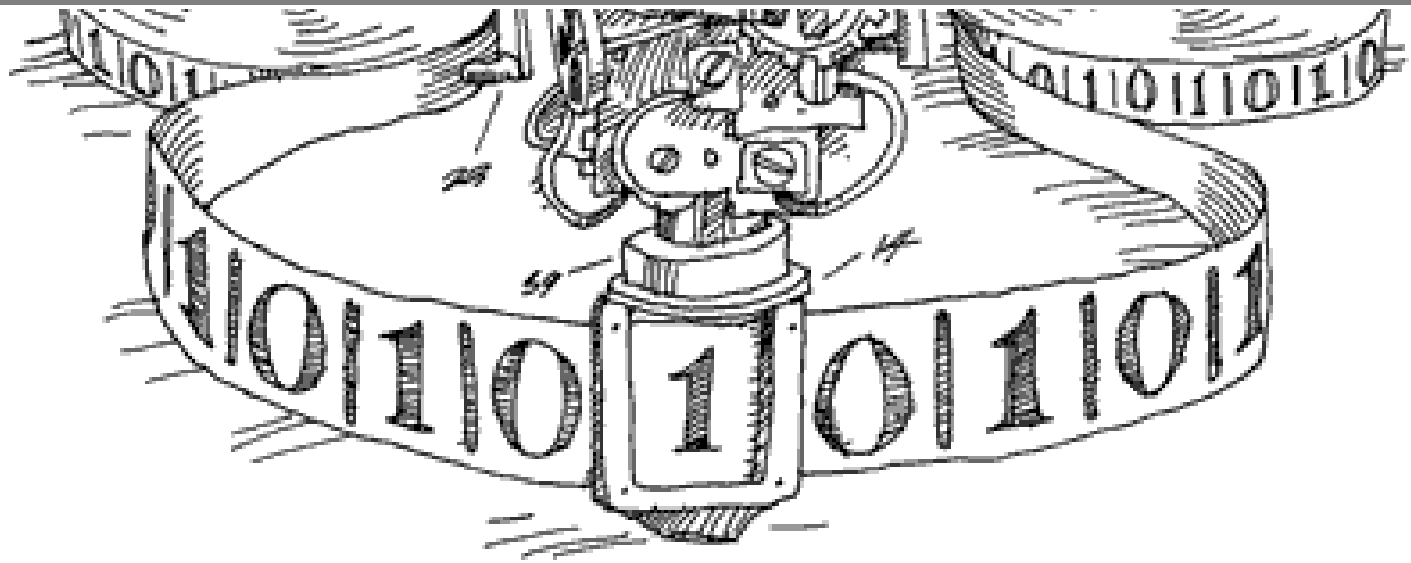
Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik

Tutorium 6

Institut für Theoretische Informatik



$\mathcal{NP}$

$\mathcal{NP}$ ist (analog zu $\mathcal{P}$) die Klasse aller Sprachen, die von einer nichtdeterministischen Turingmaschine in Polyzeit erkannt werden.

Anmerkung: Die Frage, ob $\mathcal{P} = \mathcal{NP}$ gilt oder nicht, ist ein großes, offenes Problem.

Grundidee: $\mathcal{NP}$-Entscheider

Üblicherweise geht eine NTM, die ein Problem aus $\mathcal{NP}$ entscheidet, folgendermaßen vor:

1. Rate sogenannten „Zeugen“ dafür, dass $x \in L$ (nichtdeterministisch)
2. Überprüfe, ob Zeuge korrekt (in Polyzeit).
3. Falls ja akzeptiere; falls nein, lehne ab.

Man spricht daher auch von den *effizient verifizierbaren* Entscheidungsproblemen.

Definition aus dem Skript (S. 56)

„Lasch“ ausgedrückt: Π gehört zu $\mathcal{NP}$, falls Π folgende Eigenschaft hat: Ist die Antwort bei Eingabe eines Beispiels *I* von Π „Ja“, so kann die Korrektheit der Antwort in polynomieller Zeit überprüft werden.

Ist diese Formulierung so korrekt?

