

# Komplexitätstheorie Teil II: Eine Auswahl $\mathcal{NP}$ -vollständiger Probleme

Prof. Dr. David Sabel

LFE Theoretische Informatik



- Auswahl an  $\mathcal{NP}$ -vollständigen Problemen
- $\mathcal{NP}$ -Vollständigkeitsbeweise durch polynomielle Reduktionen und zeigen, dass das Problem in  $\mathcal{NP}$  ist

## Konjunktive Normalform

Aussagenlog. Formel  $F$  ist in **konjunktiver Normalform** (CNF), wenn:

$$F = \bigwedge_{i=1}^m \left( \bigvee_{j=1}^{n_i} L_{i,j} \right)$$

- **Literal**  $L_{i,j}$  ist aussagenlogische Variable oder deren Negation
- $\left( \bigvee_{j=1}^{n_i} L_{i,j} \right)$  ist eine **Klausel**
- Erfüllende Belegung muss  $\geq 1$  Literal pro Klausel wahr machen.

Klauselmengenschreibweise:  $\{ \{ L_{1,1}, \dots, L_{1,n_1} \}, \dots, \{ L_{m,1}, \dots, L_{m,n_m} \} \}$

Annahme: Keine Klauseln, die  $x$  und  $\neg x$  enthalten

Diese Klauseln sind immer wahr und können gelöscht werden.

# Konjunktive Normalform

---

Jede aussagenlogische Formel kann in eine **äquivalente** konjunktive Normalform gebracht werden:

- $\iff$  ,  $\implies$  auflösen
- Negationen nach innen schieben und anschließend Ausmultiplizieren (Distributivität, Kommutativität anwenden, um konjunktive Normalform herzustellen)

Aber:

- Algorithmus hat im worst case **exponentielle Laufzeit**
- Algorithmus kann daher **nicht** für eine **Polynomialzeit**reduktion verwendet werden.

## Definition (3-CNF-SAT)

Das **3-CNF-SAT-Problem** lässt sich in der gegeben/gefragt-Notation formulieren als:

gegeben: Eine aussagenlogische Formel  $F$  in konjunktiver Normalform, sodass jede Klausel höchstens 3 Literale enthält.

gefragt: Ist  $F$  erfüllbar? Genauer: Gibt es eine erfüllende Belegung der Variablen mit den Wahrheitswerten 0 und 1, sodass  $F$  den Wert 1 erhält?

## Satz

3-CNF-SAT ist  $\mathcal{NP}$ -vollständig.

Beweis, Teil 1: 3-CNF-SAT ist in  $\mathcal{NP}$

- Rate nicht-deterministisch eine Belegung der Variablen
- Prüfe deterministisch, ob die Belegung die 3-CNF wahr macht. Akzeptiere in diesem Fall. Dies geht in Polynomialzeit in der Größe der 3-CNF.
- Daher kann 3-CNF-SAT auf einer NTM in Polynomialzeit entschieden werden.

## $\mathcal{NP}$ -Vollständigkeit von 3-CNF-SAT (2)

---

Beweis, Teil 2: 3-CNF-SAT ist  $\mathcal{NP}$ -hart

- Wir zeigen  $\text{SAT} \leq_p \text{3-CNF-SAT}$ .
- Gesucht: Polynomiell berechenbare, totale Funktion  $f$ , sodass  $F$  erfüllbar g.d.w. 3-CNF  $f(F)$  erfüllbar.

## $\mathcal{NP}$ -Vollständigkeit von 3-CNF-SAT (2)

Beweis, Teil 2: 3-CNF-SAT ist  $\mathcal{NP}$ -hart

- Wir zeigen  $\text{SAT} \leq_p \text{3-CNF-SAT}$ .
- Gesucht: Polynomiell berechenbare, totale Funktion  $f$ , sodass  $F$  erfüllbar g.d.w. 3-CNF  $f(F)$  erfüllbar.
- $f$  muss die **Erfüllbarkeit erhalten**, aber nicht die **Äquivalenz**
- Verfahren, um  $F$  in erfüllbarkeitsäquivalente 3-CNF  $f(F)$  umzuformen, sodass  $f(F)$  **polynomielle Größe** in  $|F|$  hat:

Tseitin-Transformation



## $\mathcal{NP}$ -Vollständigkeit von 3-CNF-SAT (3)

Tseitin-Transformation:

- Schiebe alle Negationen nach innen vor die Literale, dabei werden die Regeln  $\neg\neg F \rightarrow F$ ,  $\neg(F \wedge G) \rightarrow \neg F \vee \neg G$ ,  $\neg(F \vee G) \rightarrow \neg F \wedge \neg G$ ,  $\neg(F \iff G) \rightarrow (\neg F) \iff G$  und  $\neg(F \implies G) \rightarrow F \wedge \neg G$  angewendet.

## $\mathcal{NP}$ -Vollständigkeit von 3-CNF-SAT (3)

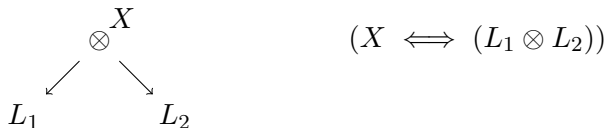
Tseitin-Transformation:

- Schiebe alle Negationen nach innen vor die Literale, dabei werden die Regeln  $\neg\neg F \rightarrow F$ ,  $\neg(F \wedge G) \rightarrow \neg F \vee \neg G$ ,  $\neg(F \vee G) \rightarrow \neg F \wedge \neg G$ ,  $\neg(F \iff G) \rightarrow (\neg F) \iff G$  und  $\neg(F \implies G) \rightarrow F \wedge \neg G$  angewendet.
- Syntaxbaum der Formel (Blätter = Literale):  
Für jeden Nichtblatt-Knoten: **neue aussagenlogische Variable**

## $\mathcal{NP}$ -Vollständigkeit von 3-CNF-SAT (3)

Tseitin-Transformation:

- Schiebe alle Negationen nach innen vor die Literale, dabei werden die Regeln  $\neg\neg F \rightarrow F$ ,  $\neg(F \wedge G) \rightarrow \neg F \vee \neg G$ ,  $\neg(F \vee G) \rightarrow \neg F \wedge \neg G$ ,  $\neg(F \iff G) \rightarrow (\neg F) \iff G$  und  $\neg(F \implies G) \rightarrow F \wedge \neg G$  angewendet.
- Syntaxbaum der Formel (Blätter = Literale):  
Für jeden Nichtblatt-Knoten: **neue aussagenlogische Variable**
- pro Gabelung: erzeuge

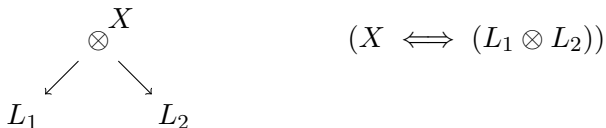


wobei  $\otimes \in \{ \iff, \wedge, \vee, \implies \}$  und  $L_1, L_2$  entweder die neue Variable oder das Literal am Blatt.

## $\mathcal{NP}$ -Vollständigkeit von 3-CNF-SAT (3)

Tseitin-Transformation:

- Schiebe alle Negationen nach innen vor die Literale, dabei werden die Regeln  $\neg\neg F \rightarrow F$ ,  $\neg(F \wedge G) \rightarrow \neg F \vee \neg G$ ,  $\neg(F \vee G) \rightarrow \neg F \wedge \neg G$ ,  $\neg(F \iff G) \rightarrow (\neg F) \iff G$  und  $\neg(F \implies G) \rightarrow F \wedge \neg G$  angewendet.
- Syntaxbaum der Formel (Blätter = Literale):  
Für jeden Nichtblatt-Knoten: **neue aussagenlogische Variable**
- pro Gabelung:  $X$  erzeuge



wobei  $\otimes \in \{ \iff, \wedge, \vee, \implies \}$  und  $L_1, L_2$  entweder die neue Variable oder das Literal am Blatt.

- Konjugiere diese Formeln zu  $F'$  und schließlich erzeuge  $W \wedge F'$ , mit  $W$  Variable für die Wurzel.

## $\mathcal{NP}$ -Vollständigkeit von 3-CNF-SAT (3)

---

### Tseitin-Transformation (Fortsetzung)

- Insgesamt:  $W \wedge \bigwedge_{i,j,k} (X_i \iff (X_j \otimes_i X_k))$ , wobei  $X_r$  Literale sind.
- Berechne für jede Subformel  $(X_i \iff (X_j \otimes_i X_k))$  die CNF mit dem üblichen Algorithmus.
- Lösche doppelte Vorkommen von Literalen.
- Ergibt 3-CNF, da pro Klausel nur 3 verschiedene Variablen vorkommen können.

## $\mathcal{NP}$ -Vollständigkeit von 3-CNF-SAT (4)

---

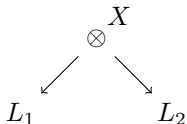
Komplexität:

- Größe der kleinen CNFs ist konstant.
- Jede Klausel hat nur 3 Literale: Mehr Variablen gibt es in den kleinen Formeln nicht, doppelte Literale löschen
- Größe der 3-CNF ist polynomiell in der ursprünglichen Formel
- Berechnung in Polynomialzeit: klar!

## $\mathcal{NP}$ -Vollständigkeit von 3-CNF-SAT (5)

$F$  erfüllbar g.d.w.  $f(F)$  erfüllbar:

- „ $\Rightarrow$ “: Sei  $I$  Belegung mit  $I(F) = 1$ . Sei  $I'$  Belegung mit
  - $I'(X) = I(X)$  für alle Variablen, die in  $F$  vorkommen
  - $I'(W) = 1$  für die Variable  $W$  an der Wurzel
  - $I'(X) = I'(L_1 \otimes L_2)$  für

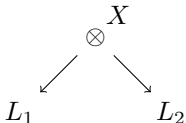


Dann gilt:  $I'(f(F)) = 1$

## $\mathcal{NP}$ -Vollständigkeit von 3-CNF-SAT (5)

$F$  erfüllbar g.d.w.  $f(F)$  erfüllbar:

- „ $\Rightarrow$ “: Sei  $I$  Belegung mit  $I(F) = 1$ . Sei  $I'$  Belegung mit
  - $I'(X) = I(X)$  für alle Variablen, die in  $F$  vorkommen
  - $I'(W) = 1$  für die Variable  $W$  an der Wurzel
  - $I'(X) = I'(L_1 \otimes L_2)$  für



Dann gilt:  $I'(f(F)) = 1$

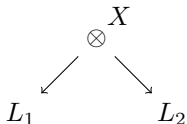
- „ $\Leftarrow$ “: Sei  $J$  Belegung von  $f(F)$  mit  $J(f(F)) = 1$   
Dann:  $I = J|_{Var(F)}$  macht  $F$  wahr.



## $\mathcal{NP}$ -Vollständigkeit von 3-CNF-SAT (5)

$F$  erfüllbar g.d.w.  $f(F)$  erfüllbar:

- „ $\Rightarrow$ “: Sei  $I$  Belegung mit  $I(F) = 1$ . Sei  $I'$  Belegung mit
  - $I'(X) = I(X)$  für alle Variablen, die in  $F$  vorkommen
  - $I'(W) = 1$  für die Variable  $W$  an der Wurzel
  - $I'(X) = I'(L_1 \otimes L_2)$  für



Dann gilt:  $I'(f(F)) = 1$

- „ $\Leftarrow$ “: Sei  $J$  Belegung von  $f(F)$  mit  $J(f(F)) = 1$   
Dann:  $I = J|_{Var(F)}$  macht  $F$  wahr.

Damit:  $SAT \leq_p 3\text{-CNF-SAT}$ .

Da  $SAT$   $\mathcal{NP}$ -hart, ist somit auch 3-CNF-SAT  $\mathcal{NP}$ -hart.

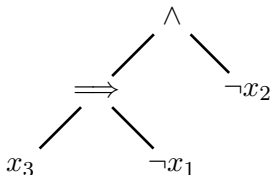
# Beispiel

$$F = \neg(\neg(x_3 \implies \neg x_1) \vee x_2)$$

Negationen nach innen schieben:

$$\begin{aligned} & \neg(\neg(x_3 \implies \neg x_1) \vee x_2) \\ \rightarrow & (\neg(\neg(x_3 \implies \neg x_1)) \wedge \neg x_2) \\ \rightarrow & (x_3 \implies \neg x_1) \wedge \neg x_2 \end{aligned}$$

Syntaxbaum dazu:



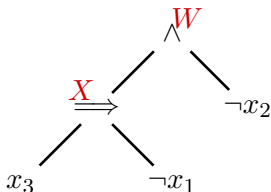
# Beispiel

$$F = \neg(\neg(x_3 \implies \neg x_1) \vee x_2)$$

Negationen nach innen schieben:

$$\begin{aligned} & \neg(\neg(x_3 \implies \neg x_1) \vee x_2) \\ \rightarrow & (\neg(\neg(x_3 \implies \neg x_1)) \wedge \neg x_2) \\ \rightarrow & (x_3 \implies \neg x_1) \wedge \neg x_2 \end{aligned}$$

Syntaxbaum dazu:



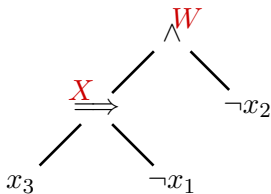
## Beispiel

$$F = \neg(\neg(x_3 \implies \neg x_1) \vee x_2)$$

Negationen nach innen schieben:

$$\begin{aligned} & \neg(\neg(x_3 \implies \neg x_1) \vee x_2) \\ \rightarrow & (\neg(\neg(x_3 \implies \neg x_1)) \wedge \neg x_2) \\ \rightarrow & (x_3 \implies \neg x_1) \wedge \neg x_2 \end{aligned}$$

Syntaxbaum dazu:



Formel:  $W \wedge (W \iff (X \wedge \neg x_2)) \wedge (X \iff (x_3 \implies \neg x_1))$

## Beispiel (2)

---

Berechnung der CNFs der kleinen Subformeln:

$$W \wedge (W \iff (X \wedge \neg x_2)) \wedge (X \iff (x_3 \implies \neg x_1))$$

## Beispiel (2)

---

Berechnung der CNFs der kleinen Subformeln:

$$\begin{aligned} & W \wedge (W \iff (X \wedge \neg x_2)) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee \neg(X \wedge \neg x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \end{aligned}$$

## Beispiel (2)

---

Berechnung der CNFs der kleinen Subformeln:

$$\begin{aligned} & W \wedge (W \iff (X \wedge \neg x_2)) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee \neg(X \wedge \neg x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee (\neg X \vee x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \end{aligned}$$

## Beispiel (2)

Berechnung der CNFs der kleinen Subformeln:

$$\begin{aligned} & W \wedge (W \iff (X \wedge \neg x_2)) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee \neg(X \wedge \neg x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee (\neg X \vee x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \wedge (X \iff (x_3 \implies \neg x_1)) \end{aligned}$$



## Beispiel (2)

Berechnung der CNFs der kleinen Subformeln:

$$\begin{aligned} & W \wedge (W \iff (X \wedge \neg x_2)) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee \neg(X \wedge \neg x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee (\neg X \vee x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee (x_3 \implies \neg x_1)) \wedge (X \vee \neg(x_3 \implies \neg x_1)) \end{aligned}$$

## Beispiel (2)

Berechnung der CNFs der kleinen Subformeln:

$$\begin{aligned} & W \wedge (W \iff (X \wedge \neg x_2)) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee \neg(X \wedge \neg x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee (\neg X \vee x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee (x_3 \implies \neg x_1)) \wedge (X \vee \neg(x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee \neg x_3 \vee \neg x_1) \wedge (X \vee \neg(\neg x_3 \vee \neg x_1)) \end{aligned}$$

## Beispiel (2)

Berechnung der CNFs der kleinen Subformeln:

$$\begin{aligned} & W \wedge (W \iff (X \wedge \neg x_2)) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee \neg(X \wedge \neg x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee (\neg X \vee x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee (x_3 \implies \neg x_1)) \wedge (X \vee \neg(x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee \neg x_3 \vee \neg x_1) \wedge (X \vee \neg(\neg x_3 \vee \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee \neg x_3 \vee \neg x_1) \wedge (X \vee (x_3 \wedge x_1)) \end{aligned}$$

