Algorithmen & Komplexität

Johannes Lengler
Institut für Theoretische Informatik

Komplexitätstheorie

P = *effizient entscheidbare Probleme*

NP = *(einseitig) effizient verifizierbare Probleme*

$P \overset{?}{=} NP$ → *1 Million US-$ (Clay-Foundation)*

[eines von sieben Millennium-Problemen]

P und NP

Wir betrachten Funktionen der Form $f : \{0,1\}^* \to \{0,1\}$.

Äquivalent: Betrachten Sprachen $L \subseteq \{0,1\}^*$.
Wir bezeichnen f (bzw. L) auch als *Entscheidungsproblem*.

Definition:

Eine Sprache $L \subseteq \{0,1\}^*$ ist in P, falls es einen Algorithmus A gibt, der in polynomieller Zeit mit A(x)=L(x) terminiert.

Eine Sprache L ist in NP (*nicht-deterministisch polynomiell*), falls es einen Verifizierer V gibt, sodass V in polynomieller Zeit terminiert, und sodass

$$L(x) = 1 \iff \exists p \in \{0,1\}^* \text{ mit } |p| = \text{poly}(|x|) : V(x,p) = 1$$

und

$$L(x) = 0 \iff \forall p \in \{0,1\}^* : V(x,p) = 0.$$

Reduzierbarkeit

Definition:

Seien $f, g : \{0,1\}^* \to \{0,1\}$ Entscheidungsprobleme. Wir sagen, dass *f* *polynomiell reduzierbar* auf *g* ist, $f \leq_p g$, falls es eine Funktion $\kappa : \{0,1\}^* \to \{0,1\}^*$ gibt, die in polynomieller Zeit berechnet werden kann, mit:

$$\forall x \in \{0,1\}^* : \; f(x) = g(\kappa(x)).$$

Lemma:

Ist $f \leq_p g$, und ist *g* in P, so ist auch *f* in P.

NP-vollständig

Definition:

Ein Entscheidungsproblem *g* heisst *NP-schwer* genau dann, wenn $f \leq_p g$ für alle Probleme f in NP gilt. Ist zusätzlich g in NP, so heisst g *NP-vollständig*.

Lemma:

Ist $f \leq_p g$, und ist *f* NP-schwer, so ist auch *g* NP-schwer.

- Wir werden sehen: Es gibt NP-vollständige Probleme.
- Wenn Sie für ein(!) solches Problem einen polynomiellen Algorithmus finden, haben sie P=NP gezeigt!!
- Umgekehrt: Falls P≠NP, dann gibt es für keines dieser Probleme einen polynomiellen Algorithmus.

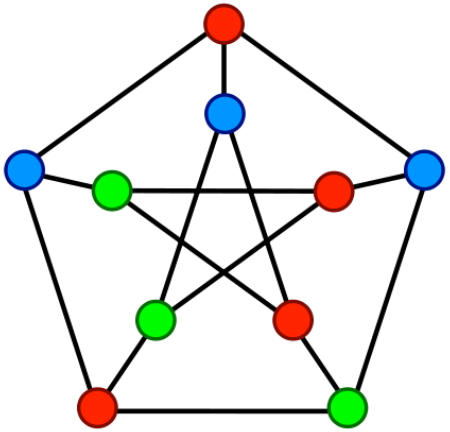
SAT ist NP-vollständig

Theorem (Cook-Levin, 1971):

- SAT ist NP-vollständig.
- 3-SAT ist NP-vollständig.

3-COL

Definition: Ein Graph G=(V,E) ist *3-färbbar*, wenn es eine Abbildung $f : V \to \{0, 1, 2\}$ gibt, sodass je zwei benachbarte Knoten auf verschiedene Werte abgebildet werden.



3-COL: $L = \{\text{Graph } G \mid \ G \text{ ist } 3\text{-färbbar}\}$

Satz: $3\text{-}SAT \leq_p 3\text{-}COL.$

Korollar: 3-COL ist NP-vollständig.

SATISFIABILITY

Disjunktive Normalform (DNF):

$$F = C_1 \vee C_2 \vee \ldots \vee C_m,$$

wobei jedes C_i von der Form $C_i = L_{i_1} \wedge L_{i_2} \wedge \ldots \wedge L_{i_\ell}$ ist,

$$L_j \in \{x_1 \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n\}.$$

Konjunktive Normalform (KNF):

$$F = C_1 \wedge C_2 \wedge \ldots \wedge C_m,$$

wobei jedes C_i von der Form $C_i = L_{i_1} \vee L_{i_2} \vee \ldots \vee L_{i_\ell}$ ist.

$$L_j \in \{x_1 \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n\}.$$

Klauseln

Literale

z.B.: $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_1) \wedge (\overline{x}_2)$

SAT ist NP-vollständig

Theorem (Cook-Levin):

- SAT ist NP-vollständig.
- 3-SAT ist NP-vollständig.

Lemma:

Sei A ein Algorithmus mit r Zeilen und seien p(n), q(n), s(n) Polynome, sodass der Algorithmus für jeden Input der Länge n

- nach höchstens s(n) Schritten mit 0 oder 1 terminiert;
- nur die Speicherzellen $M_1, \ldots, M_{p(n)}$ benutzt;
- in jede Speicherzelle höchstens q(n) Bits schreibt;

Dann gibt es für jedes n eine SAT-Formel F der Grösse poly(n), die Variablen $y_1, \ldots, y_n, z_1, \ldots, z_m$ enthält, sodass:
Für jedes $I = I_1 \ldots I_n \in \{0,1\}^n$ sei F_I die Formel, die wir durch Einsetzen von $y_1 := I_1, \ldots, y_n := I_n$ aus F erhalten. Dann ist

$$A \text{ gibt } 1 \text{ aus für Input } I \iff F_I \text{ erfüllbar.}$$

Ausserdem ist F in polynomieller Zeit berechenbar.