Zeigen, dass ein Problem in $\mathcal{NP}$ liegt

Gegeben: Entscheidungsproblem Π
Aufgabe: Zeige, dass $\Pi \in \mathcal{NP}$ gilt.

Lösung

1. Gib an, was das NTM-Orakel als **Zeugen** für die Lösung auf das Band schreiben könnte.
2. Zeige, dass solch ein Zeuge in **polynomieller Zeit** von einer **DTM** verifiziert werden kann.

Beispiel

Zeige: $SAT \in \mathcal{NP}$.
„$SAT \in \mathcal{NP}$ gilt, da für eine gegebene Variablenbelegung in polynomieller Zeit von einer DTM überprüft werden kann, ob sie erfüllend ist.“

Zeitkomplexität

Definition aus der Übung

Berechnungszeit für NTM $\mathcal{M}$ mit Eingabe $x \in \Sigma^*$:

$$t(x) := \begin{cases} \#\text{ Schritte der } \textit{schnellsten} \text{ akzeptierenden} \\ \quad \text{Berechnung, falls } x \in L(\mathcal{M}) \\ 1, \text{ sonst} \end{cases}$$

Zeitkomplexitätsfunktion $T_{\mathcal{M}} : \mathbb{N}_0 \to \mathbb{N}$

$$T_{\mathcal{M}}(n) := \max \left\{ t(x) \,\middle|\, |x| = n \right\}$$

Aufgabe

Sei $\mathcal{M}$ eine NTM (RV-Modell) mit Zeitkomplexitätsfunktion $T_{\mathcal{M}} : \mathbb{N}_0 \to \mathbb{N}$.
Die Funktion $T_{\mathcal{M}}$ sei durch $f : \mathbb{N}_0 \to \mathbb{N}$ beschränkt und f sei berechenbar.
Zeige: Die von $\mathcal{M}$ akzeptierte Sprache $L(\mathcal{M})$ ist entscheidbar.

Aufgabe

Sei $\mathcal{M}$ eine NTM (RV-Modell) mit Zeitkomplexitätsfunktion $T_{\mathcal{M}} : \mathbb{N}_0 \to \mathbb{N}$.
Die Funktion $T_{\mathcal{M}}$ sei durch $f : \mathbb{N}_0 \to \mathbb{N}$ beschränkt und f sei berechenbar.
Zeige: Die von $\mathcal{M}$ akzeptierte Sprache $L(\mathcal{M})$ ist entscheidbar.

Lösungsskizze

Baue TM, die $L(\mathcal{M})$ entscheidet, wie folgt:
Sei x die Eingabe und $n := |x|$.

- Berechne $f(n)$
- Für alle Orakelwörter bis Länge $f(n)$:
  - Simuliere $\mathcal{M}$ mit aktuellem Orakelwort
  - Falls $\mathcal{M}$ akzeptiert, akzeptiere
  - Falls $\mathcal{M}$ ablehnt oder mehr als $f(n)$ Schritte braucht, probiere nächstes Orakelwort
- Falls $\mathcal{M}$ für kein Orakelwort akzeptiert, lehne ab.

Wiederholung: Polynomielle Transformation

Definition (Vorlesung)

Eine polynomielle Transformation einer Sprache L_1 in eine Sprache L_2 ist eine Funktion $f : \Sigma_1^* \to \Sigma_2^*$ mit den Eigenschaften:

1. f ist in polynomieller Zeit von einer deterministischen TM berechenbar.
2. Für alle x gilt: $x \in L_1 \iff f(x) \in L_2$

Wir schreiben dann: $L_1 \propto L_2$ (L_1 ist polynomiell transformierbar in (reduzierbar auf) L_2).

- L_2 ist das „schwerere“ Problem.
- Kann man L_2 entscheiden, so kann man mit polynomiellem Aufwand auch L_1 entscheiden.

$\mathcal{NP}$-Schwere

$\mathcal{NP}$-Schwere

Eine Sprache L_1 ist $\mathcal{NP}$-schwer gdw.

$$\forall L_2 \in \mathcal{NP} \ : \ L_2 \propto L_1$$

Anmerkung: In diesem Sinne sind die $\mathcal{NP}$-schweren Probleme mindestens so schwer zu lösen wie alle Probleme in $\mathcal{NP}$.