## Beispiel (2)

Berechnung der CNFs der kleinen Subformeln:

$$\begin{aligned} & W \wedge (W \iff (X \wedge \neg x_2)) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee \neg(X \wedge \neg x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee (\neg X \vee x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee (x_3 \implies \neg x_1)) \wedge (X \vee \neg(x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee \neg x_3 \vee \neg x_1) \wedge (X \vee \neg(\neg x_3 \vee \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee \neg x_3 \vee \neg x_1) \wedge (X \vee (x_3 \wedge x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee \neg x_3 \vee \neg x_1) \wedge (X \vee x_3) \wedge (X \vee x_1) \end{aligned}$$

## Beispiel (2)

Berechnung der CNFs der kleinen Subformeln:

$$\begin{aligned} & W \wedge (W \iff (X \wedge \neg x_2)) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee \neg(X \wedge \neg x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee (\neg X \vee x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee (x_3 \implies \neg x_1)) \wedge (X \vee \neg(x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee \neg x_3 \vee \neg x_1) \wedge (X \vee \neg(\neg x_3 \vee \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee \neg x_3 \vee \neg x_1) \wedge (X \vee (x_3 \wedge x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee \neg x_3 \vee \neg x_1) \wedge (X \vee x_3) \wedge (X \vee x_1) \end{aligned}$$

Erfüllende Belegung  $J$ :

$$J(W) = 1, J(X) = 1, J(x_1) = 0, J(x_2) = 0, J(x_3) = 1$$

## Beispiel (2)

Berechnung der CNFs der kleinen Subformeln:

$$\begin{aligned} & W \wedge (W \iff (X \wedge \neg x_2)) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee \neg(X \wedge \neg x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge ((W \vee (\neg X \vee x_2)) \wedge (\neg W \vee (X \wedge \neg x_2))) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \wedge (X \iff (x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee (x_3 \implies \neg x_1)) \wedge (X \vee \neg(x_3 \implies \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee \neg x_3 \vee \neg x_1) \wedge (X \vee \neg(\neg x_3 \vee \neg x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee \neg x_3 \vee \neg x_1) \wedge (X \vee (x_3 \wedge x_1)) \\ \rightarrow & W \wedge (W \vee \neg X) \wedge (W \vee x_2) \wedge (\neg W \vee X) \wedge (\neg W \vee \neg x_2) \\ & \wedge (\neg X \vee \neg x_3 \vee \neg x_1) \wedge (X \vee x_3) \wedge (X \vee x_1) \end{aligned}$$

Erfüllende Belegung  $J$ :

$$J(W) = 1, J(X) = 1, J(x_1) = 0, J(x_2) = 0, J(x_3) = 1$$

Belegung  $I$ :

$$I(x_1) = 0, I(x_2) = 0, I(x_3) = 1 \text{ erfüllt } F = \neg(\neg(x_3 \implies \neg x_1) \vee x_2)$$

## Definition

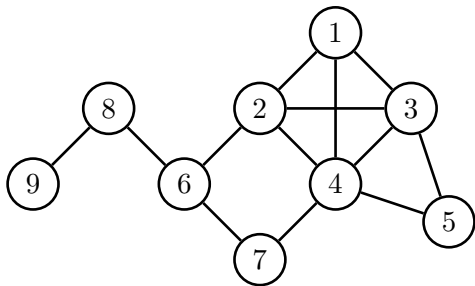
Für einen ungerichteten Graph  $G = (V, E)$  ist eine **Clique der Größe  $k$**  eine Menge  $V' \subseteq V$ , sodass  $|V'| = k$  und für alle  $u, v \in V'$  mit  $u \neq v$  gilt:  $\{u, v\} \in E$ .

# Cliquen in Graphen

## Definition

Für einen ungerichteten Graph  $G = (V, E)$  ist eine **Clique der Größe  $k$**  eine Menge  $V' \subseteq V$ , sodass  $|V'| = k$  und für alle  $u, v \in V'$  mit  $u \neq v$  gilt:  $\{u, v\} \in E$ .

Beispiel:



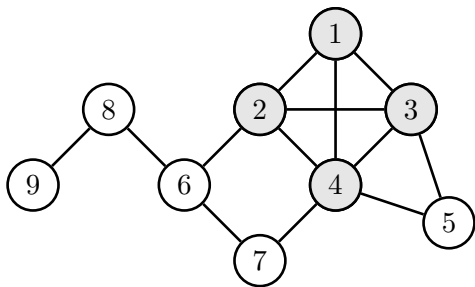


# Cliquen in Graphen

## Definition

Für einen ungerichteten Graph  $G = (V, E)$  ist eine **Clique der Größe  $k$**  eine Menge  $V' \subseteq V$ , sodass  $|V'| = k$  und für alle  $u, v \in V'$  mit  $u \neq v$  gilt:  $\{u, v\} \in E$ .

Beispiel:



Clique der Größe 4

## Definition (CLIQUE-Problem)

Das **CLIQUE-Problem** lässt sich in der gegeben/gefragt-Notation formulieren durch:

gegeben: Ein ungerichteter Graph  $G = (V, E)$  und eine Zahl  $k \in \mathbb{N}$ .

gefragt: Besitzt  $G$  eine Clique der Größe mindestens  $k$ ?

# $\mathcal{NP}$ -Vollständigkeit von CLIQUE (1)

---

## Satz

CLIQUE ist  $\mathcal{NP}$ -vollständig.

# $\mathcal{NP}$ -Vollständigkeit von CLIQUE (1)

## Satz

CLIQUE ist  $\mathcal{NP}$ -vollständig.

Beweis: CLIQUE  $\in \mathcal{NP}$

- Rate nichtdeterministisch eine Menge  $V' \subseteq V$  von  $k$  Knoten
- Prüfe (deterministisch), ob für alle  $u, v \in V'$  :  
 $u \neq v \implies \{u, v\} \in E$  gilt. Falls ja, akzeptiere.
- Daher kann eine NTM konstruiert werden, die CLIQUE in Polynomialzeit entscheidet.

## $\mathcal{NP}$ -Vollständigkeit von CLIQUE (2)

---

Beweis: CLIQUE ist  $\mathcal{NP}$ -hart:

- Ziel:  $3\text{-CNF-SAT} \leq_p \text{CLIQUE}$

## $\mathcal{NP}$ -Vollständigkeit von CLIQUE (2)

---

Beweis: CLIQUE ist  $\mathcal{NP}$ -hart:

- Ziel:  $3\text{-CNF-SAT} \leq_p \text{CLIQUE}$
- Sei  $F = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, wobei  $K_i = (L_{i,1} \vee L_{i,2} \vee L_{i,3})$  für  $i = 1, \dots, m$  (falls  $K_i < 3$  Literale hat, verdopple Literale)

## $\mathcal{NP}$ -Vollständigkeit von CLIQUE (2)

Beweis: CLIQUE ist  $\mathcal{NP}$ -hart:

- Ziel:  $3\text{-CNF-SAT} \leq_p \text{CLIQUE}$
- Sei  $F = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, wobei  $K_i = (L_{i,1} \vee L_{i,2} \vee L_{i,3})$  für  $i = 1, \dots, m$  (falls  $K_i < 3$  Literale hat, verdopple Literale)
- Für jedes  $L_{i,j}$  erzeuge: Knoten  $(i, j)$  im Graphen, d.h.

$$V = \bigcup_{i=1}^m \{(i, 1), (i, 2), (i, 3)\}$$

## $\mathcal{NP}$ -Vollständigkeit von CLIQUE (2)

Beweis: CLIQUE ist  $\mathcal{NP}$ -hart:

- Ziel:  $3\text{-CNF-SAT} \leq_p \text{CLIQUE}$
- Sei  $F = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, wobei  $K_i = (L_{i,1} \vee L_{i,2} \vee L_{i,3})$  für  $i = 1, \dots, m$  (falls  $K_i < 3$  Literale hat, verdopple Literale)
- Für jedes  $L_{i,j}$  erzeuge: Knoten  $(i, j)$  im Graphen, d.h.

$$V = \bigcup_{i=1}^m \{(i, 1), (i, 2), (i, 3)\}$$

- **Keine** Kante innerhalb der drei Knoten  $\{(i, 1), (i, 2), (i, 3)\}$



## $\mathcal{NP}$ -Vollständigkeit von CLIQUE (2)

Beweis: CLIQUE ist  $\mathcal{NP}$ -hart:

- Ziel:  $3\text{-CNF-SAT} \leq_p \text{CLIQUE}$
- Sei  $F = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, wobei  $K_i = (L_{i,1} \vee L_{i,2} \vee L_{i,3})$  für  $i = 1, \dots, m$  (falls  $K_i < 3$  Literale hat, verdopple Literale)
- Für jedes  $L_{i,j}$  erzeuge: Knoten  $(i, j)$  im Graphen, d.h.

$$V = \bigcup_{i=1}^m \{(i, 1), (i, 2), (i, 3)\}$$

- **Keine** Kante innerhalb der drei Knoten  $\{(i, 1), (i, 2), (i, 3)\}$
- Kanten zwischen „verschiedenen Klauseln“:  
 $E \subseteq \{(i, j), (i', j') \mid i \neq i' \wedge i, i' \in \{1, \dots, m\} \wedge j, j' \in \{1, 2, 3\}\}$

## $\mathcal{NP}$ -Vollständigkeit von CLIQUE (2)

Beweis: CLIQUE ist  $\mathcal{NP}$ -hart:

- Ziel:  $3\text{-CNF-SAT} \leq_p \text{CLIQUE}$
- Sei  $F = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, wobei  $K_i = (L_{i,1} \vee L_{i,2} \vee L_{i,3})$  für  $i = 1, \dots, m$  (falls  $K_i < 3$  Literale hat, verdopple Literale)
- Für jedes  $L_{i,j}$  erzeuge: Knoten  $(i, j)$  im Graphen, d.h.

$$V = \bigcup_{i=1}^m \{(i, 1), (i, 2), (i, 3)\}$$

- **Keine** Kante innerhalb der drei Knoten  $\{(i, 1), (i, 2), (i, 3)\}$
- Kanten zwischen „verschiedenen Klauseln“:  
 $E \subseteq \{ \{(i, j), (i', j')\} \mid i \neq i' \wedge i, i' \in \{1, \dots, m\} \wedge j, j' \in \{1, 2, 3\} \}$
- $E$  als maximale Menge, sodass sich **niemals zwei verbundene Literale  $L_{i,j}$  und  $L_{i',j'}$  widersprechen** (d.h.  $L_{i,j} \neq \overline{L_{i',j'}}$  mit  $\overline{L}$  negiertes Literal zu  $L$ :  $\overline{\overline{x}} = x$  und  $\overline{\neg x} = x$ ).

## $\mathcal{NP}$ -Vollständigkeit von CLIQUE (2)

Beweis: CLIQUE ist  $\mathcal{NP}$ -hart:

- Ziel:  $3\text{-CNF-SAT} \leq_p \text{CLIQUE}$
- Sei  $F = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, wobei  $K_i = (L_{i,1} \vee L_{i,2} \vee L_{i,3})$  für  $i = 1, \dots, m$  (falls  $K_i < 3$  Literale hat, verdopple Literale)
- Für jedes  $L_{i,j}$  erzeuge: Knoten  $(i, j)$  im Graphen, d.h.

$$V = \bigcup_{i=1}^m \{(i, 1), (i, 2), (i, 3)\}$$

- **Keine** Kante innerhalb der drei Knoten  $\{(i, 1), (i, 2), (i, 3)\}$
- Kanten zwischen „verschiedenen Klauseln“:  
 $E \subseteq \{ \{(i, j), (i', j')\} \mid i \neq i' \wedge i, i' \in \{1, \dots, m\} \wedge j, j' \in \{1, 2, 3\} \}$
- $E$  als maximale Menge, sodass sich **niemals zwei verbundene Literale  $L_{i,j}$  und  $L_{i',j'}$  widersprechen** (d.h.  $L_{i,j} \neq \overline{L_{i',j'}}$  mit  $\overline{L}$  negiertes Literal zu  $L$ :  $\overline{\overline{x}} = x$  und  $\overline{x} = \neg x$ ).

$$E := \{ \{(i, j), (i', j')\} \mid i \neq i' \wedge i, i' \in \{1, \dots, m\} \wedge j, j' \in \{1, 2, 3\}, L_{i,j} \neq \overline{L_{i',j'}} \}$$

## $\mathcal{NP}$ -Vollständigkeit von CLIQUE (2)

Beweis: CLIQUE ist  $\mathcal{NP}$ -hart:

- Ziel:  $3\text{-CNF-SAT} \leq_p \text{CLIQUE}$
- Sei  $F = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, wobei  $K_i = (L_{i,1} \vee L_{i,2} \vee L_{i,3})$  für  $i = 1, \dots, m$  (falls  $K_i < 3$  Literale hat, verdopple Literale)
- Für jedes  $L_{i,j}$  erzeuge: Knoten  $(i, j)$  im Graphen, d.h.

$$V = \bigcup_{i=1}^m \{(i, 1), (i, 2), (i, 3)\}$$

- **Keine** Kante innerhalb der drei Knoten  $\{(i, 1), (i, 2), (i, 3)\}$
- Kanten zwischen „verschiedenen Klauseln“:  
 $E \subseteq \{\{(i, j), (i', j')\} \mid i \neq i' \wedge i, i' \in \{1, \dots, m\} \wedge j, j' \in \{1, 2, 3\}\}$
- $E$  als maximale Menge, sodass sich **niemals zwei verbundene Literale  $L_{i,j}$  und  $L_{i',j'}$  widersprechen** (d.h.  $L_{i,j} \neq \overline{L_{i',j'}}$  mit  $\overline{L}$  negiertes Literal zu  $L$ :  $\overline{\overline{x}} = x$  und  $\overline{x} = \neg x$ ).

$$E := \{\{(i, j), (i', j')\} \mid i \neq i' \wedge i, i' \in \{1, \dots, m\} \wedge j, j' \in \{1, 2, 3\}, L_{i,j} \neq \overline{L_{i',j'}}\}$$

- $f(F) = ((V, E), m)$  ist in Polynomialzeit berechenbar.

## $\mathcal{NP}$ -Vollständigkeit von CLIQUE (3)

---

Zu Zeigen:  $F$  erfüllbar, g.d.w.  $(V, E)$  eine Clique der Größe mindestens  $m$  hat.

## $\mathcal{NP}$ -Vollständigkeit von CLIQUE (3)

---

Zu Zeigen:  $F$  erfüllbar, g.d.w.  $(V, E)$  eine Clique der Größe mindestens  $m$  hat.

„ $\Rightarrow$ “

- Wenn  $F$  erfüllbar ist:
  - $\exists I$ , sodass in jeder Klausel  $I(K_i)$  mind 1 Literal wahr.
- D.h.:  $L_{1,j_1}, \dots, L_{m,j_m}$  mit  $I(L_{1,j_1}) = 1, \dots, I(L_{m,j_m}) = 1$ .
- Diese können sich paarweise nicht widersprechen  
 $\implies$  sind im Graphen paarweise miteinander verbunden
- Sie formen eine Clique der Größe  $m$ .

## $\mathcal{NP}$ -Vollständigkeit von CLIQUE (4)

Zu Zeigen:  $F$  erfüllbar, g.d.w.  $(V, E)$  eine Clique der Größe mindestens  $m$  hat.

„ $\Leftarrow$ “

- $(V, E)$  hat Clique der Größe mindestens  $m$ .
- Dann gibt es Clique  $V' = \{(i_1, j_1), \dots, (i_m, j_m)\}$ ,
- Da  $(i, x)$  und  $(i, y)$  nie miteinander verbunden sind, müssen alle  $i_1, \dots, i_m$  paarweise verschieden sein, also  $\{i_1, \dots, i_m\} = \{1, \dots, m\}$ .
- Daher: Die Literale  $L_{i_1, j_1}, \dots, L_{i_m, j_m}$  widersprechen sich paarweise nicht
- Daher Belegung  $I$  mit  $I(x) = 1$  wenn  $L_{i_k, j_k} = x$  und  $I(x) = 0$  wenn  $L_{i, k, j_k} = \neg x$  und  $I(y) = 1$  für alle anderen Variablen
- $I$  macht  $F$  wahr

## Beispiel

---

Sei  $F = (x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_1)$

(1,1)

(1,2)

(1,3)

(2,1)

(2,2)

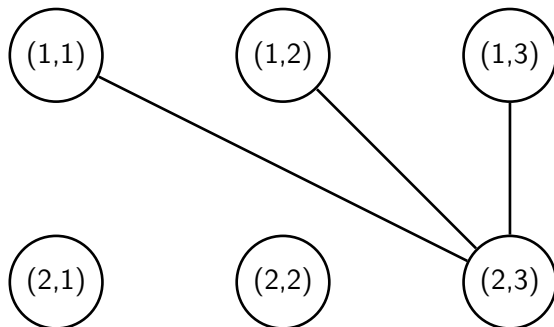
(2,3)

Es gibt keine Kanten, da sich  $(1, i)$  und  $(2, j)$  stets widersprechen.



