



CHRISTIAN-ALBRECHTS-UNIVERSITÄT ZU KIEL

Institut für Informatik, Arbeitsgruppe Theoretische Informatik
Dr. Max A. Deppert, Janina Reuter, Annika Huch

15.06.2026

Präsenzaufgaben zur Vorlesung »Analyse von Algorithmen und Komplexität«

Blatt 10

Präsenzaufgabe 10.1 (Fragen über Fragen)

Diskutieren Sie die folgenden Aussagen (Gelten sie allgemein? Gelten sie nicht? Gelten sie unter bestimmten Bedingungen? Welche?):

- (i) Für jede Sprache, die von einer NDTM in polynomieller Zeit akzeptiert werden kann, existiert eine DTM, die ebenfalls die Sprache in polynomieller Zeit akzeptiert.
- (ii) A ist NP -schwer $\implies A \in P$

Lösung. (i) Die Aussage gilt nur, wenn $P = NP$. Dann liegen alle Probleme in NP (von NDTM in polynomieller Zeit lösbar) auch in P (von DTM in polynomieller Zeit lösbar). Falls $P \neq NP$, dann nein, weil sonst folgender Widerspruch auftritt: ein Problem in $NP \setminus P$ ist von einer NDTM, nicht aber von einer DTM in polynomieller Zeit lösbar. Könnte eine DTM die NDTM in polynomieller Zeit simulieren, wäre das Problem auch in P .

- (ii) Falsch. Es gibt NP -schwere Probleme, die nicht in NP liegen, z.B. das HALTE-Problem. Diese liegen folglich auch definitiv nicht in P . NP -schwer ist eine untere Schranke, $\in NP$ eine obere, und wenn beide erfüllt sind, ist ein Problem NP -vollständig.

□

Definition 1 (Vertex Cover (VC)). Für einen (ungerichteten) Graphen $G = (V, E)$ ist ein Vertex Cover eine Menge von Knoten $C \subseteq V$, sodass $u \in C$ oder $v \in C$ für alle $\{u, v\} \in E$ gilt, also ist für alle Kanten mindestens ein Endpunkt in C .

Für das VERTEXCOVER Problem ist neben einem Graphen G eine Zahl $k \in \mathbb{Z}_{\geq 0}$ gegeben und es wird gefragt, ob ein Vertex Cover mit Größe (Anzahl der Knoten) höchstens k in G existiert.

Präsenzaufgabe 10.2 (VERTEXCOVER ist NP -vollständig)

Zeigen Sie, dass VERTEXCOVER NP -vollständig ist.

Hinweis: Reduzieren Sie CLIQUE auf VERTEXCOVER.

Lösung. VERTEXCOVER $\in NP$ ist bekannt aus der letzten Hausaufgabe.

Vorüberlegung zur Reduktion von CLIQUE auf VERTEXCOVER: Sei eine Instanz mit $G = (V, E)$ und $k \in \mathbb{Z}_{\geq 0}$ von CLIQUE gegeben. Wir invertieren zunächst den Graphen G und erhalten damit G' . Nun enthält G' genau dann eine unabhängige Menge der Größe ℓ , wenn G eine Clique der Größe ℓ enthält. Wenn ein Graph mit n Knoten eine unabhängige Menge mit mindestens k Knoten hat, dann hat der Graph auch ein Vertex Cover mit maximal $n - k$ Knoten

(das Komplement zur unabhängigen Menge). Unsere Instanz für VERTEXCOVER ist also G' und $n - k$. ($k \leq n$ können wir annehmen)

Betrachte nun folgende **Reduktion**:

Sei eine Instanz mit $G = (V, E)$ und $k \in \mathbb{Z}_{\geq 0}$ von CLIQUE gegeben. Invertiere den Graphen G und erhalte damit G' . Gebe nun G' und $n - k$ als Instanz von VERTEXCOVER zurück.

Korrektheit:

$\Rightarrow$ (Falls (G, k) eine Ja-Instanz von CLIQUE ist, dann ist $(G', n - k)$ eine Ja-Instanz von VERTEXCOVER.):

G enthalte eine Clique C der Größe $\geq k$. Dann ist $V' := V \setminus C$ ein Vertex Cover von G' der Größe $\leq n - k$. Da $|C| \geq k$, gilt $|V'| \leq n - k$. Da C eine Clique ist, also alle Knoten aus C mit allen anderen Knoten aus C verbunden sind, gibt es im invertierten Graphen G' keine Kante zwischen Knoten aus C . Also gilt für jede Kante $\{u, v\} \in E'$, dass $u \notin C$ oder $v \notin C$ und somit $u \in V'$ oder $v \in V'$.

