

SAT ist NP-vollständig (cont)

- → Es **gibt NP**-vollständige Probleme.
- **Reduktion** $L \leq_p \text{SAT}$ ist strukturerhaltend: löst zwar das Problem $x \in L$ nicht, aber wandelt es in andere Form um. → **Richtung der Reduktion!**
- Die SAT **Instanzen**, die bei der Reduktion entstehen können, sind nur eine kleine Teilmenge der möglichen SAT Formeln. Da aber manche von ihnen vermutlich **schwierig** sind, enthält das **allgemeine Problem SAT** eben schwierige Problem, und ist daher **schwierig**.
- Entscheidungs- vs. **Funktions-Probleme**:
wenn eines **NP**-vollständig, dann auch das andere.
→ Erweiterung des Begriffs der **NP**-Vollständigkeit.

Alternative (äquivalente) Definition von NP

Definition: $L \in \mathbf{NP}$ genau dann wenn $\exists M$ polynomiell-zeitbeschränkte TM und $\exists c \in \mathbb{N}$ / $\forall x \in \Sigma^*$

$$x \in L \Leftrightarrow \left(\exists y \in \Sigma^* \quad |y| = O(|x|^c) \wedge M(x, y) = \mathbf{wahr} \right)$$

y : polynomiell beschränkter und polynomiell überprüfbarer Beweis (auch: Zertifikat) auf deterministischer TM.

- \rightarrow Polynomielle Überprüfbarkeit eines Beweises statt Nichtdeterminismus
- NP-hart, NP-Vollständigkeit, ... : Terminologie geht auf Knuth (T_EX, The Art of Computer Programming) zurück
- \rightarrow PET-Probleme (Lin; Knuth 1974)
- Knuth: polynomielle Transformation, aber \rightarrow polynomielle Reduktion hat sich doch durchgesetzt

Weitere NP-vollständige Probleme

→ Ab nun sind die Beweise **einfacher**: $L \in \mathbf{NP}$ zusammen mit $\mathbf{SAT} \leq_p L$ (oder jede andere **NP**-vollständige Sprache) reichen, um die **NP-Vollständigkeit von L** zu beweisen.

- Man zeigt dadurch, daß man ein **beliebiges** Problem in **NP** lösen könnte, indem man es (mit **nur geringem Mehraufwand**) auf die Frage $x \in L ?$ reduziert.
- L muß also mindestens **genauso schwer** wie die **schwersten** Probleme in **NP** sein.
- Denn wäre L **leicht** (= in polynomieller Zeit zu beantworten), so könnte man ja **jedes** Problem in **NP** (also auch die **schwersten** darunter) **schnell** (polynomiell) in eine Frage $x \in L ?$ umwandeln, darauf **schnell** die Antwort finden, und das ursprüngliche angeblich schwere Problem so **leicht** lösen → **Widerspruch**.

Definition: 3CNF (auch 3SAT genannt)

Gegeben: Eine Boolesche Formel F in konjunktiver Normalform (CNF) mit höchstens 3 Literalen pro Klausel.

Beispiel: $(a) \wedge (b) \wedge (\neg a \vee \neg b \vee \neg c) \wedge (\neg a \vee c)$.

Gefragt: Ist F erfüllbar?

Theorem: 3CNF ist NP-vollständig.

Beweis:

Membership: Guess und check Argument ✓

Hardness: Wir zeigen $\text{SAT} \leq_p \text{3CNF}$.

Das bedeutet: Wir müssen ein polynomielles Verfahren angeben, das eine beliebige Boolesche Formel F in eine 3CNF Formel F' umformt, sodaß gilt