Transitivität von Polyreduktionen und $\mathcal{NP}$

Polynomielle Transformationen sind transitiv, d.h. wenn $L_1 \propto L_2$ und $L_2 \propto L_3$, dann gilt auch $L_1 \propto L_3$.

Ist $L_3 \in \mathcal{NP}$, so wissen wir, dass auch L_1 sowie $L_2 \in \mathcal{NP}$. Man spricht auch davon, L_1 und L_2 auf L_3 polynomiell reduziert zu haben.

Gegeben eine Sprache L_N, von der wir wissen, dass sie $\mathcal{NP}$-schwer ist, was wäre ein möglicher Ansatz, um für eine weitere Sprache L_X die $\mathcal{NP}$-Schwere zu beweisen?

Transitivität von Polyreduktionen und $\mathcal{NP}$

Polynomielle Transformationen sind transitiv, d.h. wenn $L_1 \propto L_2$ und $L_2 \propto L_3$, dann gilt auch $L_1 \propto L_3$.

Ist $L_3 \in \mathcal{NP}$, so wissen wir, dass auch L_1 sowie $L_2 \in \mathcal{NP}$. Man spricht auch davon, L_1 und L_2 auf L_3 polynomiell reduziert zu haben.

Gegeben eine Sprache L_N, von der wir wissen, dass sie $\mathcal{NP}$-schwer ist, was wäre ein möglicher Ansatz, um für eine weitere Sprache L_X die $\mathcal{NP}$-Schwere zu beweisen?

Man zeigt $L_N \propto L_X$.

$\mathcal{NP}$-Vollständigkeit

Eine Sprache L ist $\mathcal{NP}$-vollständig genau dann, wenn

$$L \in \mathcal{NP}$$

sowie

$$L \text{ ist } \mathcal{NP}\text{-schwer}$$

- $\Rightarrow$ $\mathcal{NP}$-vollständige Probleme sind die „schwersten“ Probleme aus $\mathcal{NP}$.
- Interessant vor allem, da man aus Aussagen über $\mathcal{NP}$-vollständige Probleme viel über alle Probleme aus $\mathcal{NP}$ schließen kann.
- Wäre etwa $SAT \in \mathcal{P}$, so wäre $\mathcal{P} = \mathcal{NP}$. (Warum?)

CLIQUE

Problem

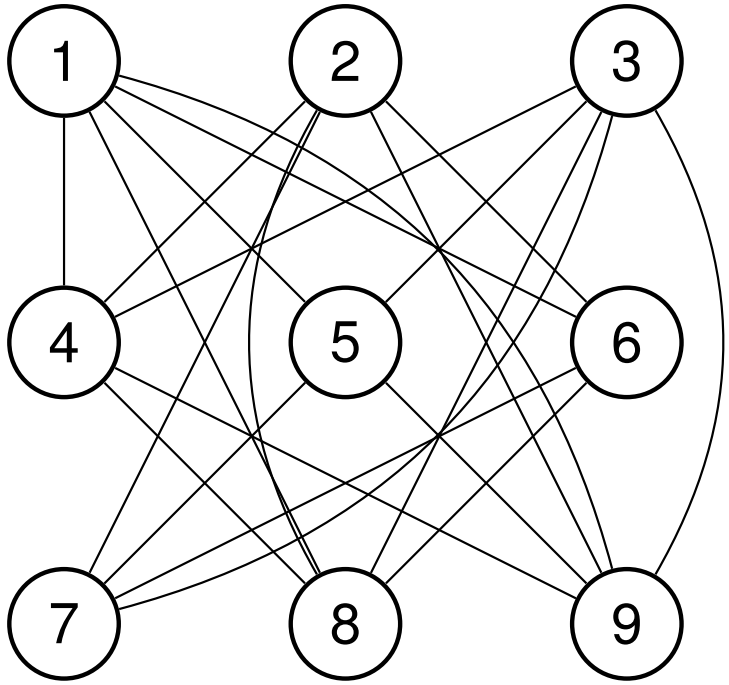
Gegeben: Graph $G = (V, E)$ und ein Parameter $K \leq |V|$
Frage: Gibt es in G eine Clique der Größe mindestens K?