## Beispiel (2)

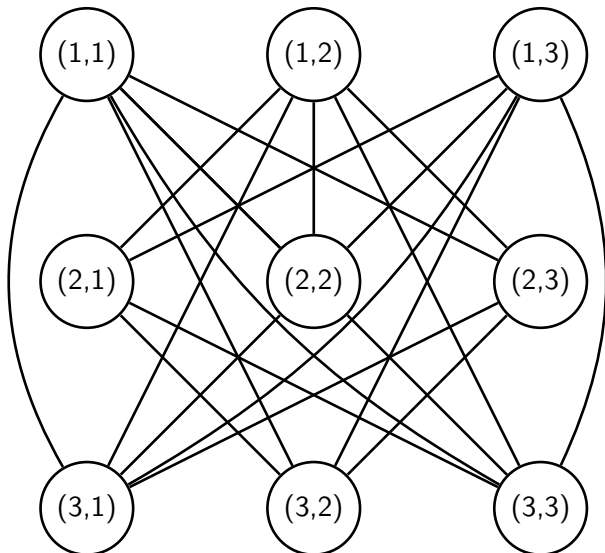
Sei  $F = (x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee x_1)$



Jede der Cliques der Größe 2 führt zu erfüllender Belegung, die  $x_1$  wahr macht.

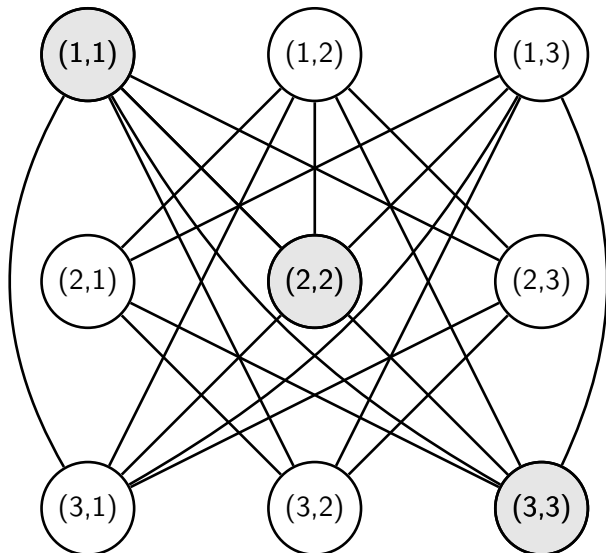
## Beispiel (2)

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



## Beispiel (2)

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



# Unabhängige Knotenmenge

## Definition

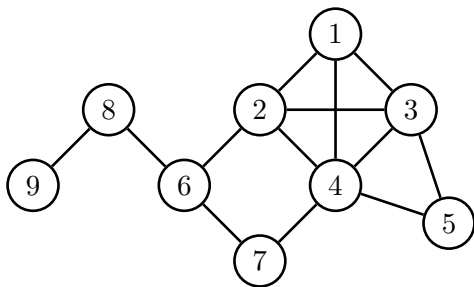
Für einen Graphen  $G = (V, E)$  ist  $V' \subseteq V$  eine **unabhängige Knotenmenge** (independent set), wenn keine zwei Knoten aus  $V'$  über eine Kante verbunden sind, d.h.  $u, v \in V' \implies \{u, v\} \notin E$ .

# Unabhängige Knotenmenge

## Definition

Für einen Graphen  $G = (V, E)$  ist  $V' \subseteq V$  eine **unabhängige Knotenmenge** (independent set), wenn keine zwei Knoten aus  $V'$  über eine Kante verbunden sind, d.h.  $u, v \in V' \implies \{u, v\} \notin E$ .

Beispiel:

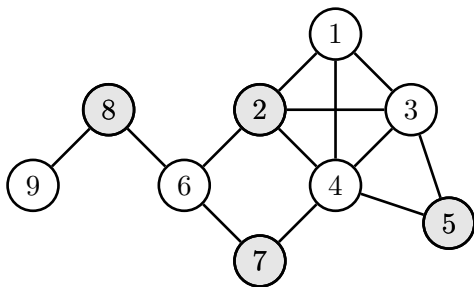


# Unabhängige Knotenmenge

## Definition

Für einen Graphen  $G = (V, E)$  ist  $V' \subseteq V$  eine **unabhängige Knotenmenge** (independent set), wenn keine zwei Knoten aus  $V'$  über eine Kante verbunden sind, d.h.  $u, v \in V' \implies \{u, v\} \notin E$ .

Beispiel:

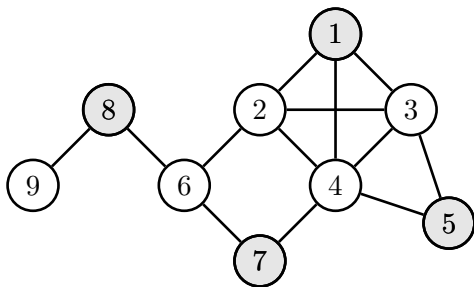


# Unabhängige Knotenmenge

## Definition

Für einen Graphen  $G = (V, E)$  ist  $V' \subseteq V$  eine **unabhängige Knotenmenge** (independent set), wenn keine zwei Knoten aus  $V'$  über eine Kante verbunden sind, d.h.  $u, v \in V' \implies \{u, v\} \notin E$ .

Beispiel:

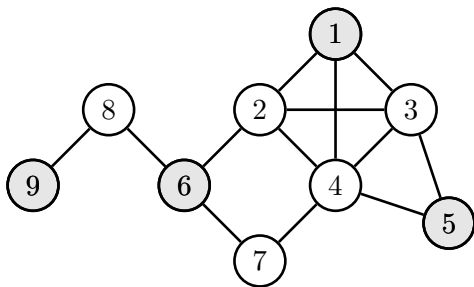


# Unabhängige Knotenmenge

## Definition

Für einen Graphen  $G = (V, E)$  ist  $V' \subseteq V$  eine **unabhängige Knotenmenge** (independent set), wenn keine zwei Knoten aus  $V'$  über eine Kante verbunden sind, d.h.  $u, v \in V' \implies \{u, v\} \notin E$ .

Beispiel:



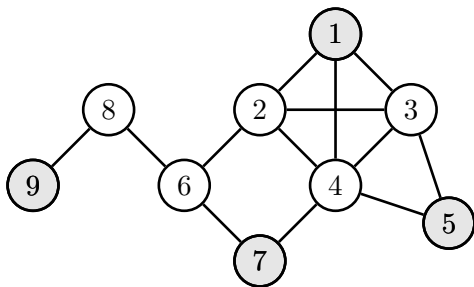


# Unabhängige Knotenmenge

## Definition

Für einen Graphen  $G = (V, E)$  ist  $V' \subseteq V$  eine **unabhängige Knotenmenge** (independent set), wenn keine zwei Knoten aus  $V'$  über eine Kante verbunden sind, d.h.  $u, v \in V' \implies \{u, v\} \notin E$ .

Beispiel:

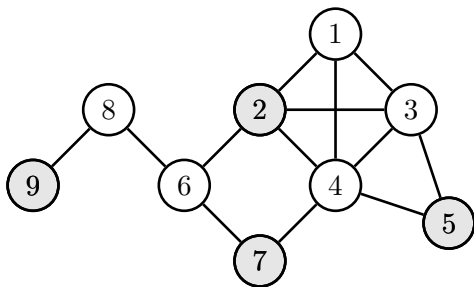


# Unabhängige Knotenmenge

## Definition

Für einen Graphen  $G = (V, E)$  ist  $V' \subseteq V$  eine **unabhängige Knotenmenge** (independent set), wenn keine zwei Knoten aus  $V'$  über eine Kante verbunden sind, d.h.  $u, v \in V' \implies \{u, v\} \notin E$ .

Beispiel:



# Komplementgraph

---

Für Graph  $G = (V, E)$  ist der Komplementgraph zu  $G$ , der Graph  $\overline{G} = (V, \overline{E})$  mit  $\overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$

Für Graph  $G = (V, E)$  ist der Komplementgraph zu  $G$ , der Graph  $\overline{G} = (V, \overline{E})$  mit  $\overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$

## Lemma

Für jeden ungerichteten Graph  $G$  gilt:  $G$  hat eine unabhängige Knotenmenge der Größe  $k$  genau dann, wenn  $\overline{G}$  eine Clique der Größe  $k$  hat.

Für Graph  $G = (V, E)$  ist der Komplementgraph zu  $G$ , der Graph  $\overline{G} = (V, \overline{E})$  mit  $\overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$

## Lemma

Für jeden ungerichteten Graph  $G$  gilt:  $G$  hat eine unabhängige Knotenmenge der Größe  $k$  genau dann, wenn  $\overline{G}$  eine Clique der Größe  $k$  hat.

Beweis:

$V'$  ist unabhängige Knotenmenge der Größe  $k$

Für Graph  $G = (V, E)$  ist der Komplementgraph zu  $G$ , der Graph  $\overline{G} = (V, \overline{E})$  mit  $\overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$

## Lemma

Für jeden ungerichteten Graph  $G$  gilt:  $G$  hat eine unabhängige Knotenmenge der Größe  $k$  genau dann, wenn  $\overline{G}$  eine Clique der Größe  $k$  hat.

Beweis:

$V'$  ist unabhängige Knotenmenge der Größe  $k$

g.d.w.  $V' \subseteq V$  mit  $u, v \in V' \implies \{u, v\} \notin E$  und  $|V'| = k$

Für Graph  $G = (V, E)$  ist der Komplementgraph zu  $G$ , der Graph  $\overline{G} = (V, \overline{E})$  mit  $\overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$

## Lemma

Für jeden ungerichteten Graph  $G$  gilt:  $G$  hat eine unabhängige Knotenmenge der Größe  $k$  genau dann, wenn  $\overline{G}$  eine Clique der Größe  $k$  hat.

Beweis:

$V'$  ist unabhängige Knotenmenge der Größe  $k$

g.d.w.  $V' \subseteq V$  mit  $u, v \in V' \implies \{u, v\} \notin E$  und  $|V'| = k$

g.d.w.  $u, v \in V' \implies \{u, v\} \in \overline{E}$

Für Graph  $G = (V, E)$  ist der Komplementgraph zu  $G$ , der Graph  $\overline{G} = (V, \overline{E})$  mit  $\overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$

## Lemma

Für jeden ungerichteten Graph  $G$  gilt:  $G$  hat eine unabhängige Knotenmenge der Größe  $k$  genau dann, wenn  $\overline{G}$  eine Clique der Größe  $k$  hat.

Beweis:

$V'$  ist unabhängige Knotenmenge der Größe  $k$

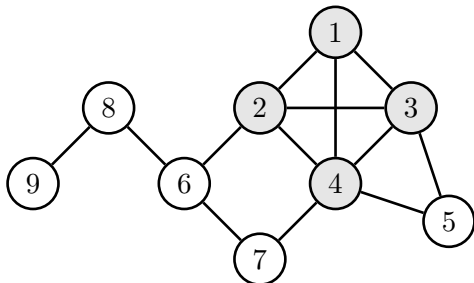
g.d.w.  $V' \subseteq V$  mit  $u, v \in V' \implies \{u, v\} \notin E$  und  $|V'| = k$

g.d.w.  $u, v \in V' \implies \{u, v\} \in \overline{E}$

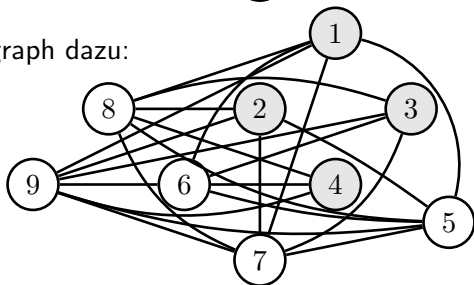
g.d.w.  $V'$  ist eine Clique der Größe  $k$  in  $\overline{G}$



# Beispiel



Komplementgraph dazu:



## Definition (INDEPENDENT-SET-Problem)

Das **INDEPENDENT-SET-Problem** lässt sich in der gegeben/gefragt-Notation formulieren durch:

gegeben: Ein ungerichteter Graph  $G = (V, E)$  und eine Zahl  $k \in \mathbb{N}$ .

gefragt: Besitzt  $G$  eine unabhängige Knotenmenge der Größe mindestens  $k$ ?

## Satz

INDEPENDENT-SET ist  $\mathcal{NP}$ -vollständig.

Beweis, Teil 1: INDEPENDENT-SET  $\in \mathcal{NP}$ :

- Rate nichtdeterministisch Menge  $V'$  von  $k$  Knoten.
- Verifiziere deterministisch, ob für jedes Paar  $\{u, v\} \in V'$  gilt:  $\{u, v\} \notin E$ . Dies geht in Polynomialzeit.
- Daher kann INDEPENDENT-SET in Polynomialzeit auf einer NTM entschieden werden.

## $\mathcal{NP}$ -Vollständigkeit von INDEPENDENT-SET (2)

INDEPENDENT-SET ist  $\mathcal{NP}$ -hart.

- Sei  $f((V, E, m)) = (V, \bar{E}, m)$  mit  $\bar{E} = \{\{u, v\} \mid u, v \in V, \{u, v\} \notin E\}$ .
- Dann gilt:
  - $(V, E)$  hat eine CLIQUE der Größe mindestens  $k$  g.d.w.
  - $(V, \bar{E})$  hat ein INDEPENDENT-SET der Größe mindestens  $k$ .
- Funktion  $f$  kann in Polynomialzeit berechnet werden.
- Daher:  $\text{CLIQUE} \leq_p \text{INDEPENDENT-SET}$
- Da CLIQUE  $\mathcal{NP}$ -hart, folgt  
INDEPENDENT-SET ist  $\mathcal{NP}$ -hart.

## Definition

Für einen Graph  $G = (V, E)$  ist  $V' \subseteq V$  eine **überdeckende Knotenmenge** (vertex cover), wenn jede Kante aus  $E$  mindestens 1 Knoten in  $V'$  hat, d.h. für alle Knoten  $u, v \in V$  :

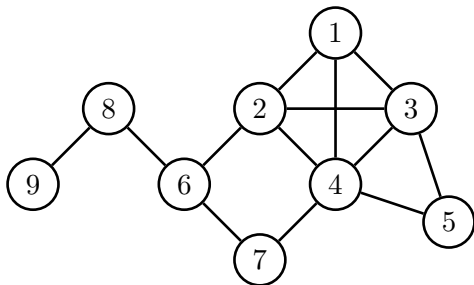
$$\{u, v\} \in E \implies u \in V' \vee v \in V'.$$

Beachte:

- $V$  ist immer eine überdeckende Knotenmenge
- Man möchte ein möglichst kleine Menge  $V'$  finden.