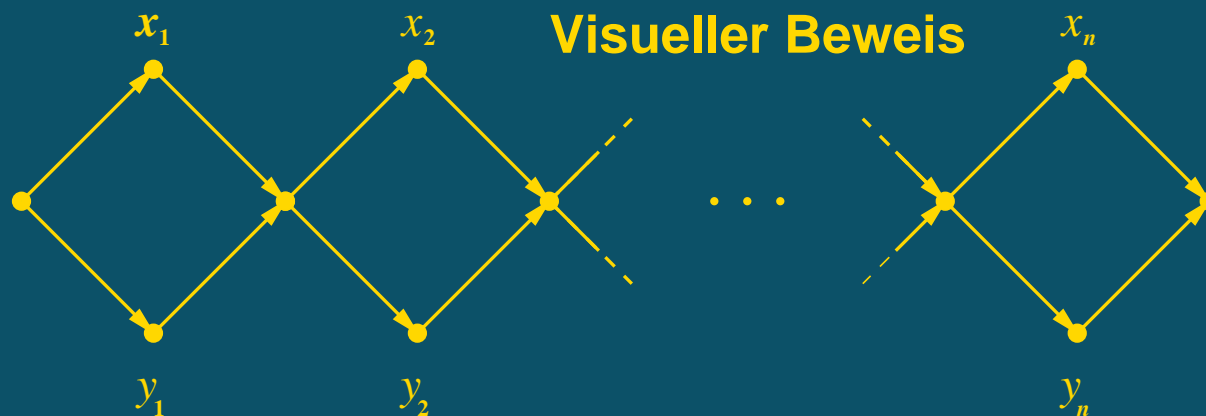
F ist erfüllbar $\Leftrightarrow F'$ ist erfüllbar.



Erstes Problem: Äquivalente Umgewandlung CNF \leftrightarrow DNF hat i.a. **exponentiellen** Aufwand, weiters werden nicht notwendigerweise Klauseln mit nur 3 Literalen erzeugt.

Beispiel: $(x_1 \vee y_1) \wedge (x_2 \vee y_2) \wedge \cdots \wedge (x_n \vee y_n) \equiv$

$$\left. \begin{array}{l} (x_1 \wedge x_2 \wedge \cdots \wedge x_{n-1} \wedge x_n) \\ \vee (y_1 \wedge x_2 \wedge \cdots \wedge x_{n-1} \wedge x_n) \\ \vee (x_1 \wedge y_2 \wedge \cdots \wedge x_{n-1} \wedge x_n) \\ \vdots \\ \vee (x_1 \wedge y_2 \wedge \cdots \wedge y_{n-1} \wedge y_n) \\ \vee (y_1 \wedge y_2 \wedge \cdots \wedge y_{n-1} \wedge y_n) \end{array} \right\} 2^n$$



Lösung: Wir zeigen nur **Erfüllbarkeitsäquivalenz** und führen dazu **neue** Variable ein. Die Umformung geschieht in mehreren Schritten. **Beispiel:** $F = \neg(\neg(x_1 \vee \neg x_3) \vee x_2)$.

1. Anwendung der Regeln von **DeMorgan**, um alle Negationszeichen zu den Variablen zu verschieben.

$$\rightarrow ((x_1 \vee \neg x_3) \wedge \neg x_2). \quad \text{Aufwand: } O(n).$$

2. Wir ordnen jedem \wedge und \vee eine **neue** Variable $\in \{y_0, y_1, \dots\}$ zu.

$$\rightarrow ((x_1 \overset{y_1}{\vee} \neg x_3) \overset{y_0}{\wedge} \neg x_2). \quad \text{Aufwand: } O(n^3).$$

3. Wir klammern so um, daß nur noch **binäre** Ausdrücke mit \wedge und \vee vorhanden sind und ordnen jedem solchen Ausdruck $(a \overset{y_j}{\circ} b)$ mit $\circ \in \{\wedge, \vee\}$ und $a, b \in \{x_1, \dots, x_n, y_0, y_1, \dots\}$, eine Teilformel der Form

$$(y_j \leftrightarrow (a \circ b))$$



zu. Alle diese Formeln sowie y_0 werden mit \wedge zu einer neuen Formel F_1 verknüpft.

$$\rightarrow F_1 = [y_0] \wedge [y_0 \leftrightarrow (y_1 \wedge \neg x_2)] \wedge [y_1 \leftrightarrow (x_1 \vee \neg x_3)].$$

Aufwand: $O(n)$.

$\rightarrow F$ und F_1 sind erfüllbarkeitsäquivalent.

4. Jeder Ausdruck [...] in F_1 wird nun in CNF umgeformt:

$$[a \leftrightarrow (b \vee c)] \mapsto (a \vee \neg b) \wedge (\neg a \vee b \vee c) \wedge (a \vee \neg c)$$

$$[a \leftrightarrow (b \wedge c)] \mapsto (\neg a \vee b) \wedge (\neg a \vee c) \wedge (a \vee \neg b \vee \neg c)$$

Aufwand: $O(n)$.

