist eine Clique von } \overline{G} \text{ und } |V \setminus S| \geq |V| - k \\
&\iff (\overline{G}, |V| - k) \in \mathrm{CLIQUE} \\
&\iff A((G, k)) \in \mathrm{CLIQUE}
\end{aligned}$$

$\blacksquare$

How To P-Reduktion

PROBLEM $\in$ NP

Beschreibung eine NTM M, die PROBLEM erkennt mit folgender Form:

1. M errät für eine Eingabe x nicht deterministisch ein Zertifikat/Beweis (z.B. eine erfüllende Belegung für SAT oder eine Clique für CLIQUE).
2. M verfiziert das Zertifikat deterministisch in Polynomialzeit.

Korrektheit

$x \in$ PROBLEM $\iff$ Es existiert ein solches Zertifikat $\iff$ Es existiert eine akzept. Berechnung von M auf x $\iff$ $x \in L(M)$

PROBLEM ist NP-schwer

Beweise

$$\text{OTHERPROBLEM} \leq_p \text{PROBLEM}$$

für ein anderes OTHERPROBLEM, dass NP-schwer ist (in der Vorlesung gezeigt oder ähnl.).

Alternativ könnte man auch etwas wie den Beweis vom Satz von Cook machen. **(Don't)**

PROBLEM ist NP-Vollständig

Zeige

1. PROBLEM $\in$ NP
2. PROBLEM ist NP-schwer

OTHERPROBLEM $\leq_p$ PROBLEM

1. **Beschreibung** eines Algorithmus A, so dass

$$x \in \text{OTHERPROBLEM} \iff A(x) \in \text{PROBLEM}$$

2. **Korrektsheitsbeweis** von

$$x \in \text{OTHERPROBLEM} \iff A(x) \in \text{PROBLEM}$$

   (nichttrivial)
3. **Polynomialzeit** von A beweisen/argumentieren. Meistens recht einfach.

Idee finden für Polynomialzeitreduktion

Grundsätzlich 2 Typen von Problemen/Sprachen:

1. Satisifiability: Logische Formeln (deren Erfüllbarkeit). Beispielsweise SAT und 3SAT, 4SAT...
2. Graphenprobleme: Eigenschaften von Graphen. Beispielsweise CLIQUE, VC, DS etc.

Satisfiablity zu Satisfiability Reduktionen

Meist müssen wir einzelne Klauseln umschreiben. Veränderung der Anzahl Literale.

- **Literale verringern.** (Für allg. siehe im Buch, Lemma 6.11)
  Beispiel: 6SAT $\leq_p$ 4SAT

$$(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5) \mapsto (y_1 \vee x_1 \vee x_2 \vee x_3) \wedge (\overline{y}_1 \vee x_4 \vee x_5)$$

$$(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6) \mapsto (y_1 \vee x_1 \vee x_2 \vee x_3) \wedge (\overline{y}_1 \vee x_4 \vee x_5 \vee x_6)$$

  Beispiel: E8SAT $\leq_p$ E4SAT

$$\begin{aligned}(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6 \vee x_7 \vee x_8) \mapsto &(y_1 \vee x_1 \vee x_2 \vee x_3) \wedge (\overline{y}_1 \vee x_4 \vee x_5 \vee y_2) \\ &\wedge (\overline{y}_2 \vee x_6 \vee x_7 \vee x_8)\end{aligned}$$

Satisfiability zu Satisfiability Reduktion

- **Mehr Literale.**
  Beispiel: 5SAT $\leq_p$ E5SAT

$$(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5) \mapsto (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5)$$

$$(x_1 \vee x_2 \vee x_3 \vee x_4) \mapsto (x_1 \vee x_2 \vee x_3 \vee x_4 \vee y_1) \wedge (x_1 \vee x_2 \vee x_3 \vee x_4 \vee \overline{y}_1)$$

$$\begin{aligned}(x_1 \vee x_2 \vee x_3) \mapsto &(x_1 \vee x_2 \vee x_3 \vee y_1 \vee y_2) \\ &\wedge (x_1 \vee x_2 \vee x_3 \vee \overline{y}_1 \vee y_2) \\ &\wedge (x_1 \vee x_2 \vee x_3 \vee y_1 \vee \overline{y}_2) \\ &\wedge (x_1 \vee x_2 \vee x_3 \vee \overline{y}_1 \vee \overline{y}_2)\end{aligned}$$

etc.

Satisfiability zu Satisfiability Reduktion

- **Mehr erfüllende Belegungen.**
  Füge Klausel hinzu. I.e.

$$\phi \mapsto \phi \wedge (y_1 \vee y_2)$$

  verdreifacht die Anzahl der erfüllenden Belegungen.

Schema

Sei $F = F_1 \wedge ... \wedge F_m$ eine KNF Formel vom Typ OTHERPROBLEM

Eingabe für A: F

- A konstruiert $B = B_1 \wedge ... \wedge B_m$ mit einem der Tricks von oben.

Korrektheit

$$
\begin{aligned}
F \in \text{OTHERPROBLEM} &\iff F \text{ erfüllbar} \\
&\iff \text{ Es existiert eine Belegung } \varphi : \varphi(F) = 1 \\
&\iff \exists \varphi : \varphi(F_i) = 1 \forall i \in \{1, ..., m\} \\
&\text{... Argumentation für beliebige Klausel} \\
&\iff \exists \varphi' : \varphi'(B_i) = 1 \forall i \in \{1, ..., m\} \\
&\iff B \text{ erfüllbar} \\
&\iff B \in \text{PROBLEM}
\end{aligned}
$$

Graphproblem zu Graphproblem - Reduktion

Patterns

1. Aus $G = (V, E)$ Komplementgraph $\overline{G}$ bilden. I.e. $\overline{G} = (V, \overline{E})$ mit

$$\overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$$

2. Eingabetupel (G, k) zu $(G, n - k)$ abbilden ($n = |V|$).
3. Beides davon (siehe CLIQUE zu VC).
4. Ersetzen einer Kante durch anderes Konstrukt
    - 2 Kanten mit Knoten dazwischen
    - Knoten hinzufügen für jede Kante und mit beiden Eckpunkten verbinden

Satisfiability zu Graphproblem

Beispiel: Lemma 6.9 SAT $\leq_p$ CLIQUE

Denkt über die Reduktion vom Satisfiability Problem zu SAT und von CLIQUE zum Graphproblem.

Versucht diese Abbildungen zu verknüpfen.

Graphenproblem zu Satisfiability

Vielleicht **Satz von Cook** verwenden?

Scheint sehr komplex eine konkrete Reduktion zu machen.

Generelle Idee

- Denke über Zertifikate für PROBLEM nach. Welche Bestandteile haben sie? Beispielsweise im Fall von Clique ist das Zertifikat eine Knotenmenge (alle untereinander verbunden).
- Kreiert eine Variablenmenge mit einer Variable pro mögliches Element für das Zertifikat.
  I.e. im Fall von CLIQUE eine Variable pro Knoten
- I.e. Wir werden dann für eine Belegung der Formel wissen welche Knoten im Zertifikat enthalten sind.
- Baue die Formel, die die Bedingungen kodiert, damit genau dann erfüllt sein kann, falls es ein Zertifikat gibt.