Erinnerung

Eine Clique ist ein vollständig verbundener Teilgraph, also eine Menge $V' \subseteq V$, so dass für alle $i, j \in V'$ mit $i \neq j$ gilt: $(i, j) \in E$.

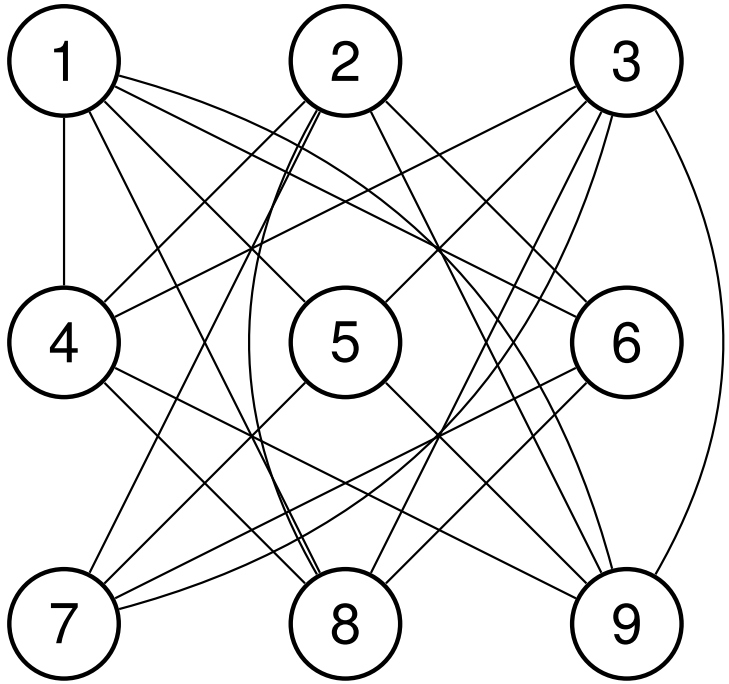
Dieses Problem ist $\mathcal{NP}$-vollständig.

CLIQUE-Beispiel



- Hat dieser Graph eine 3-CLIQUE?

CLIQUE-Beispiel



- Hat dieser Graph eine 3-CLIQUE?
- Hat dieser Graph eine 4-CLIQUE?

$\mathcal{NP}$-Vollständigkeitsbeweis

- *CLIQUE* $\in \mathcal{NP}$: Übungsblatt
- *CLIQUE* $\mathcal{NP}$-schwer: Wir zeigen 3*SAT* $\propto$ *CLIQUE*. (Warum?)

$\mathcal{NP}$-Vollständigkeitsbeweis

- *CLIQUE* $\in \mathcal{NP}$: Übungsblatt
- *CLIQUE* $\mathcal{NP}$-schwer: Wir zeigen 3*SAT* $\propto$ *CLIQUE*. (Warum?)
  Wir müssen eine polynomielle Transformation von 3*SAT*-Instanzen in *CLIQUE*-Instanzen angeben. (Warum?)

$\mathcal{NP}$-Vollständigkeitsbeweis

Sei $C = \{c_1, \ldots, c_n\}$ eine $3SAT$-Instanz mit

$$c_i = x_{i,1} \vee x_{i,2} \vee x_{i,3} \text{ mit } x_{i,j} \in \{u_1, \ldots, u_m, \overline{u_1}, \ldots, \overline{u_m}\}$$

Konstruiere eine $CLIQUE$-Instanz $(G = (V, E), K)$ folgendermaßen:

$$V := (v_{1,1}, v_{1,2}, v_{1,3}, v_{2,1}, \ldots, v_{n,1}, v_{n,2}, v_{n,3})$$

Jeder Knoten steht also für ein Literal in der $3SAT$-Instanz.

$$E := \left\{ (v_{i,j}, v_{k,m}) \,\middle|\, x_{i,j} \neq \overline{x_{k,m}} \wedge i \neq k \right\}$$

Zwei Knoten sind verbunden, wenn sie gleichzeitig erfüllbar sind und nicht in der gleichen Klausel stehen.

$$K := n$$

Wir suchen nach einer Clique der Größe n.

