# Algorithmen und Datenstrukturen 2

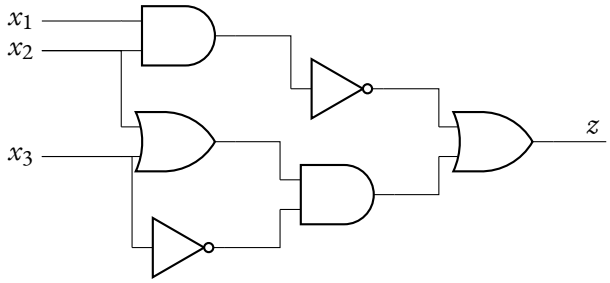
ALGO2 · WiSe-2023/24 · tcs.uni-frankfurt.de/algo2/ · 2024-02-05 · 7ae310c

## Übungen zu Woche 7: NP-Härte I

### Dienstag

Hinweis: Auf Moodle findet ihr eine Umfrage zu dieser Woche, bitte bis Montagabend ausfüllen.

**7.1 Schaltkreise.** In der Vorlesung haben wir eine Reduktion von CircuitSat auf 3SAT gesehen. Welche 3CNF Formel gibt diese Reduktion aus, wenn die Eingabe aus dem folgenden Schaltkreis besteht?




**7.2 DNF Erfüllbarkeit.** Eine aussagenlogische Formel ist in *disjunktiver Normalform* (DNF), wenn sie eine Disjunktion (Or) von Konjunktionstermen (And) ist. Ein Beispiel für eine DNF ist:

$$(\overline{x} \wedge y \wedge \overline{z}) \vee (y \wedge z) \vee (x \wedge \overline{y} \wedge \overline{z}).$$

Gegeben ist eine aussagenlogische Formel in disjunktiver Normalform. DNF-Sat entscheidet, ob diese Formel erfüllbar ist.

a) Beschreibe einen Algorithmus, der DNF-Sat in Polynomialzeit löst.

b) Wir zählen das Jahr 2050, Prof. Regloh steht kurz vor dem Ruhestand, das Institut für Informatik steht kurz vor dem Umzug auf den Campus Riedberg, und du stehst kurz vor deinem Studienabschluss. Prof. Regloh möchte am Ende seiner Forschungskarriere nochmal einen richtig großen Erfolg erzielen und hat sich schon seit ein paar Monaten psychisch auffällig verhalten. Jetzt rennt er jubelnd durch die Gänge, im Vorbeigehen schreit er dir folgendes zu:

> Angenommen, wir haben eine aussagenlogische Formel in konjunktiver Normalform mit höchstens 3 Literalen pro Klausel. Wir wollen herausfinden, ob diese Formel erfüllbar ist. Wir können das Distributivgesetz für Boolesche Operationen verwenden, um eine äquivalente Formel in disjunktiver Normalform zu konstruieren. Zum Beispiel:
>
> $$(x \vee y \vee \overline{z}) \wedge (\overline{x} \vee \overline{y}) \Leftrightarrow (x \wedge \overline{y}) \vee (y \wedge \overline{x}) \vee (\overline{z} \wedge \overline{x}) \vee (\overline{z} \wedge \overline{y}).$$
>
> Nun können wir den Algorithmus aus Aufgabenteil a) verwenden, um in Polynomialzeit herauszufinden, ob die entstandene DNF erfüllbar ist. Wir haben also 3Sat in Polynomialzeit gelöst! Da 3Sat NP-schwer ist, gilt P = NP! Ich werde eine Million Dollar als Preisgeld erhalten und berühmt werden!

Stimmt sein Beweis? Begründe deine Antwort.[1]

[1] Hier ist eine Liste von 115 Manuskripten, die das P versus NP Problem mit derartigen oder ähnlich fehlerhaften Argumenten angeblich lösen: https://www.win.tue.nl/~wscor/woeginger/P-versus-NP.htm

🎪 **7.3 Interval Scheduling.** Betrachte das folgende Entscheidungsproblem:

> IntervalScheduling:
> Sei eine Menge von Intervallen auf einer Zeitleiste sowie eine Zahl $k$ gegeben. Enthält diese Menge eine Teilmenge von sich nicht überschneidenden Intervallen der Größe mindestens $k$?