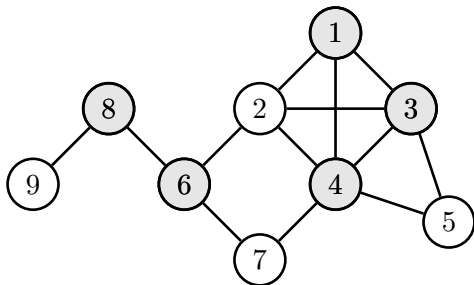
# Beispiel

---



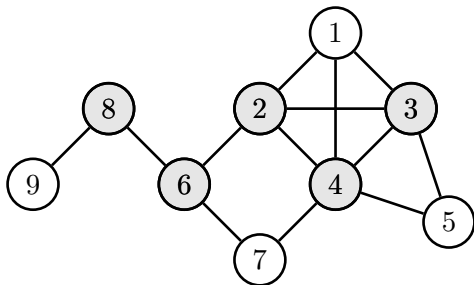
# Beispiel

---



# Beispiel

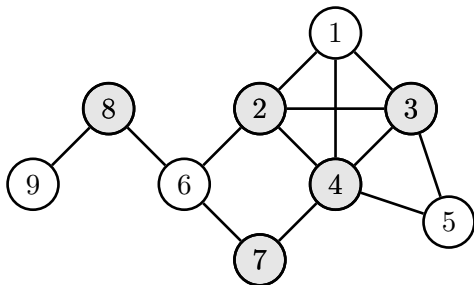
---





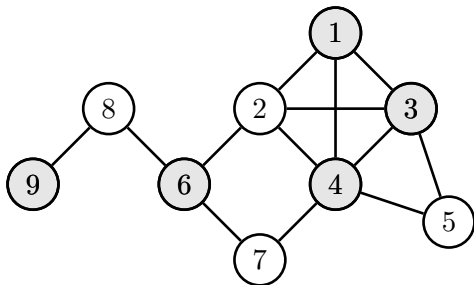
# Beispiel

---



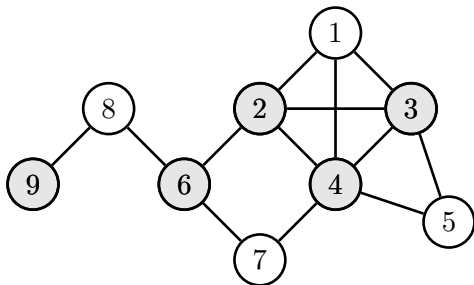
# Beispiel

---



# Beispiel

---



# Überdeckende Knotenmengen vs. unabhängige Knotenmengen

## Lemma

$G = (V, E)$  hat eine unabhängige Knotenmenge der Größe  $k$ ,  
g.d.w.  $G$  hat eine überdeckende Knotenmenge der Größe  $|V| - k$ .

Beweis:

$V' \subseteq V$  ist unabhängige Knotenmenge

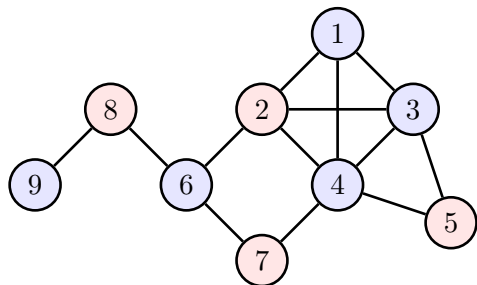
g.d.w.  $u, v \in V' \implies \{u, v\} \notin E$

g.d.w.  $\{u, v\} \in E \implies u \notin V' \vee v \notin V'$

g.d.w.  $\{u, v\} \in E \implies (u \in V \setminus V') \vee (v \in V \setminus V')$

g.d.w.  $V \setminus V'$  ist überdeckende Knotenmenge

# Beispiel



 unabhängige Knotenmenge

 überdeckende Knotenmenge

## Definition (VERTEX-COVER-Problem)

Das **VERTEX-COVER-Problem** lässt sich in der gegeben/gefragt-Notation formulieren durch:

gegeben: Ein ungerichteter Graph  $G = (V, E)$  und eine Zahl  $k \in \mathbb{N}$ .

gefragt: Besitzt  $G$  eine überdeckende Knotenmenge der Größe höchstens  $k$ ?

# $\mathcal{NP}$ -Vollständigkeit von VERTEX-COVER

---

## Satz

VERTEX-COVER- ist  $\mathcal{NP}$ -vollständig.

# $\mathcal{NP}$ -Vollständigkeit von VERTEX-COVER

---

## Satz

VERTEX-COVER- ist  $\mathcal{NP}$ -vollständig.



# $\mathcal{NP}$ -Vollständigkeit von VERTEX-COVER

## Satz

VERTEX-COVER- ist  $\mathcal{NP}$ -vollständig.

Beweis: Sei  $G = (V, E)$ . VERTEX-COVER  $\in \mathcal{NP}$ :

- Rate nichtdeterministisch eine Menge  $V' \subseteq V$  von  $k$  Knoten
- Prüfe deterministisch (in Polynomialzeit), ob für alle  $\{u, v\} \in E$ :  
 $u \in V' \vee v \in V'$  gilt.
- D.h. VERTEX-COVER wird in Polynomialzeit von NTM entschieden.

# $\mathcal{NP}$ -Vollständigkeit von VERTEX-COVER

## Satz

VERTEX-COVER- ist  $\mathcal{NP}$ -vollständig.

Beweis: Sei  $G = (V, E)$ . VERTEX-COVER  $\in \mathcal{NP}$ :

- Rate nichtdeterministisch eine Menge  $V' \subseteq V$  von  $k$  Knoten
- Prüfe deterministisch (in Polynomialzeit), ob für alle  $\{u, v\} \in E$ :  
 $u \in V' \vee v \in V'$  gilt.
- D.h. VERTEX-COVER wird in Polynomialzeit von NTM entschieden.

VERTEX-COVER ist  $\mathcal{NP}$ -hart:

- Sei  $f((V, E, m)) = (V, E, |V| - m)$ . Dann gilt:  
 $(V, E)$  hat independent set der Größe mindestens  $m$  g.d.w.  $(V, E)$   
hat vertex cover der Größe höchstens  $|V| - m$ .
- Da  $f$  in Polynomialzeit berechnet werden kann, gilt  
INDEPENDENT-SET  $\leq_p$  VERTEX-COVER.

## Definition (SETCOVER-Problem)

Das **SETCOVER-Problem** lässt sich in der gegeben/gefragt-Notation formulieren durch:

gegeben: Mengen  $T_1, \dots, T_k$  mit  $T_1, \dots, T_k \subseteq M$ , wobei  $M$  eine endliche Grundmenge ist und eine Zahl  $n \leq k$ .

gefragt: Gibt es eine Auswahl von  $n$  Mengen  $T_{i_1}, \dots, T_{i_n}$  ( $i_j \in \{1, \dots, k\}$ ) mit  $T_{i_1} \cup \dots \cup T_{i_n} = M$ ?

## Definition (SETCOVER-Problem)

Das **SETCOVER-Problem** lässt sich in der gegeben/gefragt-Notation formulieren durch:

gegeben: Mengen  $T_1, \dots, T_k$  mit  $T_1, \dots, T_k \subseteq M$ , wobei  $M$  eine endliche Grundmenge ist und eine Zahl  $n \leq k$ .

gefragt: Gibt es eine Auswahl von  $n$  Mengen  $T_{i_1}, \dots, T_{i_n}$  ( $i_j \in \{1, \dots, k\}$ ) mit  $T_{i_1} \cup \dots \cup T_{i_n} = M$ ?

Beispiel:

$T_1 = \{1, 2, 3, 5\}, T_2 = \{1, 2\}, T_3 = \{3, 4\}, T_4 = \{3\}$  mit  $M = \{1, 2, 3, 4, 5\}$  und  $n = 2$

## Definition (SETCOVER-Problem)

Das **SETCOVER-Problem** lässt sich in der gegeben/gefragt-Notation formulieren durch:

gegeben: Mengen  $T_1, \dots, T_k$  mit  $T_1, \dots, T_k \subseteq M$ , wobei  $M$  eine endliche Grundmenge ist und eine Zahl  $n \leq k$ .

gefragt: Gibt es eine Auswahl von  $n$  Mengen  $T_{i_1}, \dots, T_{i_n}$  ( $i_j \in \{1, \dots, k\}$ ) mit  $T_{i_1} \cup \dots \cup T_{i_n} = M$ ?

Beispiel:

$T_1 = \{1, 2, 3, 5\}, T_2 = \{1, 2\}, T_3 = \{3, 4\}, T_4 = \{3\}$  mit  $M = \{1, 2, 3, 4, 5\}$  und  $n = 2$

Lösung:  $T_1, T_3$ , da  $T_1 \cup T_3 = M$ .

## Satz

SETCOVER ist  $\mathcal{NP}$ -vollständig.

Beweis, Teil 1: SETCOVER  $\in \mathcal{NP}$

- Rate nichtdeterministisch die  $n$  Mengen  $T_{i_1}, \dots, T_{i_n}$ .
- Verifiziere deterministisch, ob  $T_{i_1} \cup \dots \cup T_{i_n} = M$  gilt.
- Daher kann SETCOVER in Polynomialzeit auf einer NTM entschieden werden.

## $\mathcal{NP}$ -Vollständigkeit von SETCOVER (2)

Beweis, Teil 2: SETCOVER ist  $\mathcal{NP}$ -hart.

- Ziel:  $3\text{-CNF-SAT} \leq_p \text{SETCOVER}$ .
- Sei  $F = K_1 \wedge \dots \wedge K_m$  eine 3-CNF.
- Seien  $x_1, \dots, x_n$  die aussagenlogischen Variablen.
- Setze  $M = \{1, \dots, m, m+1, \dots, m+n\}$ .
- Für  $i = 1, \dots, n$  sei
$$T_{i,a} = \{j \mid \text{Literal } x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+i\}$$
$$T_{i,b} = \{j \mid \text{Literal } \neg x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+i\}$$
- Das Mengensystem sei  $T_{1,a}, \dots, T_{n,a}, T_{1,b}, \dots, T_{n,b} \subseteq M$ .

## $\mathcal{NP}$ -Vollständigkeit von SETCOVER (3)

Zur Erinnerung:  $T_{i,a} = \{j \mid \text{Literal } x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+i\}$   
 $T_{i,b} = \{j \mid \text{Literal } \neg x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+i\}$

Sei  $I$  Belegung mit  $I(F) = 1$ .

- Wenn  $I(x_i) = 1$ , dann wähle  $T_{i,a}$ , sonst wähle  $T_{i,b}$
- Ergibt  $n$  gewählte Mengen
- Jede Zahl aus  $M$  kommt vor:
  - Da jede Klausel durch  $I$  erfüllt, kommen alle  $1, \dots, m$  vor
  - Da jede Variable  $x_i$  mit 0 oder 1 belegt wird, kommen alle  $m+1, \dots, m+n$  vor
- Vereinigung der  $n$  Mengen ergibt daher  $M$ .
- SETCOVER ist lösbar.



## $\mathcal{NP}$ -Vollständigkeit von SETCOVER (4)

Zur Erinnerung:  $T_{i,a} = \{j \mid \text{Literal } x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+i\}$   
 $T_{i,b} = \{j \mid \text{Literal } \neg x_i \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+i\}$

Seien  $U_1, \dots, U_n \subseteq T_{1,a}, \dots, T_{n,a}, T_{1,b}, \dots, T_{n,b}$  mit  
 $U_1 \cup \dots \cup U_n = M$

- Da  $m+1, \dots, m+n$  in der Vereinigung, muss für  $i = 1, \dots, n$  genau  $T_{i,a}$  oder  $T_{i,b}$  in der Vereinigung sein.
- O.B.d.A.  $U_i \in \{T_{i,a}, T_{i,b}\}$
- Setze  $I(x_i) = 1$  wenn  $U_i = T_{i,a}$  und  $I(x_i) = 0$  wenn  $U_i = T_{i,b}$
- Da  $1, \dots, m$  in der Vereinigung, wird in jeder Klausel ein Literal durch  $I$  wahr gemacht
- $F$  ist erfüllbar.

Da diese Übersetzung in Polynomialzeit berechenbar ist, gilt  
 $3\text{-SAT-CNF} \leq_p \text{SETCOVER}$ .

3-CNF:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$$

3-CNF:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$$

SET-COVER-Instanz dazu:

$$T_{1,a} = \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{1, 3, 5\}$$

$$T_{1,b} = \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{2, 4, 5\}$$

$$T_{2,a} = \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{1, 4, 6\}$$

$$T_{2,b} = \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{2, 3, 6\}$$

3-CNF:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$$

SET-COVER-Instanz dazu:

$$T_{1,a} = \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{1, 3, 5\}$$

$$T_{1,b} = \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{2, 4, 5\}$$

$$T_{2,a} = \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{1, 4, 6\}$$

$$T_{2,b} = \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{2, 3, 6\}$$

Gesucht: Vereinigung von  $n = 2$  Mengen  $T_{i,j}, T_{i',j'}$  mit

$$T_{i,j} \cup T_{i',j'} = \{1, 2, 3, 4, 5, 6\}$$

3-CNF:

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$$

SET-COVER-Instanz dazu:

$$T_{1,a} = \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{1, 3, 5\}$$

$$T_{1,b} = \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{2, 4, 5\}$$

$$T_{2,a} = \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{1, 4, 6\}$$

$$T_{2,b} = \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{2, 3, 6\}$$

Gesucht: Vereinigung von  $n = 2$  Mengen  $T_{i,j}, T_{i',j'}$  mit

$$T_{i,j} \cup T_{i',j'} = \{1, 2, 3, 4, 5, 6\}$$

Keine Lösung!

## Beispiel (2)

---

3-CNF:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

## Beispiel (2)

3-CNF:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

SET-COVER-Instanz dazu:

$$T_{1,a} = \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{1, 2, 4\}$$

$$T_{1,b} = \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{3, 4\}$$

$$T_{2,a} = \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{1, 5\}$$

$$T_{2,b} = \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{2, 3, 5\}$$

$$T_{3,a} = \{j \mid x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} = \{1, 3, 6\}$$

$$T_{3,b} = \{j \mid \neg x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} = \{6\}$$

## Beispiel (2)

3-CNF:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

SET-COVER-Instanz dazu:

$$T_{1,a} = \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{1, 2, 4\}$$

$$T_{1,b} = \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{3, 4\}$$

$$T_{2,a} = \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{1, 5\}$$

$$T_{2,b} = \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{2, 3, 5\}$$

$$T_{3,a} = \{j \mid x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} = \{1, 3, 6\}$$

$$T_{3,b} = \{j \mid \neg x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} = \{6\}$$

Gesucht: Vereinigung von  $n = 3$  Mengen  $T_{i,j}, T_{i',j'}$  mit

$$T_{i,j} \cup T_{i',j'} = \{1, 2, 3, 4, 5, 6\}$$



## Beispiel (2)

3-CNF:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

SET-COVER-Instanz dazu:

$$T_{1,a} = \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{1, 2, 4\}$$

$$T_{1,b} = \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{3, 4\}$$

$$T_{2,a} = \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{1, 5\}$$

$$T_{2,b} = \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{2, 3, 5\}$$

$$T_{3,a} = \{j \mid x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} = \{1, 3, 6\}$$

$$T_{3,b} = \{j \mid \neg x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} = \{6\}$$

Gesucht: Vereinigung von  $n = 3$  Mengen  $T_{i,j}, T_{i',j'}$  mit

$$T_{i,j} \cup T_{i',j'} = \{1, 2, 3, 4, 5, 6\}$$

Lösung z.B.  $T_{1,a}, T_{2,b}, T_{3,b}$

## Beispiel (2)

3-CNF:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

SET-COVER-Instanz dazu:

$$T_{1,a} = \{j \mid x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{1, 2, 4\}$$

$$T_{1,b} = \{j \mid \neg x_1 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+1\} = \{3, 4\}$$

$$T_{2,a} = \{j \mid x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{1, 5\}$$

$$T_{2,b} = \{j \mid \neg x_2 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+2\} = \{2, 3, 5\}$$

$$T_{3,a} = \{j \mid x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} = \{1, 3, 6\}$$

$$T_{3,b} = \{j \mid \neg x_3 \text{ kommt in Klausel } K_j \text{ vor}\} \cup \{m+3\} = \{6\}$$

Gesucht: Vereinigung von  $n = 3$  Mengen  $T_{i,j}, T_{i',j'}$  mit

$$T_{i,j} \cup T_{i',j'} = \{1, 2, 3, 4, 5, 6\}$$

Lösung z.B.  $T_{1,a}, T_{2,b}, T_{3,b}$

Belegung dazu  $I(x_1) = 1, I(x_2) = 0, I(x_3) = 0$

## Definition (SUBSETSUM-Problem)

Das **SUBSETSUM-Problem** lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Natürliche Zahlen  $a_1, \dots, a_k \in \mathbb{N}$  und  $s \in \mathbb{N}$

gefragt: Gibt es eine Teilmenge  $I \subseteq \{1, \dots, k\}$ ,  
sodass  $\sum_{i \in I} a_i = s$ ?