$\mathcal{NP}$-Vollständigkeitsbeweis

Sei $C = \{c_1, \ldots, c_n\}$ eine $3SAT$-Instanz mit

$$c_i = x_{i,1} \vee x_{i,2} \vee x_{i,3} \text{ mit } x_{i,j} \in \{u_1, \ldots, u_m, \overline{u_1}, \ldots, \overline{u_m}\}$$

Konstruiere eine $CLIQUE$-Instanz $(G = (V, E), K)$ folgendermaßen:

$$V := (v_{1,1}, v_{1,2}, v_{1,3}, v_{2,1}, \ldots, v_{n,1}, v_{n,2}, v_{n,3})$$

Jeder Knoten steht also für ein Literal in der $3SAT$-Instanz.

$$E := \left\{ (v_{i,j}, v_{k,m}) \,\middle|\, x_{i,j} \neq \overline{x_{k,m}} \wedge i \neq k \right\}$$

Zwei Knoten sind verbunden, wenn sie gleichzeitig erfüllbar sind und nicht in der gleichen Klausel stehen.

$$K := n$$

Wir suchen nach einer Clique der Größe n.

$\mathcal{NP}$-Vollständigkeitsbeweis

Sei $C = \{c_1, \ldots, c_n\}$ eine $3SAT$-Instanz mit

$$c_i = x_{i,1} \vee x_{i,2} \vee x_{i,3} \text{ mit } x_{i,j} \in \{u_1, \ldots, u_m, \overline{u_1}, \ldots, \overline{u_m}\}$$

Konstruiere eine $CLIQUE$-Instanz $(G = (V, E), K)$ folgendermaßen:

$$V := (v_{1,1}, v_{1,2}, v_{1,3}, v_{2,1}, \ldots, v_{n,1}, v_{n,2}, v_{n,3})$$

Jeder Knoten steht also für ein Literal in der $3SAT$-Instanz.

$$E := \left\{ (v_{i,j}, v_{k,m}) \,\middle|\, x_{i,j} \neq \overline{x_{k,m}} \wedge i \neq k \right\}$$

Zwei Knoten sind verbunden, wenn sie gleichzeitig erfüllbar sind und nicht in der gleichen Klausel stehen.

$$K := n$$

Wir suchen nach einer Clique der Größe n.

$\mathcal{NP}$-Vollständigkeitsbeweis

Sei $C = \{c_1, \ldots, c_n\}$ eine $3SAT$-Instanz mit

$$c_i = x_{i,1} \vee x_{i,2} \vee x_{i,3} \text{ mit } x_{i,j} \in \{u_1, \ldots, u_m, \overline{u_1}, \ldots, \overline{u_m}\}$$

Konstruiere eine $CLIQUE$-Instanz $(G = (V, E), K)$ folgendermaßen:

$$V := (v_{1,1}, v_{1,2}, v_{1,3}, v_{2,1}, \ldots, v_{n,1}, v_{n,2}, v_{n,3})$$

Jeder Knoten steht also für ein Literal in der $3SAT$-Instanz.

$$E := \left\{ (v_{i,j}, v_{k,m}) \,\middle|\, x_{i,j} \neq \overline{x_{k,m}} \wedge i \neq k \right\}$$

Zwei Knoten sind verbunden, wenn sie gleichzeitig erfüllbar sind und nicht in der gleichen Klausel stehen.

$$K := n$$

Wir suchen nach einer Clique der Größe n.

$\mathcal{NP}$-Vollständigkeitsbeweis

Sei $C = \{c_1, \ldots, c_n\}$ eine $3SAT$-Instanz mit

$$c_i = x_{i,1} \vee x_{i,2} \vee x_{i,3} \text{ mit } x_{i,j} \in \{u_1, \ldots, u_m, \overline{u_1}, \ldots, \overline{u_m}\}$$

Konstruiere eine $CLIQUE$-Instanz $(G = (V, E), K)$ folgendermaßen:

$$V := (v_{1,1}, v_{1,2}, v_{1,3}, v_{2,1}, \ldots, v_{n,1}, v_{n,2}, v_{n,3})$$

Jeder Knoten steht also für ein Literal in der $3SAT$-Instanz.

$$E := \left\{ (v_{i,j}, v_{k,m}) \,\middle|\, x_{i,j} \neq \overline{x_{k,m}} \wedge i \neq k \right\}$$

Zwei Knoten sind verbunden, wenn sie gleichzeitig erfüllbar sind und nicht in der gleichen Klausel stehen.

$$K := n$$

Wir suchen nach einer Clique der Größe n.