Wir erhalten also die gewünschte 3CNF Formel

$$\rightarrow F' = y_0 \wedge (\neg y_0 \vee y_1) \wedge (\neg y_0 \vee \neg x_2) \wedge (y_0 \vee \neg y_1 \vee x_2) \\ \wedge (y_1 \vee \neg x_1) \wedge (\neg y_1 \vee x_1 \vee \neg x_3) \wedge (y_1 \vee x_3)$$

\rightarrow Gesamtaufwand ist polynomiell, damit haben wir $\text{SAT} \leq_p \text{3CNF}$ gezeigt. \checkmark



3CNF (cont)

- Richtung der Reduktion; Gadget-Strukturerhaltung
- Vergleich mit SAT NP-Vollständigkeitsbeweis
- 4CNF, 5CNF, ...
- 3DNF, ..., 2CNF, HORNSAT

Definition: HORNSAT: Erfüllbarkeit von CNF Formel F , dabei pro Klausel höchstens ein positives Literal.

Beispiel: $(\neg x_1 \vee x_2 \vee \neg x_3)$, $(\neg x_1 \vee \neg x_3)$, und x_2 sind HORN, $(\neg x_1 \vee x_2 \vee x_3)$ ist nicht HORN.

- $(\neg a \vee \neg b \vee \neg c \vee \neg \dots \vee z) \equiv ((a \wedge b \wedge c \wedge \dots) \rightarrow z)$
- $[\neg a \vee \neg b \vee \neg c \vee \neg \dots] \equiv [(a \wedge b \wedge c \wedge \dots) \rightarrow \mathbf{falsch}]$
- $(z) \equiv (\mathbf{wahr} \rightarrow z) \quad \rightsquigarrow \text{Prolog}$



HORNSAT

Theorem: HORNSAT $\in \mathbf{P}$ (daher vermutlich nicht ...)

Beweis: polynomieller \rightarrow

Algorithmus HORNSAT(F)

$W := \{\}$ (* Kumulator für wahre Variablen *)

repeat

if $\exists((x_1 \wedge x_2 \wedge \dots \wedge x_n) \rightarrow y) \in F$, y Variable
sodaß für alle x_i gilt daß $x_i \in W$ aber $y \notin W$,

then $W := W \cup \{y\}$

until W ändert sich nicht mehr (* endet! *)

if $\forall[(x_1 \wedge x_2 \wedge \dots \wedge x_n) \rightarrow \mathbf{falsch}] \in F$
 $\exists x_i$, sodaß $x_i \notin W$ (* endet ebenfalls! *)

then return F ist erfüllbar (durch W)

else return F ist unerfüllbar. \checkmark

Definition: MAX2SAT (auch MAX2CNF genannt)

Gegeben: Eine Formel F in 2CNF und eine Zahl K .

Gefragt: Gibt es eine Wahrheitsbelegung, sodaß mindestens K Klauseln von F erfüllt sind?

Theorem: MAX2SAT ist NP-vollständig.

Beweis:

Membership: Guess und check Argument ✓

Hardness: Wir zeigen $3\text{CNF} \leq_p \text{MAX2SAT}$.

Das bedeutet: Wir müssen ein polynomielles Verfahren angeben, das eine beliebige 3CNF Formel F in ein MAX2SAT-Problem $\langle F', K \rangle$ umformt, sodaß gilt

F ist erfüllbar \Leftrightarrow mind. K Klauseln von F' sind erfüllbar. ▷

MAX2SAT (cont)

Sei $F \stackrel{\text{def}}{=} \bigwedge_{i=1}^m (x_i \vee y_i \vee z_i)$ eine beliebige 3CNF Formel,
notfalls mit $x_i = z_i$, oder sogar $x_i = y_i$.

Dann sei $F' \stackrel{\text{def}}{=} \bigwedge_{i=1}^m G_i$, wobei G_i folgendes Gadget ist:

$$G_i \stackrel{\text{def}}{=} (x_i) \wedge (y_i) \wedge (z_i) \wedge (w_i) \wedge \\ (\neg x_i \vee \neg y_i) \wedge (\neg y_i \vee \neg z_i) \wedge (\neg z_i \vee \neg x_i) \wedge \\ (x_i \vee \neg w_i) \wedge (y_i \vee \neg w_i) \wedge (z_i \vee \neg w_i)$$

- Angenommen, x_i, y_i und z_i sind alle wahr, dann sind 7 Klauseln von G_i erfüllbar.
- Angenommen, x_i und y_i sind wahr, dann sind ebenfalls 7 Klauseln von G_i erfüllbar.
- Angenommen, nur x_i ist wahr, dann sind ebenfalls 7 Klauseln von G_i erfüllbar.