$\Leftarrow$ (Falls $(G', n - k)$ eine Ja-Instanz von VERTEXCOVER ist, dann ist (G, k) eine Ja-Instanz von CLIQUE.):

G' enthalte ein Vertex Cover V' der Größe $\leq n - k$. Dann ist $C := V \setminus V'$ eine Clique der Größe $\geq k$ in G . Da $|V'| \leq n - k$, gilt $|C| \geq k$. Für jede Menge $\{u, v\} \subseteq C$ gilt, dass $\{u, v\} \notin E'$. Ansonsten wäre V' kein korrektes Vertex Cover. Da G' der invertierte Graph von G ist, gilt somit $\{u, v\} \in E$ für jede Menge $\{u, v\} \subseteq C$.

Laufzeit: Das Invertieren des Graphen kann naiv in $O(|V|^2 \cdot |E|)$ Zeit, also in polynomieller Zeit, erledigt werden.

Also existiert eine polynomielle Reduktion von CLIQUE auf VERTEXCOVER. Da CLIQUE NP-vollständig ist und VERTEXCOVER $\in$ NP gilt, ist somit VERTEXCOVER NP-vollständig. $\square$

Präsenzaufgabe 10.3 (Turingmaschinen)

Entwerfen Sie eine Turingmaschine, welche die Sprache $L = \{0^{2^n} \mid n \in \mathbb{Z}_{\geq 0}\}$ über dem Alphabet $\Sigma = \{0\}$ entscheidet. Geben Sie die Laufzeit Ihrer Turingmaschine an. Begründen Sie die Korrektheit und die Laufzeit Ihrer Turingmaschine. Diskutieren Sie die Laufzeit Ihrer Turingmaschine im Vergleich zu einem Algorithmus (mit RAM) für diese Sprache.

Lösung. Die Turingmaschine sieht folgendermaßen aus:

- (1) Falls genau eine 0 auf dem Band steht, akzeptiere.
- (2) Bewege den Kopf von links nach rechts über die Eingabe und ersetze jede zweite 0 durch ein neues Symbol x .
- (3) Falls das letzte Symbol der Eingabe nun eine 0 ist, verwirfe.
- (4) Gehe zu Schritt (1).

Korrektheit: Die Sprache L besteht aus allen Wörtern, die nur Nullen enthalten und deren Länge eine Zweierpotenz ist. Eine Zahl ist genau dann eine Zweierpotenz, wenn sie nach wiederholtem Teilen durch 2 irgendwann eine 1 ergibt. Genau das überprüft die Turingmaschine. Wenn bei diesem wiederholten Teilen eine ungerade Zahl auftritt, ist die Zahl keine Zweierpotenz. In diesem Fall verwirft die Turingmaschine die Eingabe (siehe Schritt (3)).

Laufzeit: Die angegebene Turingmaschine hat eine Laufzeit von $O(n \log(n))$, wobei n hier die Länge der Eingabe bezeichnet. Schritt (1) und Schritt (2) benötigen jeweils eine Laufzeit von

$O(n)$. Schritt (3) geht in konstanter Laufzeit $O(1)$. Schritt (4) wiederholt die Schritte (1) bis (3) bis nur noch eine 0 auf dem Band steht oder vorher abgebrochen wird. Da in jedem Durchlauf der Schritte (1) bis (3) die Hälfte der Nullen (abgerundet) durch x Symbole ersetzt wird, steht nach $\log n$ Durchläufen nur noch eine Null auf dem Band und es wird nach Schritt (1) akzeptiert oder es steht nach $\lfloor \log n \rfloor - 1$ Durchläufen eine ungerade Anzahl von Nullen auf dem Band und es wird nach Schritt (3) verworfen. Also gibt es maximal $\log n + 1$ Durchläufe.

Diskussion: Ein Algorithmus mit RAM könnte folgendermaßen aussehen:

1. Zähle die Buchstaben und erhalte Anzahl m .
2. Falls $\log m$ ganzzahlig ist, akzeptiere. Sonst verwirfe.

Dieser Algorithmus hat eine Laufzeit von $O(n)$, wobei n hier die Länge der Eingabe bezeichnet. Ein Algorithmus mit RAM kann also den zusätzlichen Speicher nutzen, um wesentlich schneller eine Aufgabe zu erledigen, im Vergleich zu einem Algorithmus, der auf einer Turingmaschine läuft. $\square$