$\mathcal{NP}$-Vollständigkeitsbeweis

- Wenn eine n-Clique existiert:
  - n erfüllbare Literale
  - Alle in jeweils unterschiedlichen Klauseln
    (da Literalknoten in derselben Klausel nicht verbunden sind)
  - $\rightsquigarrow$ erfüllende Wahrheitsbelegung
- Wenn eine erfüllende Wahrheitsbelegung existiert:
  - In jeder Klausel mindestens ein Literal erfüllt
  - Nimm aus jeder Klausel ein erfülltes Literal
  - Diese bilden zusammen eine Clique nach Definition des Graphen:
    „Zwei Knoten sind verbunden, wenn sie gleichzeitig erfüllbar sind und nicht in der gleichen Klausel stehen.“

Also: C ist Ja-Instanz von $3SAT \Leftrightarrow (G, K)$ ist Ja-Instanz von $CLIQUE$.

Transformation ist außerdem in Polyzeit machbar $\Rightarrow 3SAT \propto CLIQUE \Rightarrow$ $CLIQUE$ ist $\mathcal{NP}$-schwer.

Da $CLIQUE$ $\mathcal{NP}$-schwer ist und in $\mathcal{NP}$ liegt $\Rightarrow$ $CLIQUE$ $\mathcal{NP}$-vollständig.

$\mathcal{NP}$-Vollständigkeitsbeweis

- Wenn eine n-Clique existiert:
  - n erfüllbare Literale
  - Alle in jeweils unterschiedlichen Klauseln
    (da Literalknoten in derselben Klausel nicht verbunden sind)
  - $\rightsquigarrow$ erfüllende Wahrheitsbelegung
- Wenn eine erfüllende Wahrheitsbelegung existiert:
  - In jeder Klausel mindestens ein Literal erfüllt
  - Nimm aus jeder Klausel ein erfülltes Literal
  - Diese bilden zusammen eine Clique nach Definition des Graphen:
    „Zwei Knoten sind verbunden, wenn sie gleichzeitig erfüllbar sind und nicht in der gleichen Klausel stehen.“

Also: C ist Ja-Instanz von $3SAT \Leftrightarrow (G, K)$ ist Ja-Instanz von $CLIQUE$.

Transformation ist außerdem in Polyzeit machbar $\Rightarrow 3SAT \propto CLIQUE \Rightarrow$ $CLIQUE$ ist $\mathcal{NP}$-schwer.

Da $CLIQUE$ $\mathcal{NP}$-schwer ist und in $\mathcal{NP}$ liegt $\Rightarrow$ $CLIQUE$ $\mathcal{NP}$-vollständig.

$\mathcal{NP}$-Vollständigkeitsbeweis

- Wenn eine n-Clique existiert:
  - n erfüllbare Literale
  - Alle in jeweils unterschiedlichen Klauseln
    (da Literalknoten in derselben Klausel nicht verbunden sind)
  - $\rightsquigarrow$ erfüllende Wahrheitsbelegung
- Wenn eine erfüllende Wahrheitsbelegung existiert:
  - In jeder Klausel mindestens ein Literal erfüllt
  - Nimm aus jeder Klausel ein erfülltes Literal
  - Diese bilden zusammen eine Clique nach Definition des Graphen:
    „Zwei Knoten sind verbunden, wenn sie gleichzeitig erfüllbar sind und nicht in der gleichen Klausel stehen.“

Also: C ist Ja-Instanz von $3SAT \Leftrightarrow (G, K)$ ist Ja-Instanz von $CLIQUE$.

Transformation ist außerdem in Polyzeit machbar $\Rightarrow 3SAT \propto CLIQUE \Rightarrow$ $CLIQUE$ ist $\mathcal{NP}$-schwer.