- Wenn x_i , y_i und z_i **falsch** sind, dann sind **nur noch höchstens 6** Klauseln von G_i erfüllbar.
- Wegen der **Symmetrie** der Literale x_i , y_i und z_i deckt dies alle möglichen Fälle ab.
- Daher setzen wir $K \stackrel{\text{def}}{=} 7m$:
 - \Rightarrow) Angenommen, F ist erfüllbar. Dann ist in jedem $(x_i \vee y_i \vee z_i)$ in F mindestens ein Literal **wahr** und daher in **jedem** G_i **7** Klauseln erfüllbar, in ganz F' also **$7m$** .
 - \Leftarrow) Falls in F' **$7m$** Klauseln erfüllbar sind, dann muß in **jedem** G_i mindestens ein Literal aus x_i , y_i und z_i **wahr** sein, und daher muß F erfüllbar sein.
- Da die Formel F' genau $10m$ Klauseln mit maximaler Größe 2 hat, ist die Konstruktion sicher in **polynomieller Zeit** durchführbar. ✓

Definition: CLIQUE (Maxclique, vollständiger Subgraph)

Gegeben: (ungerichteter) Graph $G = (V, E)$, $k \in \mathbb{N}$

Gefragt: Besitzt G eine Clique der Größe k ? (Eine Clique der Größe k ist eine Teilmenge V' der Knotenmenge V mit $|V'| = k$, sodaß $\forall u, v \in V'$ gilt: $\{u, v\} \in E$).

Theorem: CLIQUE ist NP-vollständig.

Beweis:

Membership: Guess und check Argument ✓

Hardness: Wir zeigen $3\text{CNF} \leq_p \text{CLIQUE}$.

Das bedeutet: Wir müssen ein polynomielles Verfahren angeben, das eine beliebige 3CNF Formel F in ein CLIQUE-Problem $\langle G, k \rangle$ umformt, sodaß gilt

F ist erfüllbar $\Leftrightarrow G$ hat eine Clique der Größe k



CLIQUE (cont)

Sei $F \stackrel{\text{def}}{=} \bigwedge_{i=1}^m (z_{i,1} \vee z_{i,2} \vee z_{i,3})$ eine beliebige 3CNF Formel (notfalls aufgestockt wie beim MAX2SAT Beweis, um auf exakt 3 Literale pro Klausel zu kommen), wobei

$$z_{i,j} \in \{x_1, x_2, \dots\} \cup \{\neg x_1, \neg x_2, \dots\}$$

Ex: $F = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$

Dann sei F ein Graph $G = (V, E)$ und $k \in \mathbb{N}$ zugeordnet:

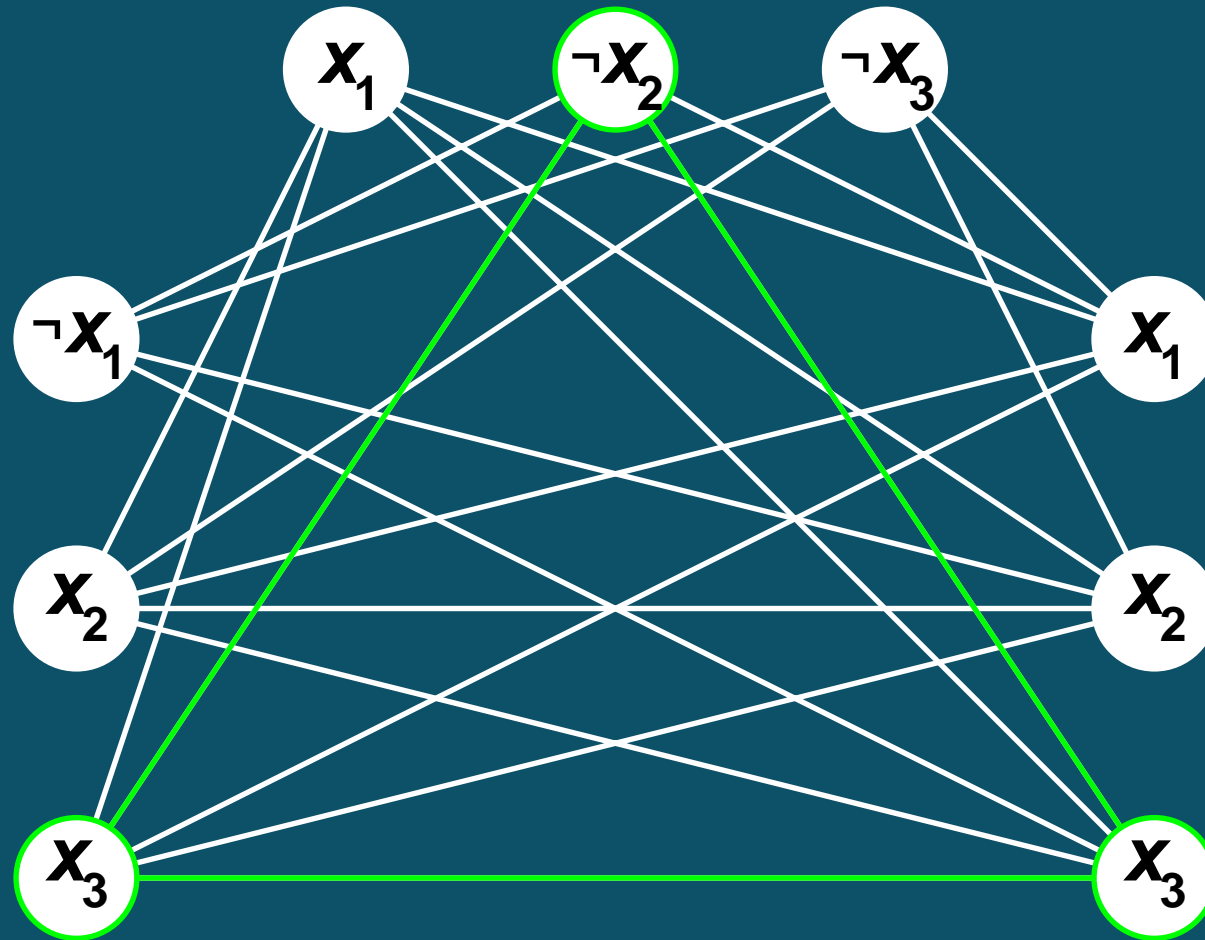
$$V \stackrel{\text{def}}{=} \{(1, 1), (1, 2), (1, 3), \dots, (m, 1), (m, 2), (m, 3)\}$$

$$E \stackrel{\text{def}}{=} \left\{ \{(i, j), (p, q)\} \mid i \neq p \text{ und } z_{i,j} \neq \neg z_{p,q} \right\}$$

$$k \stackrel{\text{def}}{=} m$$



Ex: $F = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$
wird zB durch $x_1 = \text{falsch}$, $x_2 = \text{falsch}$ und $x_3 = \text{wahr}$ erfüllt. Eine entsprechende Clique:



CLIQUE (cont)

Es gilt nun: F ist erfüllbar durch eine Belegung B

- \Leftrightarrow es in jeder Klausel ein Literal gibt, das unter B den Wert **wahr** hat, zB $z_{1,j_1}, z_{2,j_2}, \dots, z_{m,j_m}$
- \Leftrightarrow es Literale $z_{1,j_1}, z_{2,j_2}, \dots, z_{m,j_m}$ gibt, die paarweise **nicht komplementär** ($z_{i,j} \neq \neg z_{p,q}$) sind.
- \Leftrightarrow es Knoten $(1, j_1), (2, j_2), \dots, (m, j_m)$ in G gibt, die **paarweise verbunden** ($z_{i,j} \neq \neg z_{p,q}$) sind.
- \Leftrightarrow es eine **Clique der Größe $k = m$** in G gibt.

Die Konstruktion ist sicher polynomiell machbar. ✓

Definition: INDEPENDENT SET (Max ...)