Beispiel:

$a_1, \dots, a_6 = 1, 21, 5, 16, 12, 19$  und  $s = 49$

Lösung: 2, 4, 5, da  $21 + 16 + 12 = 49$

## Satz

Das SUBSETSUM-Problem ist  $\mathcal{NP}$ -vollständig.

Beweis, Teil 1: SUBSETSUM  $\in \mathcal{NP}$ :

- Rate nichtdeterministisch eine Teilmenge  $I \subseteq \{1, \dots, k\}$
- Prüfe (deterministisch) ob  $\sum_{i \in I} a_i = s$  gilt
- Daher: SUBSETSUM in Polynomialzeit auf NTM entscheidbar.

## $\mathcal{NP}$ -Vollständigkeit von SUBSETSUM (2)

---

SUBSETSUM ist  $\mathcal{NP}$ -hart: Zeige  $3\text{-CNF-SAT} \leq_p \text{SUBSETSUM}$

- Sei  $F = K_1 \wedge \dots \wedge K_m$  eine 3-CNF
- Annahme: Alle Klauseln  $K_j$  bestehen aus genau 3 Literalen (Vervielfache Literale anderenfalls)
- Seien  $\text{Var}(F) = \{x_1, \dots, x_n\}$  die Variablen in  $F$

## $\mathcal{NP}$ -Vollständigkeit von SUBSETSUM (2)

---

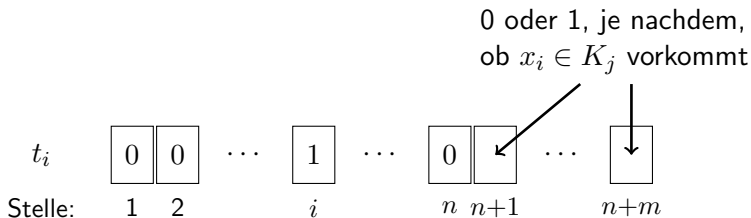
Konstruktion der SUBSETSUM-Instanz:

Erzeuge  $n + m$ -stellige Zahlen  $t_i, f_i$  für  $i = 1, \dots, n$ :

# $\mathcal{NP}$ -Vollständigkeit von SUBSETSUM (2)

## Konstruktion der SUBSETSUM-Instanz:

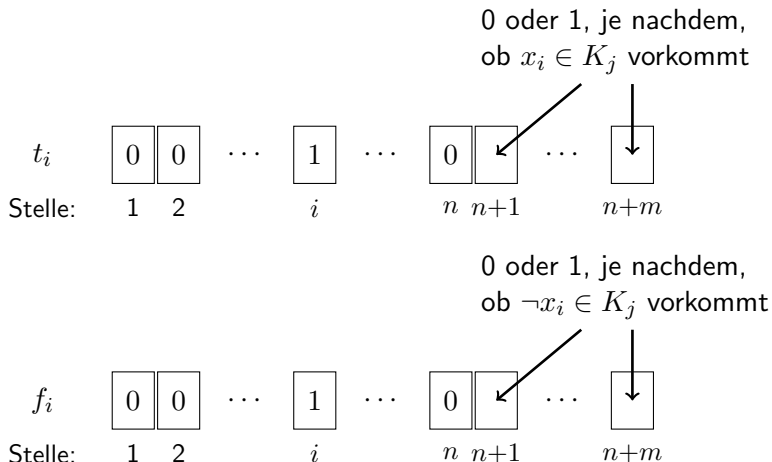
Erzeuge  $n + m$ -stellige Zahlen  $t_i, f_i$  für  $i = 1, \dots, n$ :



# $\mathcal{NP}$ -Vollständigkeit von SUBSETSUM (2)

## Konstruktion der SUBSETSUM-Instanz:

Erzeuge  $n + m$ -stellige Zahlen  $t_i, f_i$  für  $i = 1, \dots, n$ :





## $\mathcal{NP}$ -Vollständigkeit von SUBSETSUM (3)

---

Weitere Zahlen:

Erzeuge  $n + m$ -stellige Zahlen  $c_j, d_j$  für  $j = 1, \dots, m$ :

# $\mathcal{NP}$ -Vollständigkeit von SUBSETSUM (3)

Weitere Zahlen:

Erzeuge  $n + m$ -stellige Zahlen  $c_j, d_j$  für  $j = 1, \dots, m$ :

$$\begin{array}{ccccccc} c_j & \boxed{0} & \boxed{0} & \cdots & \boxed{0} & \boxed{0} & \cdots & \boxed{1} & \cdots & \boxed{0} \\ \text{Stelle:} & 1 & 2 & & n & n+1 & & n+j & & n+m \end{array}$$
$$\begin{array}{ccccccc} d_j & \boxed{0} & \boxed{0} & \cdots & \boxed{0} & \boxed{0} & \cdots & \boxed{2} & \cdots & \boxed{0} \\ \text{Stelle:} & 1 & 2 & & n & n+1 & & n+j & & n+m \end{array}$$

# $\mathcal{NP}$ -Vollständigkeit von SUBSETSUM (3)

Weitere Zahlen:

Erzeuge  $n + m$ -stellige Zahlen  $c_j, d_j$  für  $j = 1, \dots, m$ :

$$\begin{array}{ccccccc} c_j & \boxed{0} & \boxed{0} & \cdots & \boxed{0} & \boxed{0} & \cdots & \boxed{1} & \cdots & \boxed{0} \\ \text{Stelle:} & 1 & 2 & & n & n+1 & & n+j & & n+m \end{array}$$

$$\begin{array}{ccccccc} d_j & \boxed{0} & \boxed{0} & \cdots & \boxed{0} & \boxed{0} & \cdots & \boxed{2} & \cdots & \boxed{0} \\ \text{Stelle:} & 1 & 2 & & n & n+1 & & n+j & & n+m \end{array}$$

Zielsumme  $s$ :

$$\begin{array}{ccccccc} s & \boxed{1} & \boxed{1} & \cdots & \boxed{1} & \boxed{4} & \boxed{4} & \cdots & \boxed{4} \\ \text{Stelle:} & 1 & 2 & & n & n+1 & n+2 & & n+m \end{array}$$

## $\mathcal{NP}$ -Vollständigkeit von SUBSETSUM (4)

Sei also  $f(F) = ((t_1, \dots, t_n, f_1, \dots, f_n, c_1, \dots, c_m, d_1, \dots, d_m), s)$ .

Es gilt:

- Die Summe jeder Teilmenge der Zahlen erzeugt keine Überträge.
- Die  $n$  Einsen in  $s$  sorgen dafür, dass in  $I$  jeweils  $t_i$  oder  $f_i$  enthalten ist aber nicht beide.
- Die Wahl der  $t_i$  und  $f_i$  in  $I$  zählt gleichzeitig in den Stellen  $n + 1$  bis  $n + m$ , welche Klauseln durch Setzen von  $x_i$  auf 1 (bzw.  $x_i$  auf 0) wahr gemacht werden.
- In der Summe kann dies pro Stelle  $j$  eine Zahl zwischen 0 und 3 sein
- Durch Hinzunahme der  $c_j$  und/oder  $d_j$  kann die Zielsumme 4 pro Stelle  $j$  erreicht werden, wenn mindestens 1 Literal wahr ist.

## $\mathcal{NP}$ -Vollständigkeit von SUBSETSUM (5)

Sei  $I$  Lösung von SUBSETSUM-Instanz

$((t_1, \dots, t_n, f_1, \dots, f_n, c_1, \dots, c_m, d_1, \dots, c_m), s)$ .

- Konstruiere Belegung  $B$  für  $F$ .
- Setze für  $i \in I$  mit  $1 \leq i \leq n$ :  $B(x_i) = 1$
- Setze für  $i \in I$  mit  $n + 1 \leq i \leq n + n$ :  $B(x_i) = 0$ .
- Macht in jeder Klausel mind. 1 Literal wahr

## $\mathcal{NP}$ -Vollständigkeit von SUBSETSUM (5)

Sei  $I$  Lösung von SUBSETSUM-Instanz

$((t_1, \dots, t_n, f_1, \dots, f_n, c_1, \dots, c_m, d_1, \dots, c_m), s)$ .

- Konstruiere Belegung  $B$  für  $F$ .
- Setze für  $i \in I$  mit  $1 \leq i \leq n$ :  $B(x_i) = 1$
- Setze für  $i \in I$  mit  $n + 1 \leq i \leq n + n$ :  $B(x_i) = 0$ .
- Macht in jeder Klausel mind. 1 Literal wahr

Sei  $B$  erfüllende Belegung für  $F$ .

- Konstruiere Indexmenge  $I$
- $I$  enthält den Index von  $t_i$  wenn  $B(x_i) = 1$ ,
- $I$  enthält den Index von  $f_i$  wenn  $B(x_i) = 0$
- $I$  enthält die Indizes der  $c_j, d_j$ , sodass sich die hinteren  $m$  Stellen zu 4 aufsummieren

Immer möglich, da für jede Stelle die Summe schon mindestens 1 ist (da  $B$  erfüllende Belegung).

## $\mathcal{NP}$ -Vollständigkeit von SUBSETSUM (5)

Sei  $I$  Lösung von SUBSETSUM-Instanz

$((t_1, \dots, t_n, f_1, \dots, f_n, c_1, \dots, c_m, d_1, \dots, c_m), s)$ .

- Konstruiere Belegung  $B$  für  $F$ .
- Setze für  $i \in I$  mit  $1 \leq i \leq n$ :  $B(x_i) = 1$
- Setze für  $i \in I$  mit  $n + 1 \leq i \leq n + n$ :  $B(x_i) = 0$ .
- Macht in jeder Klausel mind. 1 Literal wahr

Sei  $B$  erfüllende Belegung für  $F$ .

- Konstruiere Indexmenge  $I$
- $I$  enthält den Index von  $t_i$  wenn  $B(x_i) = 1$ ,
- $I$  enthält den Index von  $f_i$  wenn  $B(x_i) = 0$
- $I$  enthält die Indizes der  $c_j, d_j$ , sodass sich die hinteren  $m$  Stellen zu 4 aufsummieren

Immer möglich, da für jede Stelle die Summe schon mindestens 1 ist (da  $B$  erfüllende Belegung).

Damit folgt: 3-CNF-SAT  $\leq_p$  SUBSETSUM.

## Beispiel

---

$$F = (x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_1)$$

wird übersetzt in:

$$a_1 = t_1 = 110$$

$$a_2 = f_1 = 101$$

$$a_3 = c_1 = 010$$

$$a_4 = c_2 = 001$$

$$a_5 = d_1 = 020$$

$$a_6 = d_2 = 002$$

$$s = 144$$

(Unlösbar)



## Beispiel (2)

$F = (x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3 \vee x_4)$  wird übersetzt in

$a_1$	$= t_1$	$= 100010100$	$a_9$	$= c_1$	$= 000010000$
$a_2$	$= t_2$	$= 010011010$	$a_{10}$	$= c_2$	$= 000001000$
$a_3$	$= t_3$	$= 001000001$	$a_{11}$	$= c_3$	$= 000000100$
$a_4$	$= t_4$	$= 000100101$	$a_{12}$	$= c_4$	$= 000000010$
$a_5$	$= f_1$	$= 100001010$	$a_{13}$	$= c_5$	$= 000000001$
$a_6$	$= f_2$	$= 010000101$	$a_{14}$	$= d_1$	$= 000020000$
$a_7$	$= f_3$	$= 001001010$	$a_{15}$	$= d_2$	$= 000002000$
$a_8$	$= f_4$	$= 000110000$	$a_{16}$	$= d_3$	$= 000000200$
			$a_{17}$	$= d_4$	$= 000000020$
			$a_{18}$	$= d_5$	$= 000000002$
			$s$		$= 111144444$

Lösung  $I = \{2, 4, 5, 7, 9, 10, 11, 12, 13, 14, 16, 18\}$ .

Erfüllende Belegung  $B(x_1) = 0, B(x_2) = 1, B(x_3) = 0, B(x_4) = 1$ .

## Definition (KNAPSACK-Problem)

Das **KNAPSACK-Problem** lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben:  $k$  Gegenstände mit Gewichten  $w_1, \dots, w_k \in \mathbb{N}$  und Nutzenwerten  $n_1, \dots, n_k \in \mathbb{N}$ , sowie zwei Zahlen  $s_w, s_n \in \mathbb{N}$ .

gefragt: Gibt es eine Teilmenge  $I \subseteq \{1, \dots, k\}$ , sodass 
$$\sum_{i \in I} w_i \leq s_w \text{ und } \sum_{i \in I} n_i \geq s_n?$$

## Definition (KNAPSACK-Problem)

Das **KNAPSACK-Problem** lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben:  $k$  Gegenstände mit Gewichten  $w_1, \dots, w_k \in \mathbb{N}$  und Nutzenwerten  $n_1, \dots, n_k \in \mathbb{N}$ , sowie zwei Zahlen  $s_w, s_n \in \mathbb{N}$ .

gefragt: Gibt es eine Teilmenge  $I \subseteq \{1, \dots, k\}$ , sodass 
$$\sum_{i \in I} w_i \leq s_w \text{ und } \sum_{i \in I} n_i \geq s_n?$$

Beachte für  $w_i = n_i$  und  $s_n = s_w$  ergibt sich genau das SUBSETSUM-Problem!

# $\mathcal{NP}$ -Vollständigkeit von KNAPSACK

---

## Satz

KNAPSACK ist  $\mathcal{NP}$ -vollständig.

# $\mathcal{NP}$ -Vollständigkeit von KNAPSACK

## Satz

KNAPSACK ist  $\mathcal{NP}$ -vollständig.

Beweis:

KNAPSACK  $\in \mathcal{NP}$ :

- Rate eine Teilmenge  $I \subseteq \{1, \dots, k\}$  nichtdeterministisch
- Prüfe deterministisch, ob  $\sum_{i \in I} w_i \leq s_w$  und  $\sum_{i \in I} n_i \geq s_n$
- Daher kann KNAPSACK in Polynomialzeit auf einer NTM entschieden werden.

## Satz

KNAPSACK ist  $\mathcal{NP}$ -vollständig.

Beweis:

KNAPSACK  $\in \mathcal{NP}$ :

- Rate eine Teilmenge  $I \subseteq \{1, \dots, k\}$  nichtdeterministisch
- Prüfe deterministisch, ob  $\sum_{i \in I} w_i \leq s_w$  und  $\sum_{i \in I} n_i \geq s_n$
- Daher kann KNAPSACK in Polynomialzeit auf einer NTM entschieden werden.

KNAPSACK ist  $\mathcal{NP}$ -hart:

- Sei  $((a_1, \dots, a_k), s)$  eine SUBSETSUM-Instanz
- Sei  $f((a_1, \dots, a_k), s) = ((w_1, \dots, w_k), (n_1, \dots, n_k), s_w, s_m)$  mit  $w_i = a_i, n_i = a_i$  für  $i = 1, \dots, k$  und  $s_w = s$  und  $s_m = s$ .
- $((a_1, \dots, a_k), s)$  lösbar g.d.w.  $f((a_1, \dots, a_k), s)$  lösbar
- $f$  ist in polynomieller Zeit von einer DTM berechenbar.
- SUBSETSUM  $\leq_p$  KNAPSACK.

## Definition (PARTITION-Problem)

Das **PARTITION-Problem** lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Natürliche Zahlen  $a_1, \dots, a_k \in \mathbb{N}$

gefragt: Gibt es eine Teilmenge  $I \subseteq \{1, \dots, k\}$ , sodass

$$\sum_{i \in I} a_i = \sum_{i \in \{1, \dots, k\} \setminus I} a_i?$$

## Satz

PARTITION ist  $\mathcal{NP}$ -vollständig.

Beweis:

$\text{PARTITION} \in \mathcal{NP}$

- Rate nichtdeterministisch  $I \subseteq \{1, \dots, k\}$
- Prüfe deterministisch, ob  $\sum_{i \in I} a_i = \sum_{i \in \{1, \dots, k\} \setminus I} a_i$
- Daher kann PARTITION in Polynomialzeit auf einer NTM entschieden werden.