Da $CLIQUE$ $\mathcal{NP}$-schwer ist und in $\mathcal{NP}$ liegt $\Rightarrow$ $CLIQUE$ $\mathcal{NP}$-vollständig.

$\mathcal{NP}$-Vollständigkeitsbeweis

- Wenn eine n-Clique existiert:
  - n erfüllbare Literale
  - Alle in jeweils unterschiedlichen Klauseln
    (da Literalknoten in derselben Klausel nicht verbunden sind)
  - $\rightsquigarrow$ erfüllende Wahrheitsbelegung
- Wenn eine erfüllende Wahrheitsbelegung existiert:
  - In jeder Klausel mindestens ein Literal erfüllt
  - Nimm aus jeder Klausel ein erfülltes Literal
  - Diese bilden zusammen eine Clique nach Definition des Graphen:
    „Zwei Knoten sind verbunden, wenn sie gleichzeitig erfüllbar sind und nicht in der gleichen Klausel stehen.“

Also: C ist Ja-Instanz von $3SAT \Leftrightarrow (G, K)$ ist Ja-Instanz von $CLIQUE$.

Transformation ist außerdem in Polyzeit machbar $\Rightarrow 3SAT \propto CLIQUE \Rightarrow$ $CLIQUE$ ist $\mathcal{NP}$-schwer.

Da $CLIQUE$ $\mathcal{NP}$-schwer ist und in $\mathcal{NP}$ liegt $\Rightarrow$ $CLIQUE$ $\mathcal{NP}$-vollständig.

$\mathcal{NP}$-Vollständigkeitsbeweis

- Wenn eine n-Clique existiert:
  - n erfüllbare Literale
  - Alle in jeweils unterschiedlichen Klauseln
    (da Literalknoten in derselben Klausel nicht verbunden sind)
  - $\rightsquigarrow$ erfüllende Wahrheitsbelegung
- Wenn eine erfüllende Wahrheitsbelegung existiert:
  - In jeder Klausel mindestens ein Literal erfüllt
  - Nimm aus jeder Klausel ein erfülltes Literal
  - Diese bilden zusammen eine Clique nach Definition des Graphen:
    „Zwei Knoten sind verbunden, wenn sie gleichzeitig erfüllbar sind und nicht in der gleichen Klausel stehen.“

Also: C ist Ja-Instanz von $3SAT \Leftrightarrow (G, K)$ ist Ja-Instanz von $CLIQUE$.

Transformation ist außerdem in Polyzeit machbar $\Rightarrow 3SAT \propto CLIQUE \Rightarrow$ $CLIQUE$ ist $\mathcal{NP}$-schwer.

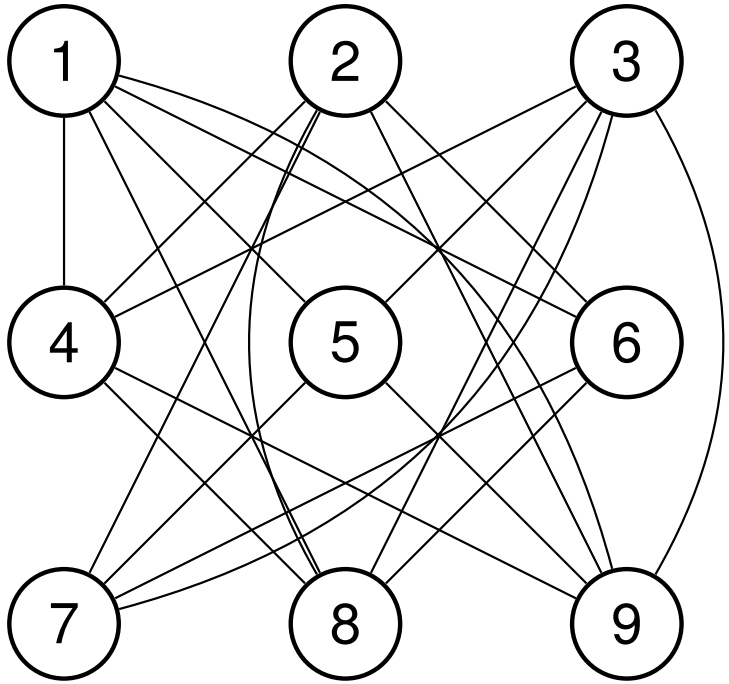
Da $CLIQUE$ $\mathcal{NP}$-schwer ist und in $\mathcal{NP}$ liegt $\Rightarrow$ $CLIQUE$ $\mathcal{NP}$-vollständig.