Gegeben: (ungerichteter) Graph $G = (V, E)$, $k \in \mathbb{N}$

Gefragt: Besitzt G ein Independent Set der Größe k ?
(Ein Independent Set der Größe k ist eine Teilmenge V' der Knotenmenge V mit $|V'| = k$, sodaß $\forall u, v \in V'$ gilt: $\{u, v\} \notin E$).

Theorem: INDEPENDENT SET ist NP-vollständig.

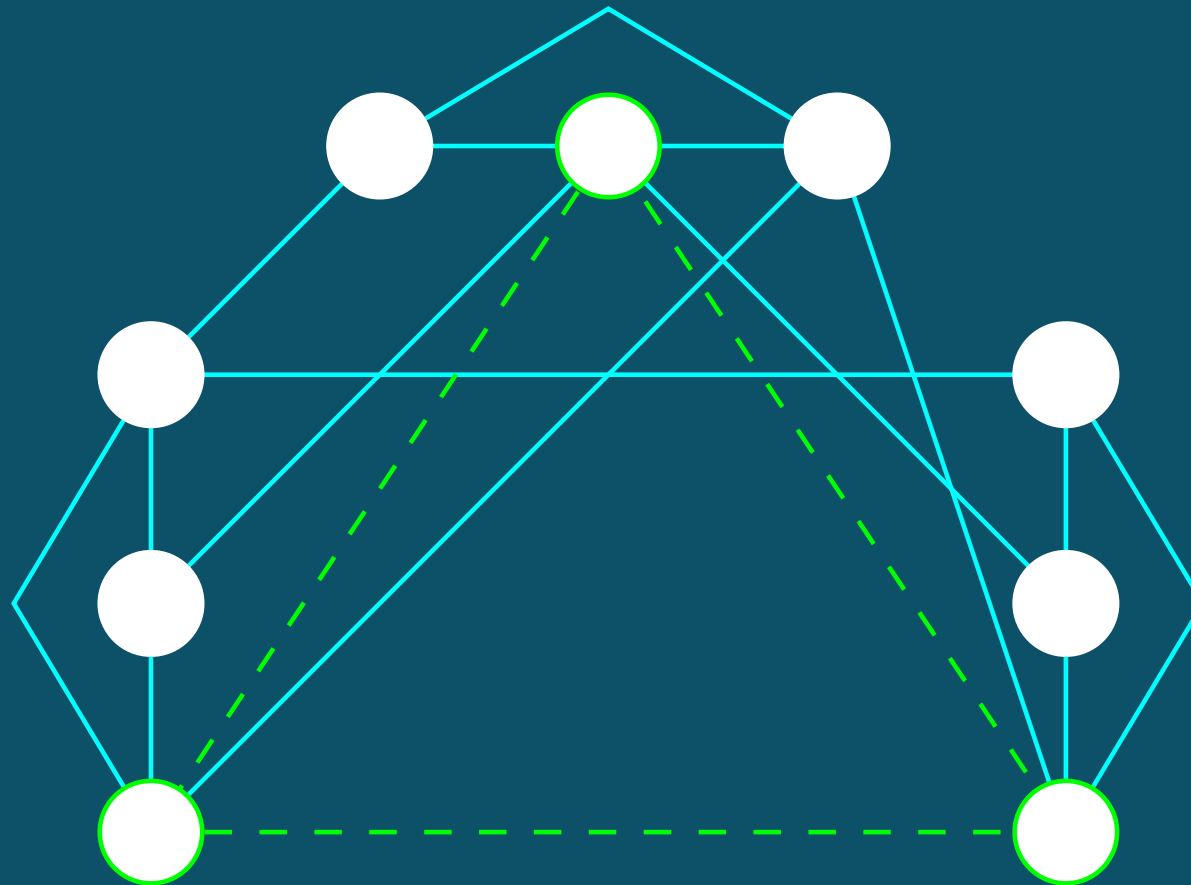
Beweis:

Membership: Guess und check Argument ✓

Hardness: CLIQUE \leq_p INDEPENDENT SET.

Bemerkung: Es genügt, den Graphen $G = (V, E)$ des CLIQUE-Problems $\langle G, k \rangle$ zum Graphen $G' = (V, \bar{E})$ zu invertieren ($\{u, v\} \in E \Leftrightarrow \{u, v\} \notin \bar{E}$), um das INDEPENDENT SET-Problems $\langle G', k \rangle$ zu bekommen. ✓ ▷

Invertierter Beispielgraph aus dem Beweis für Clique mit Independent Set der Größe 3:



Definition: VERTEX COVER (Node Cover, Min ...)