## $\mathcal{NP}$ -Vollständigkeit von PARTITION (2)

### PARTITION ist $\mathcal{NP}$ -hart

- Wir zeigen: SUBSETSUM  $\leq_p$  PARTITION
- Sei  $f((a_1, \dots, a_k), s) = (a_1, \dots, a_k, a_{k+1}, a_{k+2})$   
mit  $a_{k+1} = A + s$  und  $a_{k+2} = 2A - s$ , wobei  $A = \sum_{i=1}^k a_i$ .
- $f$  kann von einer DTM in polynomieller Zeit berechnet werden.
- Wenn  $(a_1, \dots, a_k, a_{k+1}, a_{k+2})$  Lösung  $I \subseteq \{1, \dots, k+2\}$  hat, dann  $\sum_{i \in I} a_i = 2A = \sum_{i \in \{1, \dots, k+2\} \setminus I} a_i$ .
  - Nicht  $k+1$  und  $k+2$  in  $I$ , da sonst die Summe zu groß ist.
  - Wenn  $k+2 \in I$ , dann  $I' = I \setminus \{k+2\}$ .
  - Wenn  $k+1 \in I$ , dann  $I' = \{1, \dots, k\} \setminus I$ .
  - In beiden Fällen ist  $\sum_{i \in I'} = s$  und  $I'$  ist Lösung von  $((a_1, \dots, a_k), s)$ .
- Wenn  $I \subseteq \{1, \dots, k\}$  eine Lösung von  $((a_1, \dots, a_k), s)$ , dann ist  $I \cup \{k+2\}$  eine Lösung für  $(a_1, \dots, a_k, a_{k+1}, a_{k+2})$ .
- Daher gilt  $((a_1, \dots, a_k), s)$  lösbar g.d.w.  $f((a_1, \dots, a_k), s)$  lösbar
- SUBSETSUM  $\leq_p$  PARTITION

- Sei  $((1,2,3,4,5,6),14)$  eine SUBSETSUM-Instanz.
- PARTITION-Instanz dazu:  $(1,2,3,4,5,6,35,28)$
- Lösung  $I = \{1, 3, 4, 6, 8\}$  da  
 $1 + 3 + 4 + 6 + 28 = 42 = 2 + 5 + 35.$
- Lösung der SUBSETSUM-Instanz  $\{1, 3, 4, 6\}$

## Definition (BINPACKING-Problem)

Das **BINPACKING-Problem** lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Natürliche Zahlen  $a_1, \dots, a_k \in \mathbb{N}$ , die Behältergröße  $b \in \mathbb{N}$ , und die Anzahl der Behälter  $m$

gefragt: Kann man alle gegebenen Zahlen so auf die Behälter aufteilen, sodass keiner der Behälter überläuft? Formal: Gibt es eine totale Funktion  $assign : \{1, \dots, k\} \rightarrow \{1, \dots, m\}$ , sodass für alle  $j \in \{1, \dots, m\}$  gilt:  $\sum_{assign(i)=j} a_i \leq b$ ?

## Satz

BINPACKING ist  $\mathcal{NP}$ -vollständig.

Beweis: BINPACKING  $\in \mathcal{NP}$

- Rate nichtdeterministisch für jede Zahl  $a_i$  in welchen Behälter sie gehört.
- Prüfe, dass die geratene Zuordnung keinen Behälter überlaufen lässt.
- BINPACKING kann daher in Polynomialzeit auf einer NTM entschieden werden.

# $\mathcal{NP}$ -Vollständigkeit von BINPACKING (2)

## BINPACKING ist $\mathcal{NP}$ -hart

- Sei  $(a_1, \dots, a_k)$  eine PARTITION-Instanz.
- Sei  $f(a_1, \dots, a_k)$  die BINPACKING-Instanz mit Zahlen  $a_1, \dots, a_k$ , Behältergröße  $b = \lfloor \frac{\sum_{i=1}^k a_i}{2} \rfloor$  und  $m = 2$  Behältern.
- Wenn  $\sum_{i=1}^k a_i$  ungerade, dann ist die PARTITION-Instanz unlösbar und die BINPACKING-Instanz ebenso.
- Wenn  $I \subseteq \{1, \dots, k\}$  Lösung für PARTITION, dann ist

$$\text{assign}(i) = \begin{cases} 1, & \text{wenn } i \in I \\ 2, & \text{wenn } i \notin I \end{cases}$$

eine Lösung für BINPACKING.

- Mit  $I = \{i \mid 1 \leq i \leq k, \text{assign}(i) = 1\}$  kann aus Lösung für BINPACKING eine Lösung für PARTITION erstellt werden.
- $f$  ist polynomial berechenbar: PARTITION  $\leq_p$  BINPACKING.

## Definition

In einem gerichteten Graphen ist ein **Hamilton-Kreis**, ein Kreis, der genau alle Knoten einmal besucht.

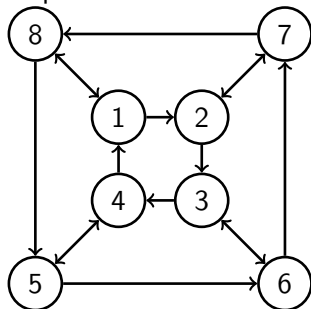
Formal: Für  $G = (V, E)$  mit  $V = \{v_1, \dots, v_n\}$  ist ein Hamilton-Kreis eine Permutation  $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ , sodass  $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$  und  $(v_{\pi(n)}, v_{\pi(1)}) \in E$ .

## Definition

In einem gerichteten Graphen ist ein **Hamilton-Kreis**, ein Kreis, der genau alle Knoten einmal besucht.

Formal: Für  $G = (V, E)$  mit  $V = \{v_1, \dots, v_n\}$  ist ein Hamilton-Kreis eine Permutation  $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ , sodass  $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$  und  $(v_{\pi(n)}, v_{\pi(1)}) \in E$ .

Beispiel:

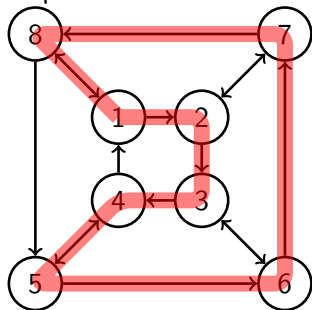


## Definition

In einem gerichteten Graphen ist ein **Hamilton-Kreis**, ein Kreis, der genau alle Knoten einmal besucht.

Formal: Für  $G = (V, E)$  mit  $V = \{v_1, \dots, v_n\}$  ist ein Hamilton-Kreis eine Permutation  $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ , sodass  $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$  und  $(v_{\pi(n)}, v_{\pi(1)}) \in E$ .

Beispiel:





## Definition (DIRECTED-HAMILTON-CYCLE-Problem)

Das **DIRECTED-HAMILTON-CYCLE-Problem** lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Ein gerichteter Graph  $G = (V, E)$  mit  $V = \{v_1, \dots, v_n\}$

gefragt: Gibt es einen Hamilton-Kreis in  $G$ ?

## Satz

DIRECTED-HAMILTON-CYCLE ist  $\mathcal{NP}$ -vollständig.

Beweis: **DIRECTED-HAMILTON-CYCLE**  $\in \mathcal{NP}$ :

- Rate nichtdeterministisch die Permutation  $\pi$
- Verifiziere, ob  $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$  und  $(v_{\pi(n)}, v_{\pi(1)}) \in E$  gilt.
- DIRECTED-HAMILTON-CYCLE kann daher kann auf einer NTM in Polynomialzeit entschieden werden.

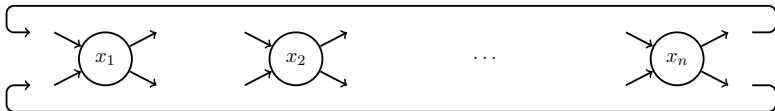
### DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$ -hart

- Ziel:  $3\text{-CNF-SAT} \leq_p \text{DIRECTED-HAMILTON-CYCLE}$
- Sei  $F = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, sodass jede Klausel  $K_i$  genau 3 Literale enthält. Sei  $\text{Var}(F) = \{x_1, \dots, x_n\}$

## $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (2)

### DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$ -hart

- Ziel:  $3\text{-CNF-SAT} \leq_p \text{DIRECTED-HAMILTON-CYCLE}$
- Sei  $F = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, sodass jede Klausel  $K_i$  genau 3 Literale enthält. Sei  $\text{Var}(F) = \{x_1, \dots, x_n\}$
- Erzeuge zunächst



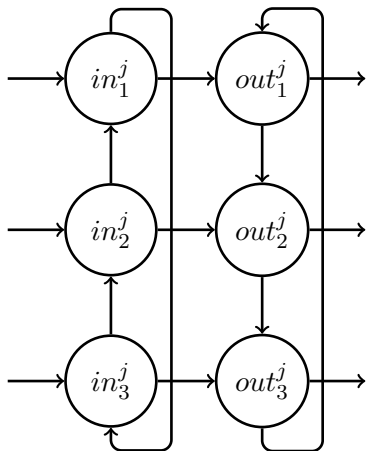
- Idee: Hamilton-Kreis läuft oben durch  $x_i$ , wenn  $I(x_i) = 1$  und unten durch  $x_i$ , wenn  $I(x_i) = 0$ .

# $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist  $\mathcal{NP}$ -hart

Teilgraphen  $K_j$  für jede Klausel  $K_j$ :

Eigenschaften:



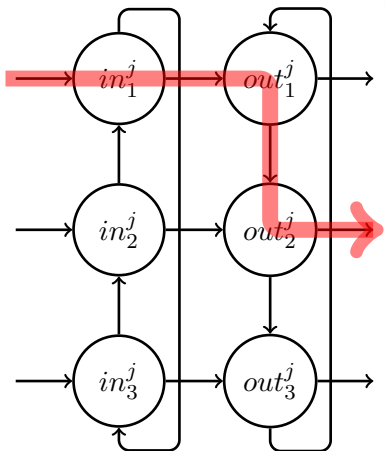
# $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

## DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$ -hart

Teilgraphen  $K_j$  für jede Klausel  $K_j$ :

Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang  $in_i^j$  in  $K_j$  hereingeht, muss  $K_j$  durch den Knoten  $out_i^j$  verlassen.



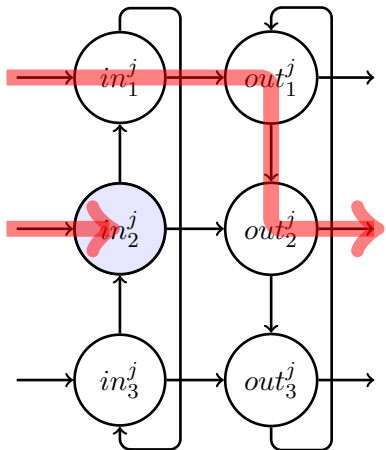
# $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist  $\mathcal{NP}$ -hart

Teilgraphen  $K_j$  für jede Klausel  $K_j$ :

Eigenschaften:

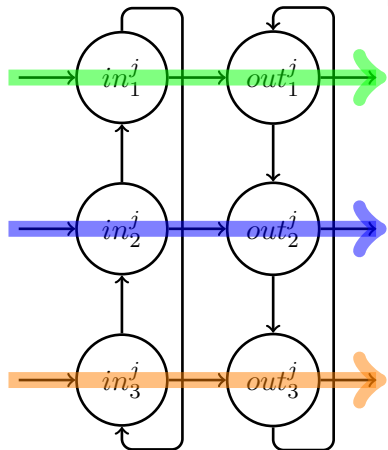
- Jeder Hamilton-Kreis, der durch Eingang  $in_i^j$  in  $K_j$  hereingeht, muss  $K_j$  durch den Knoten  $out_i^j$  verlassen. (sonst bleibt man stecken)



# $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist  $\mathcal{NP}$ -hart

Teilgraphen  $K_j$  für jede Klausel  $K_j$ :



Eigenschaften:

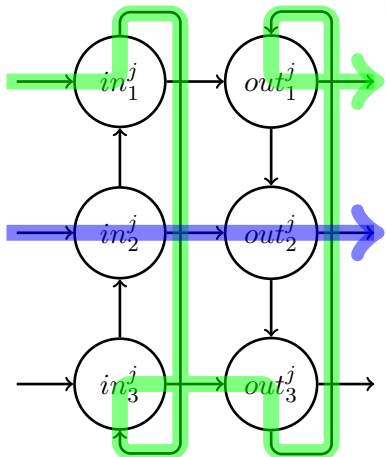
- Jeder Hamilton-Kreis, der durch Eingang  $in_i^j$  in  $K_j$  hereingeht, muss  $K_j$  durch den Knoten  $out_i^j$  verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden



# $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist  $\mathcal{NP}$ -hart

Teilgraphen  $K_j$  für jede Klausel  $K_j$ :



Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang  $in_i^j$  in  $K_j$  hereingeht, muss  $K_j$  durch den Knoten  $out_i^j$  verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden

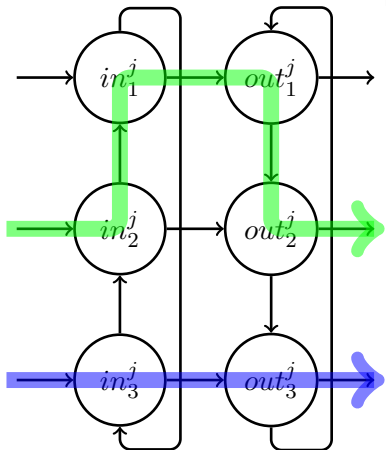
# $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist  $\mathcal{NP}$ -hart

Teilgraphen  $K_j$  für jede Klausel  $K_j$ :

Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang  $in_i^j$  in  $K_j$  hereingeht, muss  $K_j$  durch den Knoten  $out_i^j$  verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden



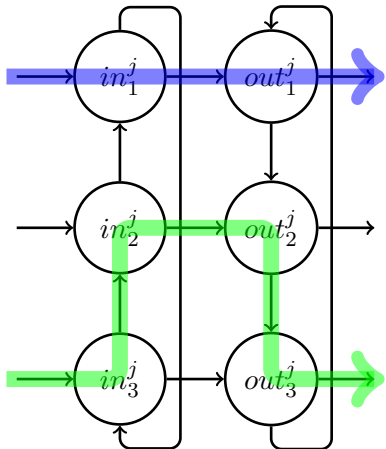
# $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

DIRECTED-HAMILTON-CYCLE ist  $\mathcal{NP}$ -hart

Teilgraphen  $K_j$  für jede Klausel  $K_j$ :

Eigenschaften:

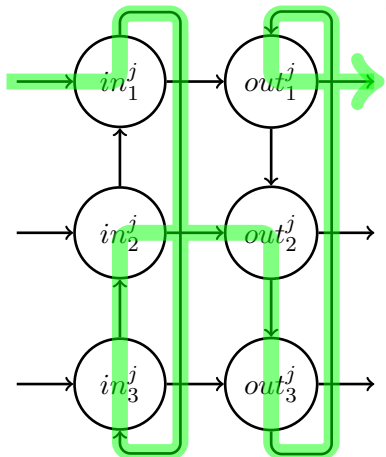
- Jeder Hamilton-Kreis, der durch Eingang  $in_i^j$  in  $K_j$  hereingeht, muss  $K_j$  durch den Knoten  $out_i^j$  verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden



# $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

## DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$ -hart

Teilgraphen  $K_j$  für jede Klausel  $K_j$ :



Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang  $in_i^j$  in  $K_j$  hereingeht, muss  $K_j$  durch den Knoten  $out_i^j$  verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden

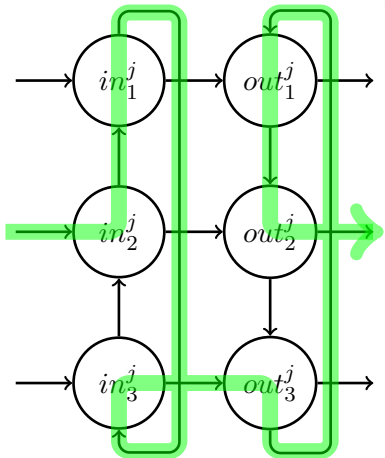
# $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

## DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$ -hart

Teilgraphen  $K_j$  für jede Klausel  $K_j$ :

Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang  $in_i^j$  in  $K_j$  hereingeht, muss  $K_j$  durch den Knoten  $out_i^j$  verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden



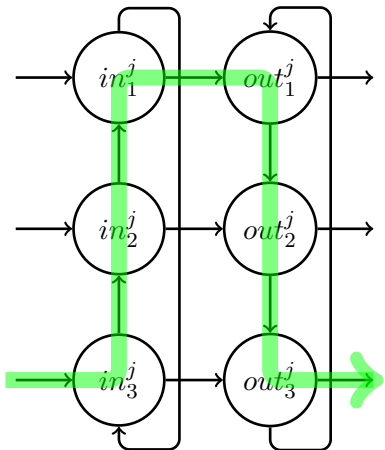
# $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

## DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$ -hart

Teilgraphen  $K_j$  für jede Klausel  $K_j$ :

Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang  $in_i^j$  in  $K_j$  hereingeht, muss  $K_j$  durch den Knoten  $out_i^j$  verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden



# $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (3)

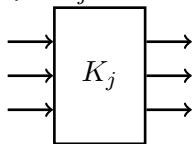
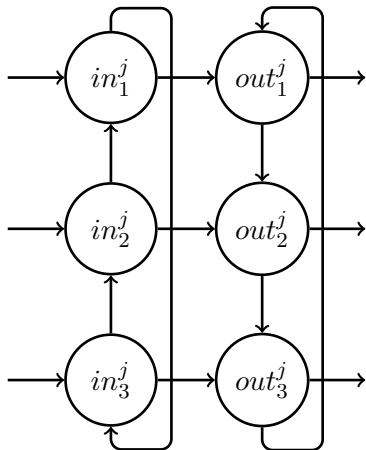
## DIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$ -hart

Teilgraphen  $K_j$  für jede Klausel  $K_j$ :

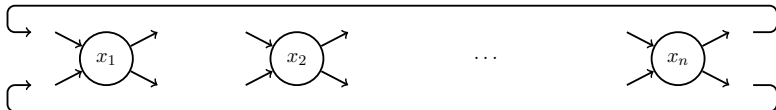
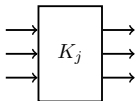
Eigenschaften:

- Jeder Hamilton-Kreis, der durch Eingang  $in_i^j$  in  $K_j$  hereingeht, muss  $K_j$  durch den Knoten  $out_i^j$  verlassen. (sonst bleibt man stecken)
- Der Graph kann einmal, zweimal oder dreimal in einem Hamilton-Kreis durchlaufen werden

Teilgraph  $K_j$  abstrakt als Bauteil:



## $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (4)

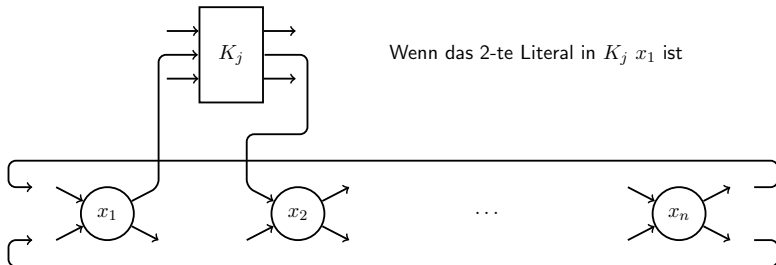


Verbindungen:

- $i$ -ter Ein-/Ausgang von  $K_j$  wird zwischen  $x_k$  und  $x_{k+1}$  verbunden, wenn  
 $i$ -tes Literal in Klausel  $K_j$  ist  $x_k$  oder  $\neg x_k$
- obere Verbindung, wenn Literal  $x_k$  ist
- untere Verbindung, wenn Literal  $\neg x_k$  ist
- mehrere  $K_j$  hintereinander erlaubt.



## $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (4)



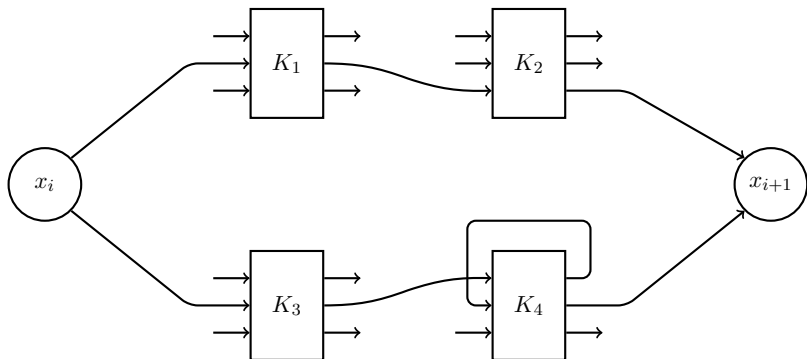
Verbindungen:

- $i$ -ter Ein-/Ausgang von  $K_j$  wird zwischen  $x_k$  und  $x_{k+1}$  verbunden, wenn  $i$ -tes Literal in Klausel  $K_j$  ist  $x_k$  oder  $\neg x_k$
- obere Verbindung, wenn Literal  $x_k$  ist
- untere Verbindung, wenn Literal  $\neg x_k$  ist
- mehrere  $K_j$  hintereinander erlaubt.

## $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (5)

Beispiel:  $F = K_1 \wedge K_2 \wedge K_3 \wedge K_4$  eine 3-CNF, sodass

- $x_i$  kommt an 2. Position in  $K_1$  vor
- $x_i$  kommt an 3. Position in  $K_2$  vor
- $\neg x_i$  kommt an 2. Position in  $K_3$  vor
- $\neg x_i$  kommt an 1. und 2. Position in  $K_4$  vor



## $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (6)

---

- Konstruktion des Graph in Polynomialzeit möglich:  
Füge iterativ die  $K_j$ -Graphen ein.
- Alle Anschlüsse werden angeschlossen: der Graph ist wohl geformt.

## $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (6)

- Konstruktion des Graph in Polynomialzeit möglich:  
Füge iterativ die  $K_j$ -Graphen ein.
- Alle Anschlüsse werden angeschlossen: der Graph ist wohl geformt.
- Wenn der Graph einen Hamilton-Kreis hat, dann läuft der Kreis entweder oben durch  $x_i$  oder unten durch  $x_i$ .
  - Das liefert Belegung  $I(x_i) = 1$  bzw.  $I(x_i) = 0$
  - Da jedes  $K_j$  mindestens einmal von Hamilton-Kreis durchlaufen wird: Jede Klausel wird durch  $I$  erfüllt.

## $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (6)

- Konstruktion des Graph in Polynomialzeit möglich:  
Füge iterativ die  $K_j$ -Graphen ein.
- Alle Anschlüsse werden angeschlossen: der Graph ist wohl geformt.
- Wenn der Graph einen Hamilton-Kreis hat, dann läuft der Kreis entweder oben durch  $x_i$  oder unten durch  $x_i$ .
  - Das liefert Belegung  $I(x_i) = 1$  bzw.  $I(x_i) = 0$
  - Da jedes  $K_j$  mindestens einmal von Hamilton-Kreis durchlaufen wird: Jede Klausel wird durch  $I$  erfüllt.
- Wenn  $I$  Belegung mit  $I(F) = 1$ , dann gibt es einen Hamilton-Kreis:
  - Durchlaufe Knoten  $x_i$  oben, wenn  $I(x_i) = 1$   
und unten, wenn  $I(x_i) = 0$ .
  - Durchlaufe  $K_j$  ein bis drei Mal, je nachdem welche und wie viele Literale in der Klausel  $K_j$  durch  $I$  wahr gemacht werden.

## $\mathcal{NP}$ -Vollständigkeit von DIRECTED-HAMILTON-CYCLE (6)

- Konstruktion des Graph in Polynomialzeit möglich:  
Füge iterativ die  $K_j$ -Graphen ein.
- Alle Anschlüsse werden angeschlossen: der Graph ist wohl geformt.
- Wenn der Graph einen Hamilton-Kreis hat, dann läuft der Kreis entweder oben durch  $x_i$  oder unten durch  $\bar{x}_i$ .
  - Das liefert Belegung  $I(x_i) = 1$  bzw.  $I(x_i) = 0$
  - Da jedes  $K_j$  mindestens einmal von Hamilton-Kreis durchlaufen wird: Jede Klausel wird durch  $I$  erfüllt.
- Wenn  $I$  Belegung mit  $I(F) = 1$ , dann gibt es einen Hamilton-Kreis:
  - Durchlaufe Knoten  $x_i$  oben, wenn  $I(x_i) = 1$   
und unten, wenn  $I(x_i) = 0$ .
  - Durchlaufe  $K_j$  ein bis drei Mal, je nachdem welche und wie viele Literale in der Klausel  $K_j$  durch  $I$  wahr gemacht werden.
- Daher  $3\text{-CNF-SAT} \leq_p \text{DIRECTED-HAMILTON-CYCLE}$

## Hamilton-Kreis im ungerichteten Graph

Sei  $G = (V, E)$  ein ungerichteter Graph. Ein **Hamilton-Kreis** ist eine Permutation  $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ , sodass  $\{v_{\pi(i)}, v_{\pi(i+1)}\} \in E$  und  $\{v_{\pi(n)}, v_{\pi(1)}\} \in E$ ?

## Definition (UNDIRECTED-HAMILTON-CYCLE-Problem)

Das **UNDIRECTED-HAMILTON-CYCLE-Problem** lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Ein ungerichteter Graph  $G = (V, E)$ .

gefragt: Gibt es einen Hamilton-Kreis in  $G$ ?

## Satz

UNDIRECTED-HAMILTON-CYCLE ist  $\mathcal{NP}$ -vollständig.

Beweis:  $\text{UNDIRECTED-HAMILTON-CYCLE} \in \mathcal{NP}$ :

- Rate die Permutation  $\pi$  nichtdeterministisch
- Verifiziere, ob  $\{v_{\pi(i)}, v_{\pi(i+1)}\} \in E$  und  $\{v_{\pi(n)}, v_{\pi(1)}\} \in E$  gilt.
- Dies kann auf einer NTM in Polynomialzeit durchgeführt werden.

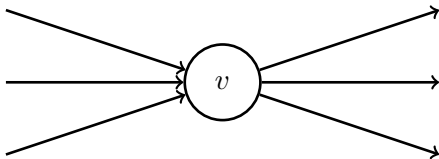


# $\mathcal{NP}$ -Vollständigkeit von UNDIRECTED-HAMILTON-CYCLE

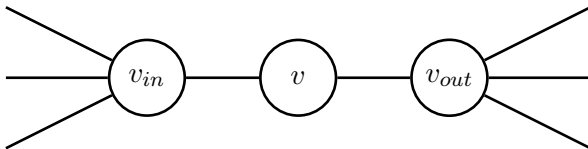
## UNDIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$ -hart:

- Sei  $f$  die Funktion, die aus einem gerichteten Graphen  $(V, E)$  einen ungerichteten Graphen macht, sodass
$$f(V) = \bigcup \{ \{v_{in}, v, v_{out}\} \mid v \in V \}$$
 und
$$f(E) = \{ \{u_{out}, v_{in}\} \mid (u, v) \in E \} \cup \bigcup \{ \{ \{v_{in}, v\}, \{v, v_{out}\} \} \mid v \in V \}.$$

D.h. jeder Knoten  $v$  mit Ein- und Ausgängen:



wird ersetzt durch



## UNDIRECTED-HAMILTON-CYCLE ist $\mathcal{NP}$ -hart:

- In  $f(G)$  kann der Knoten  $v$  durch einen Hamilton-Kreis nur in einer Richtung durchlaufen werden (von  $v_{in}$  durch  $v$  durch  $v_{out}$  oder umgekehrt).
- Falls  $v_{in,\pi(1)}, v_{\pi(1)}, v_{out,\pi(1)}, \dots, v_{in,\pi(n)}, v_{\pi(n)}, v_{out,\pi(n)}, v_{in,\pi(1)}$  oder  $v_{out,\pi(1)}, v_{\pi(1)}, v_{in,\pi(1)}, \dots, v_{out,\pi(n)}, v_{\pi(n)}, v_{in,\pi(n)}, v_{out,\pi(1)}$  ein ungerichteter Hamilton-Kreis, dann ist  $v_{\pi(1)}, \dots, v_{\pi(n)}, v_{\pi(1)}$  gerichteter Hamilton Kreis.
- Umgekehrte Richtung: Trivial:
- Da  $f$  in Polynomialzeit berechenbar ist, folgt  $\text{DIRECTED-HAMILTON-CYCLE} \leq_p \text{UNDIRECTED-HAMILTON-CYCLE}$ .

## Definition (TRAVELLING-SALESMAN-Problem)

Das TRAVELLING-SALESMAN-Problem lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Ein Menge von Knoten (Städten)  $V = \{v_1, \dots, v_n\}$ , eine  $(n \times n)$ -Matrix  $(M_{i,j})$  mit Entfernungen  $M_{i,j} \in \mathbb{N}$  zwischen den Städten  $v_i$  und  $v_j$ , sowie eine Zahl  $k \in \mathbb{N}$ .

gefragt: Gibt es eine Rundreise, die alle Städte besucht, der Startort gleich dem Zielort ist, und nicht länger als  $k$  ist?

Formal: Gibt es eine Permutation  $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ , sodass  $(\sum_{i=1}^{n-1} M_{\pi(i), \pi(i+1)}) + M_{\pi(n)+\pi(1)} \leq k$ ?

Interaktive Webanwendung zu TSP:

[https://www-m9.ma.tum.de/games/tsp-game/index\\_de.html](https://www-m9.ma.tum.de/games/tsp-game/index_de.html)

## Satz

Das TRAVELLING-SALESMAN-Problem ist  $\mathcal{NP}$ -vollständig.

Beweis: TRAVELLING-SALESMAN  $\in \mathcal{NP}$ :

- Rate die Permutation  $\pi$  nichtdeterministisch
- Prüfe, ob  $(\sum_{i=1}^{n-1} M_{\pi(i),\pi(i+1)}) + M_{\pi(n)+\pi(1)} \leq k$  gilt.
- Dies kann auf einer NTM in Polynomialzeit durchgeführt werden.

# $\mathcal{NP}$ -Vollständigkeit von TRAVELLING-SALESMAN

TRAVELLING-SALESMAN ist  $\mathcal{NP}$ -hart:

Sei  $G = (V, E)$  mit  $V = \{1, \dots, n\}$ . Dann sei  $f(G) = (V, (M_{i,j}, n))$  mit

$$M_{i,j} = \begin{cases} 1, & \text{wenn } \{i, j\} \in E \\ 2, & \text{wenn } \{i, j\} \notin E \end{cases}$$

- Wenn  $G$  einen ungerichteten Hamilton-Kreis hat, dann ist das eine Rundreise der Länge  $n$ .
- Wenn  $f(G)$  eine Rundreise der Länge  $\leq n$  hat, dann muss die Länge genau  $n$  sein, und es können nur Kanten mit Entfernung 1 verwendet werden. Daher hat  $G$  einen ungerichteten Hamilton-Kreis.

Da  $f$  Polynomialzeit von einer DTM berechnet werden kann:

UNDIRECTED-HAMILTON-CYCLE  $\leq_p$  TRAVELLING-SALESMAN.

## Definition (GRAPH-COLORING-Problem)

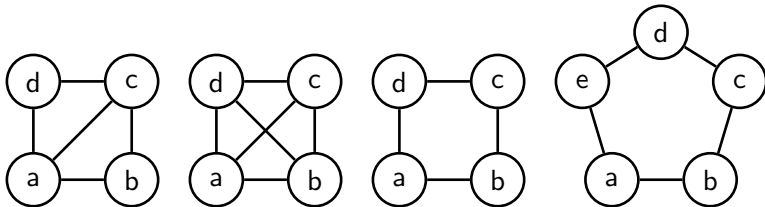
Das GRAPH-COLORING-Problem lässt sich in der gegeben/gefragt-Notation wie folgt formulieren:

gegeben: Ein ungerichteter Graph  $G = (V, E)$  und eine Zahl  $k \in \mathbb{N}$ .

gefragt: Gibt es Färbung der Knoten in  $V$  mit  $\leq k$  Farben, sodass keine zwei benachbarten Knoten in  $G$ , die gleiche Farbe erhalten?