Beispiel

Reduktion folgender 3*SAT*-Instanz auf eine *CLIQUE*-Instanz:

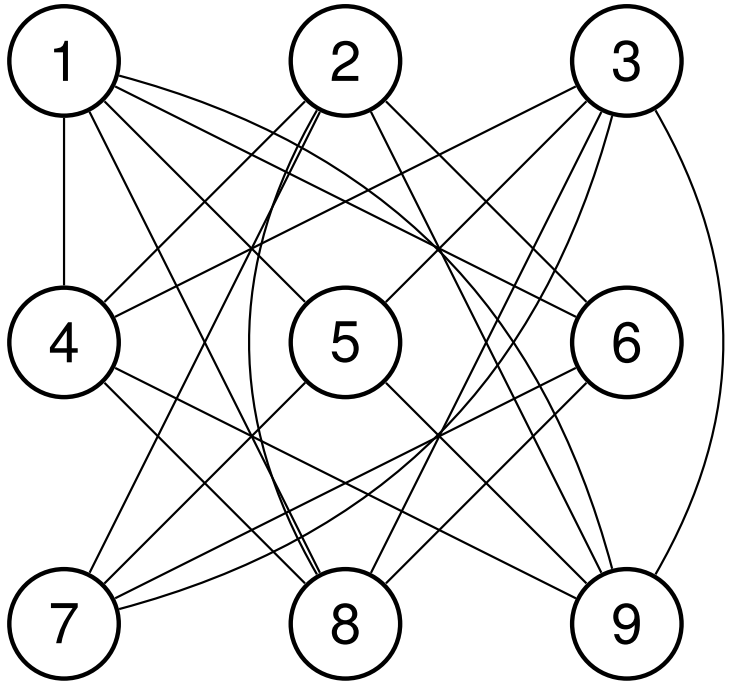
$$C = \{(x_1 \vee x_2 \vee x_3), (x_1 \vee \overline{x_2} \vee \overline{x_3}), (\overline{x_1} \vee x_2 \vee x_3)\}$$

CLIQUE-Beispiel



- Hat dieser Graph eine 3-CLIQUE?

CLIQUE-Beispiel



- Hat dieser Graph eine 3-CLIQUE?
- Hat dieser Graph eine 4-CLIQUE?

$\mathcal{NP}$-Vollständigkeit eines Problems zeigen

Karlsruher Institut für Technologie

Gegeben: Entscheidungsproblem Π
Aufgabe: Zeige, dass Π $\mathcal{NP}$-vollständig ist. [evtl: Benutze hierzu die $\mathcal{NP}$-Vollständigkeit von X.]

Lösung

1. Zeige: $\Pi \in \mathcal{NP}$ (!)
2. Finde ein passendes $\mathcal{NP}$-vollständiges Problem X bzw. wähle das X aus der Aufgabenstellung.
3. „Sei eine Instanz I von X gegeben. Konstruiere daraus eine Instanz I' von Π wie folgt: . . .“ $\rightarrow$ Konstruktion beschreiben
4. Zeige, dass Ja-Instanzen genau auf Ja-Instanzen abgebildet werden.
5. Erwähne, dass die Konstruktion in **polynomieller Zeit** geht.

Allgemeines zu $\mathcal{NP}$-Vollständigkeitsbeweisen

In der Regel ist es schwer zu zeigen, dass ein Problem $\mathcal{NP}$-schwer ist, aber leicht zu zeigen, dass ein Problem in $\mathcal{NP}$ liegt.

Da man üblicherweise ein $\mathcal{NP}$-vollständiges Problem als Vorausetzung für die Reduktion benötigt (Ausnahme: Satz von Cook), lohnt es sich einige kennen zu lernen.

Bis zum nächsten Mal!

Lizenzen