Gegeben: (ungerichteter) Graph $G = (V, E)$, $k \in \mathbb{N}$

Gefragt: Besitzt G einen Vertex Cover der Größe k ? (Ein Vertex Cover der Größe k ist eine Teilmenge $V' \subseteq V$ mit $|V'| = k$, sodaß $\forall \{u, v\} \in E$ gilt: $u \in V'$ oder $v \in V'$).

Theorem: VERTEX COVER ist NP-vollständig.

Beweis:

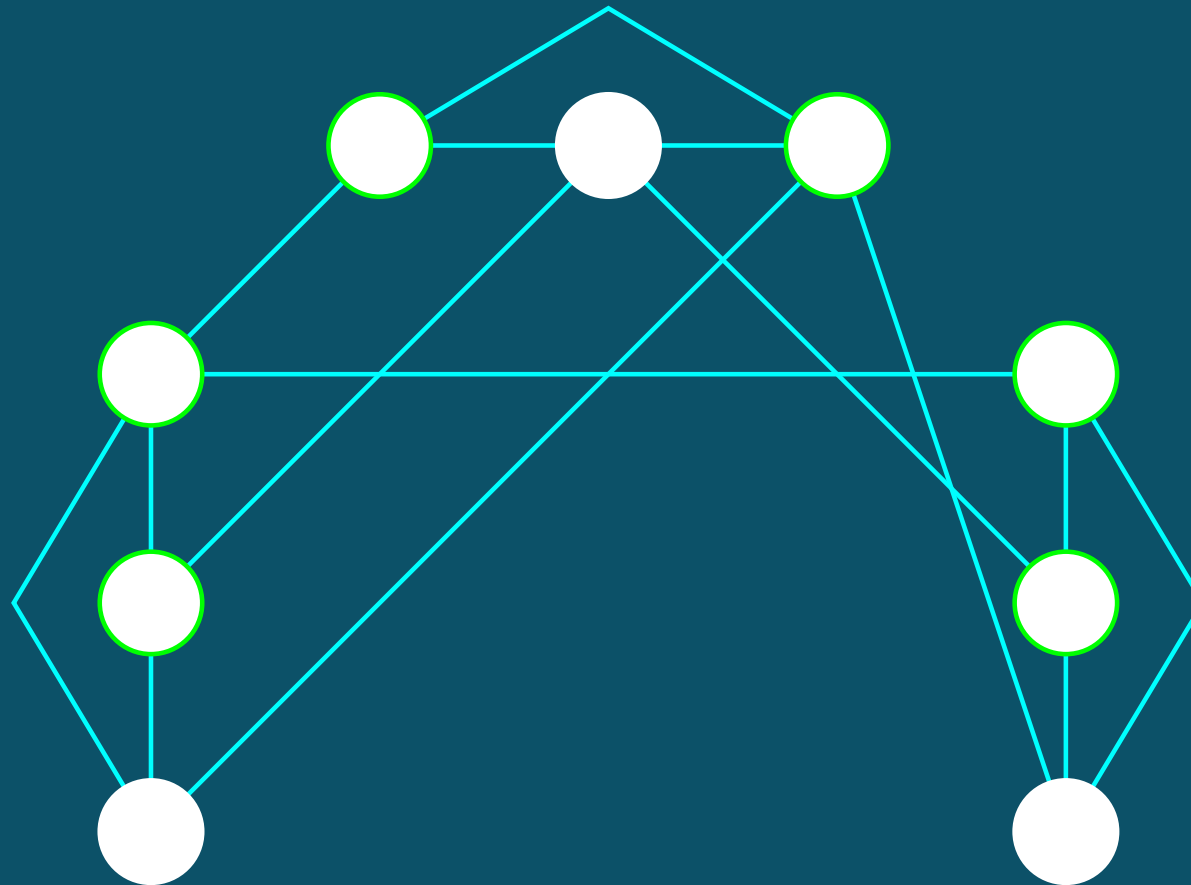
Membership: Guess und check Argument ✓

Hardness: INDEPENDENT SET \leq_p VERTEX COVER.