Im Folgenden schreiben wir $A \leq_p B$, wenn es eine Polynomialzeit-Reduktion gibt, die das Problem $A$ auf das Problem $B$ reduziert. Das heißt, das Problem $A$ soll in Polynomialzeit gelöst werden, falls es einen Algorithmus gibt, der $B$ in Polynomialzeit löst.

a) 🔑 Gilt IntervalScheduling $\leq_p$ MaximumIndependentSet? Begründe deine Antwort.
*Tipp:* Modelliere Intervalle als Knoten. Was sind die Kanten?

b) 🔑 Gilt MaximumIndependentSet $\leq_p$ IntervalScheduling? Begründe deine Antwort.
*Tipp:* Erinnere dich an Greedy-Algorithmen. Welche Konsequenzen hätte eine solche Reduktion?

🎪 **7.4 Independent Set.** Eine *unabhängige Menge* in einem Graphen $G = (V, E)$ ist eine Teilmenge $S \subseteq V$, sodass $\{u, v\} \notin E$ für alle $u, v \in S$ gilt. Betrachte die folgenden drei Probleme:

IndependentSet: Gegeben ein Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$, gib True aus, wenn $G$ eine unabhängige Menge der Größe $k$ besitzt, sonst False.

MaximumIndependentSet: Gegeben ein Graph $G = (V, E)$, gib die Größe der größten unabhängigen Menge in $G$ aus.

SearchMaximumIndependentSet: Gegeben ein Graph $G = (V, E)$, gib eine unabhängige Menge in $G$ mit maximaler Größe aus.

Hierbei ist IndependentSet ein *Entscheidungsproblem*, MaximumIndependentSet ist ein *Optimierungsproblem* und SearchMaximumIndependentSet ist ein *Suchproblem*. Wir zeigen nun, dass alle drei Problem äquivalent sind bezüglich Polynomialzeit-Reduktionen.

a) 👍 Entwirf einen Polynomialzeit-Algorithmus $A$ für IndependentSet, der einen hypothetischen Polynomialzeit-Algorithmus $C$ für SearchMaximumIndependentSet als Subroutine benutzt.

b) 🔑 Entwirf einen Polynomialzeit-Algorithmus $B$ für MaximumIndependentSet, der einen hypothetischen Polynomialzeit-Algorithmus $A$ für IndependentSet als Subroutine benutzt.

c) 🔭 Entwirf einen Polynomialzeit-Algorithmus $C$ für SearchMaximumIndependentSet, der einen hypothetischen Polynomialzeit-Algorithmus $B$ für MaximumIndependentSet als Subroutine benutzt. *Tipp:* Lösche immer einen Knoten und frag das Orakel $B$.

d) 🔑 Was wissen wir nun über alle drei Probleme, falls P $\neq$ NP gilt?

**Donnerstag**

Moodle🐣 🔑 **7.5 Search-to-Decision für 3SAT.** Angenommen es gibt einen Algorithmus $A$, der in Polynomialzeit für jede 3CNF-Formel $\Phi$ entscheidet, ob diese erfüllbar ist. Konstruiere einen Polynomialzeit-Algorithmus $B$, der $A$ als Subroutine nutzt, um eine erfüllende Belegung von $\Phi$ zu berechnen.

**7.6 Perfektes Matching.** Sei $G = (V, E)$ ein ungerichteter Graph. Eine Teilmenge $M \subseteq E$ ist ein *Matching*, wenn keine zwei Kanten aus $M$ mit demselben Knoten verbunden sind. Ein Matching ist *perfekt*, wenn jeder Knoten $v \in V$ zu einer Kante $m \in M$ inzident ist. Eine äquivalente Definition betrachtet die Größe des Matchings: $M$ ist perfekt genau dann, wenn $|M| = |V|/2$ gilt. Wir definieren das PerfectMatching-Problem:

> PerfectMatching:
> Gegeben sei ein ungerichteter Graph $G = (V, E)$. Existiert ein perfektes Matching in $G$?