Bemerkung: I ist ein Independent Set der Größe i des Graphen $G = (V, E) \Leftrightarrow V' \stackrel{\text{def}}{=} V - I$ ist ein Vertex Cover der Größe $k \stackrel{\text{def}}{=} |V| - i$ von G . ✓



Beispielgraph aus Beweis für Independent Set mit **Vertex Cover** der Größe 6:



Definition: SUBSET SUM (Knapsack, Rucksack)

Gegeben: Zahlen $a_1, a_2, \dots, a_k \in \mathbb{N}$ und $b \in \mathbb{N}$

Gefragt: Gibt es $I \subseteq \{1, 2, \dots, k\}$ mit $\sum_{i \in I} a_i = b$?

Theorem: SUBSET SUM ist NP-vollständig.

Beweis:

Membership: Guess und check Argument ✓

Hardness: $3\text{CNF} \leq_p \text{SUBSET SUM}$.

Sei $F \stackrel{\text{def}}{=} \bigwedge_{i=1}^m (z_{i,1} \vee z_{i,2} \vee z_{i,3})$ eine beliebige 3CNF Formel
wobei $z_{i,j} \in \{x_1, x_2, \dots, x_n\} \cup \{\neg x_1, \neg x_2, \dots, \neg x_n\}$.

Dann ist die Zahl b gegeben durch (zb im Dezimalsystem)

$$b \stackrel{\text{def}}{=} \underbrace{444 \dots 444}_m \underbrace{11 \dots 11}_n$$

Falls zB F aus 3 Klauseln besteht und darin 5 Variablen vorkommen, so ist $b = 444 11111$.



$$\text{zB } F = (x_1 \vee \neg x_3 \vee x_5) \wedge (\neg x_1 \vee x_5 \vee x_4) \wedge (\neg x_2 \vee \neg x_2 \vee \neg x_5)$$

Wir müssen noch die Zahlen a_i definieren (4 Klassen):

- (1) Positive Variablenvorkommen (inkl. Anzahl):

$$v_1 = 100 \ 10000$$

$$v_2 = 000 \ 01000$$

$$v_3 = 000 \ 00100$$

$$v_4 = 010 \ 00010$$

$$v_5 = 110 \ 00001$$

- (2) Negative Variablenvorkommen (inkl. Anzahl):

$$v'_1 = 010 \ 10000$$

$$v'_2 = 002 \ 01000$$

$$v'_3 = 100 \ 00100$$

$$v'_4 = 000 \ 00010$$

$$v'_5 = 001 \ 00001$$



- (3) Ausgleichszahlen (falls ein Literal **falsch**):

$$c_1 = 100\ 00000$$

$$c_2 = 010\ 00000$$

$$c_3 = 001\ 00000$$

- (4) Ausgleichszahlen (falls zwei Literale **falsch**):

$$d_1 = 200\ 00000$$

$$d_2 = 020\ 00000$$

$$d_3 = 002\ 00000$$

- Es gibt **keine** Ausgleichszahlen für den Fall, daß drei Literale **falsch** sind :-)

- \exists erfüllende Belegung $B \iff$

\exists Auswahl A der Zahlen, die sich zu b aufsummiert:

$$x_i \text{ in } B \text{ wahr} \iff v_i \text{ in } A$$

$$x_i \text{ in } B \text{ falsch} \iff v_i' \text{ in } A$$



- Da hinterer Teil von $b = 444\ 11111$ nur aus **Einsen** besteht kann nur **entweder** v_i **oder** v'_i in A sein; einer davon **muß** aber drin sein, da die anderen Variablen an der entsprechenden Stelle alle eine Null stehen haben.
- Bsp: $B = \{x_1, x_4\}$ (andere **falsch**) ergibt Auswahl $v_1, v'_2, v'_3, v_4, v'_5$. Summe davon ist **213 11111** (wichtig: im vorderen Teil sind alle größer als Null, da die Ausgleichszahlen nur Zahlen größer als Null ausgleichen können, und hinten sind alle Eins, dh alle Variablen haben einen Wert zugewiesen bekommen).
- Durch Hinzunehmen von geeigneten Ausgleichszahlen, in diesem Fall d_1, c_2, d_2, c_3 , erhalten wir die gewünschte Summe $b = 444\ 11111$.
- Da es **keine Überträge** zwischen den Stellen gibt, „funktioniert“ der Beweis in beide Richtungen. ✓