Dieses Problem kann in Polynomialzeit gelöst werden. Für allgemeine Graphen handelt es sich hierbei um einen recht komplizierten Algorithmus, der einen Meilenstein in der kombinatorischen Optimierung darstellt, und viele Anwendungen in Theorie und Praxis hat. Auf bipartiten Graphen ist das Problem einfacher zu lösen:

a) 🔑 Entwirf einen Polynomialzeit-Algorithmus für PerfectMatching auf bipartiten Graphen.
*Tipp:* Flussnetzwerk

Prof. Regloh behauptet, dass der folgende Algorithmus das Problem von allgemeinen Graphen auf bipartite Graphen reduziert:

> ReduceGraph($G$):
> Gegeben sei ein ungerichteter Graph $G = (V, E)$. Erstelle daraus einen bipartiten Graphen $H = (V \times \{1, 2\}, E_H)$ wie folgt:
> Jeder Knoten $u \in V$ wird zu zwei Kopien $(u, 1)$ und $(u, 2)$. Die Knotenmenge von $H$ ist also partitioniert in zwei disjunkte Teile $V_1$ und $V_2$ eines bipartiten Graphen, wobei
>
> $$V_1 = \{(u, 1) \mid u \in V\} \text{ und } V_2 = \{(u, 2) \mid u \in V\}.$$
>
> Die Menge $E_H$ der Kanten von $H$ ist definiert via
>
> $$E_H = \left\{ \{(u, 1), (v, 2)\} \;\middle|\; \{u, v\} \in E \right\}.$$
>
> Mit anderen Worten: Für alle Kanten $\{u, v\}$ auf $G$ fügen wir eine Kante zwischen $(u, 1)$ und $(v, 2)$ in $H$ ein. Beachte, dass $G$ keine Eigenschleifen hat und es dadurch in $H$ für alle $u \in V$ keine Kante $\{(u, 1), (u, 2)\}$ geben kann.

Ist die vorgeschlagene Reduktion korrekt? Um das herauszufinden, müssen wir überprüfen, ob $H$ genau dann ein perfektes Matching hat, wenn $G$ eins hat.

b) 🔑 Zeige die folgende Aussage: Wenn $G$ ein perfektes Matching hat, dann hat auch $H$ eins.

c) 🔑 Widerlege die folgende Aussage: Wenn $G$ kein perfektes Matching hat, dann hat auch $H$ keins.
- Konstruiere ein Gegenbeispiel, in dem $G$ drei Knoten hat. *Tipp:* Dreieck
- Konstruiere ein Gegenbeispiel, in dem $G$ eine gerade Anzahl an Knoten hat.

Begründe jeweils, warum die behauptete Implikation falsch ist.

## Weitere Aufgaben und Projekte

**7.7 Kundenanalyse.** Zur Analyse des Kundenverhaltens benutzen Geschäfte oftmals ein zweidimensionales Array $A$, in dem jede Zeile zu einem Kunden und jede Spalte zu einem verkauften Produkt korrespondiert. Ein Eintrag $A[i, j]$ gibt an, welche Menge von Produkt $j$ der Kunde $i$ gekauft hat, so hat Chelsea im folgenden Beispiel siebenmal Katzenstreu gekauft:

| | Waschmittel | Bier | Windeln | Katzenstreu |
|---|---|---|---|---|
| Raj | 0 | 6 | 0 | 3 |
| Alanis | 2 | 3 | 0 | 0 |
| Chelsea | 0 | 0 | 0 | 7 |

Ein Geschäft möchte nun zur Marktforschung eine *diverse* Teilmenge der Kunden finden. Eine Teilmenge $S$ der Kunden heißt *divers*, wenn keine zwei Kunden aus $S$ jemals dasselbe Produkt gekauft haben (das heißt, es gibt für jedes Produkt nur genau einen Kunden aus $S$, der es gekauft hat). Nun lässt sich das folgende Entscheidungsproblem formulieren:

> DiverseTeilmenge:
> Seien ein $n \times m$ Array $A$, wie oben definiert, und eine natürliche Zahl $k$ mit $k \leq n$ gegeben.
> Gibt es eine *diverse* Teilmenge $S$ der Kunden mit $|S| \geq k$?

Zeige, dass DiverseTeilmenge NP-vollständig ist.

**7.8 Search-to-Decision für 3COLOR.** Angenommen es gibt einen Algorithmus $A$, der in Polynomialzeit für jeden Graphen $G$ entscheidet, ob dieser eine echte Knotenfärbung mit drei Farben zulässt. Konstruiere einen Polynomialzeit-Algorithmus $B$, der $A$ als Subroutine nutzt, um eine solche echte Färbung zu konstruieren. (Hinweis: $A$ darf wirklich nur ungefärbte Graphen als Eingabe erhalten.)