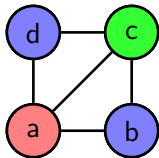
# Beispiel

---

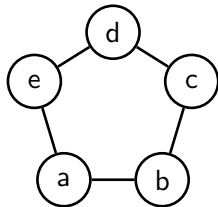
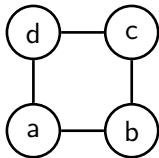
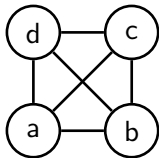


Wie viele Farben sind jeweils minimal notwendig?

# Beispiel



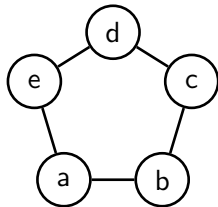
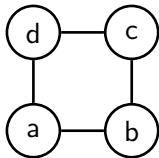
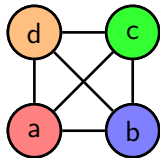
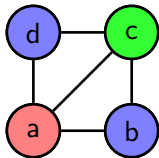
3



Wie viele Farben sind jeweils minimal notwendig?

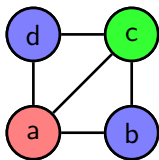


# Beispiel

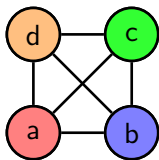


Wie viele Farben sind jeweils minimal notwendig?

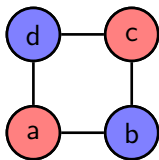
# Beispiel



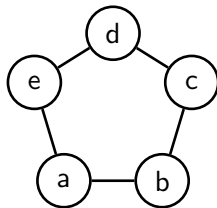
3



4

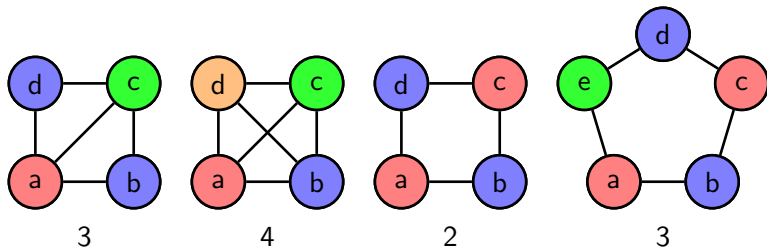


2



Wie viele Farben sind jeweils minimal notwendig?

# Beispiel



Wie viele Farben sind jeweils minimal notwendig?

## Satz

GRAPH-COLORING ist  $\mathcal{NP}$ -vollständig.

Beweis: GRAPH-COLORING  $\in \mathcal{NP}$

- Rate nichtdeterministisch die Farbe aus  $\{1, \dots, k\}$  für jeden Knoten  $v \in V$ .
- Prüfe, dass die Farben von  $u$  und  $v$  stets verschieden sind, für alle  $\{u, v\} \in E$ .  
Das geht in (deterministischer) Polynomialzeit.
- Daher kann GRAPH-COLORING auf einer NTM in polynomieller Zeit entschieden werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (2)

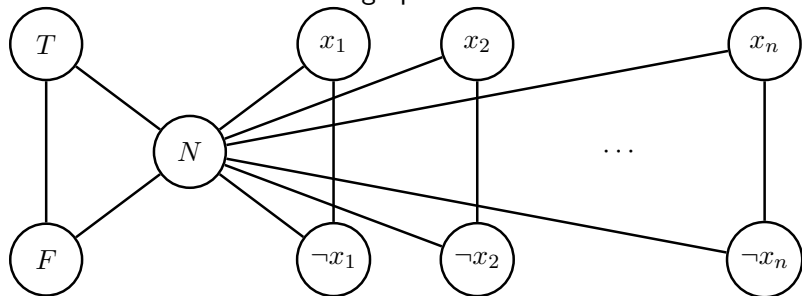
---

GRAPH-COLORING ist  $\mathcal{NP}$ -hart.

- Ziel:  $3\text{-CNF-SAT} \leq_p \text{GRAPH-COLORING}$
- Sei  $F = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, sodass jede Klausel  $K_i$  genau 3 Literale enthält
- Sei  $\text{Var}(F) = \{x_1, \dots, x_n\}$ .
- Wir erzeugen ein GRAPH-COLORING Problem mit  $k = 3$

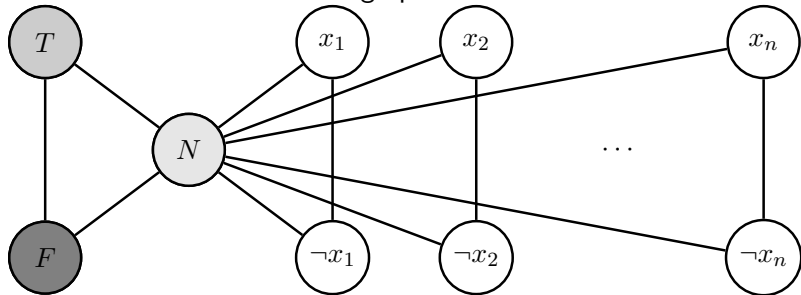
## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

Konstruiere zunächst den Teilgraphen



## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

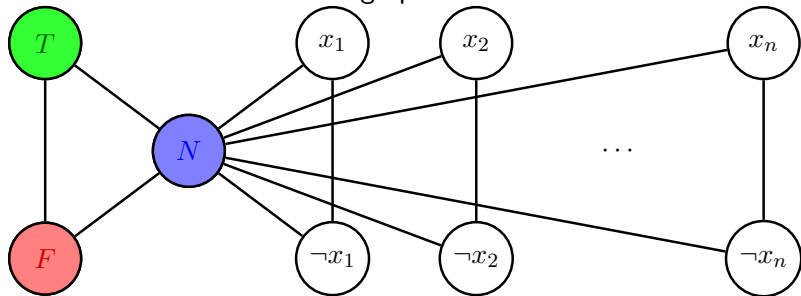
Konstruiere zunächst den Teilgraphen



- Wenn Graph 3-färbbar ist, dann müssen  $T$ ,  $N$ ,  $F$  mit allen drei Farben gefärbt werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

Konstruiere zunächst den Teilgraphen

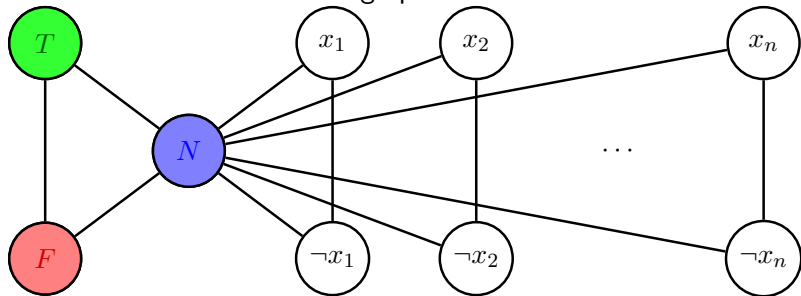


- Wenn Graph 3-färbbar ist, dann müssen  $T, N, F$  mit allen drei Farben gefärbt werden.
- O.B.d.A. seien dies genau die Farben  $T, N, F$  mit genau den Knoten



## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

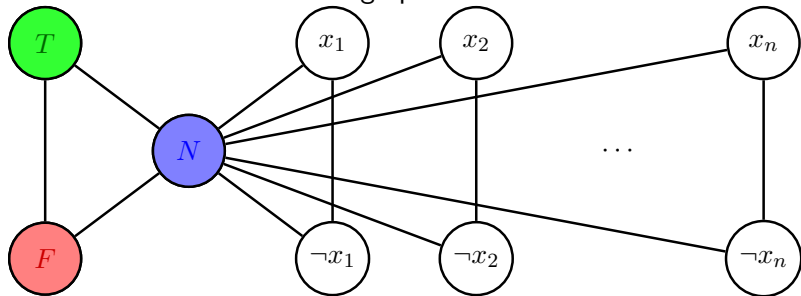
Konstruiere zunächst den Teilgraphen



- Wenn Graph 3-färbbar ist, dann müssen  $T, N, F$  mit allen drei Farben gefärbt werden.
- O.B.d.A. seien dies genau die Farben  $T, N, F$  mit genau den Knoten
- Für 3-Färbung muss  $(x_i, \neg x_i)$  mit  $(T, F)$  oder  $(F, T)$  gefärbt werden.

# $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

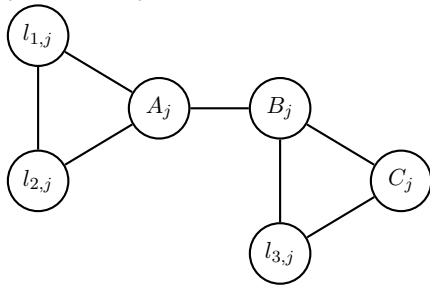
Konstruiere zunächst den Teilgraphen



- Wenn Graph 3-färbbar ist, dann müssen  $T, N, F$  mit allen drei Farben gefärbt werden.
- O.B.d.A. seien dies genau die Farben  $T, N, F$  mit genau den Knoten
- Für 3-Färbung muss  $(x_i, \neg x_i)$  mit  $(T, F)$  oder  $(F, T)$  gefärbt werden.
- Daher: 3-Färbung erzeugt Belegung  $I(x_i) = 1$ , wenn  $x_i$  mit  $T$  gefärbt,  $I(x_i) = 0$ , wenn  $x_i$  mit  $F$  gefärbt.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

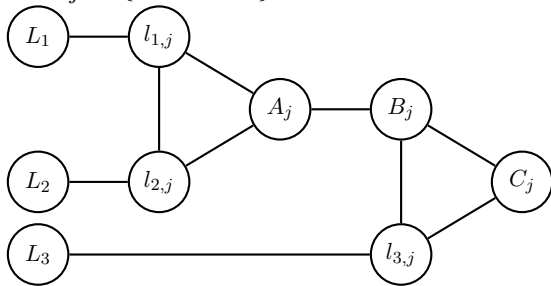
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

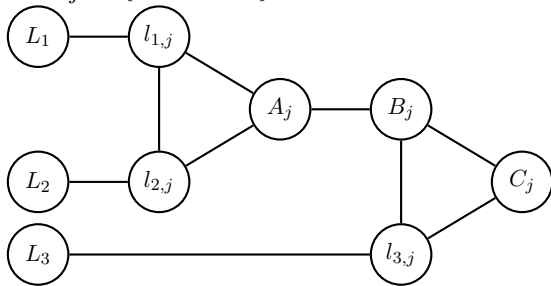
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

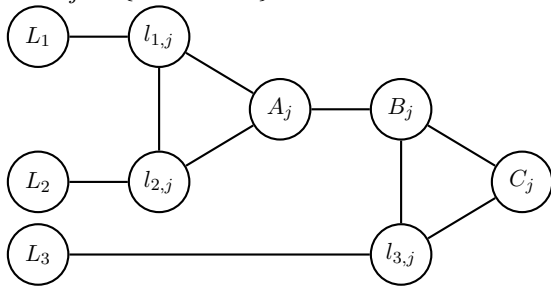
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

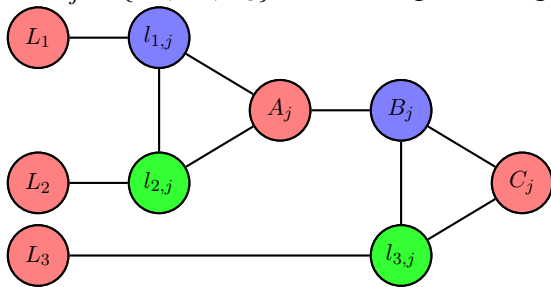
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

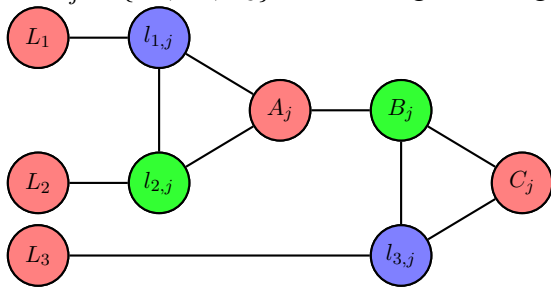
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:

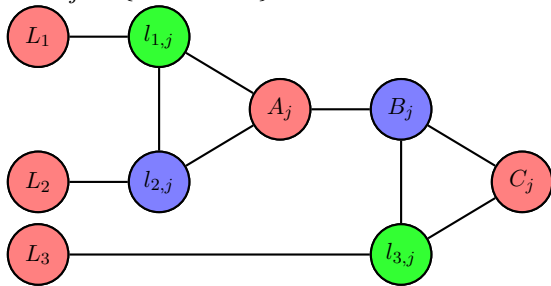


- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.



## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

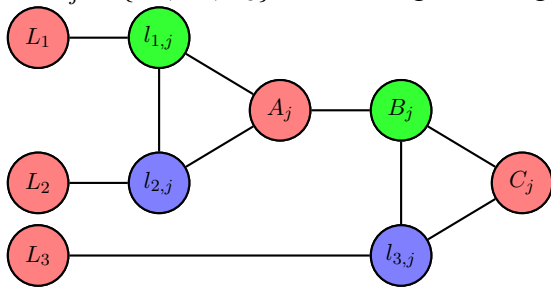
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

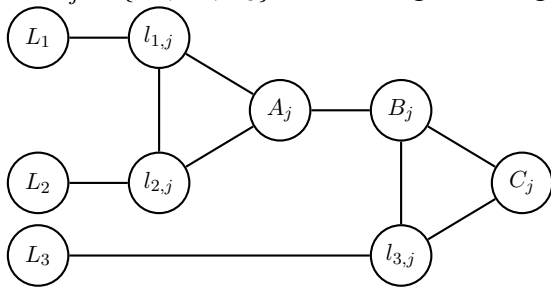
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

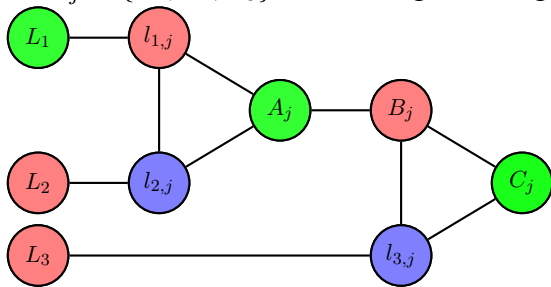
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.
- Wenn mind. ein  $L_i$  mit  $T$ , kann  $C_j$  mit  $T$  gefärbt werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

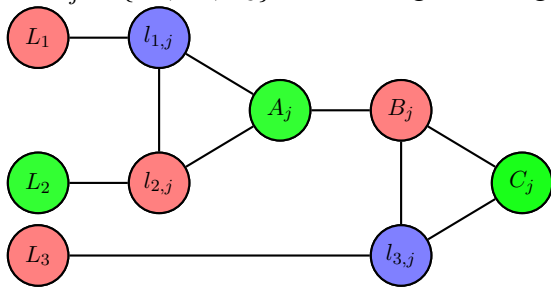
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.
- Wenn mind. ein  $L_i$  mit  $T$ , kann  $C_j$  mit  $T$  gefärbt werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

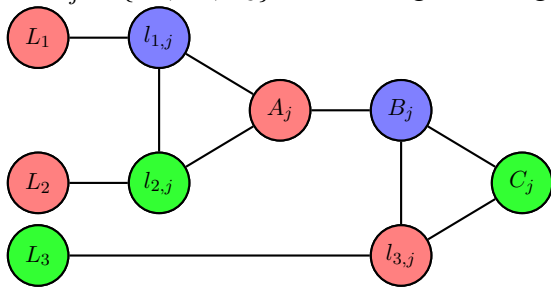
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.
- Wenn mind. ein  $L_i$  mit  $T$ , kann  $C_j$  mit  $T$  gefärbt werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

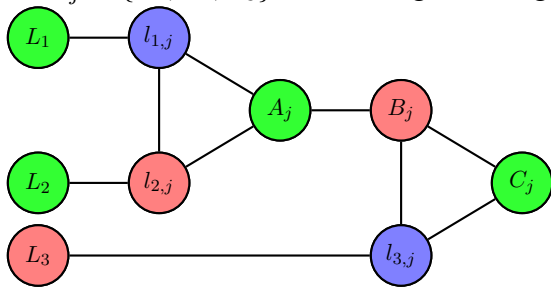
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.
- Wenn mind. ein  $L_i$  mit  $T$ , kann  $C_j$  mit  $T$  gefärbt werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

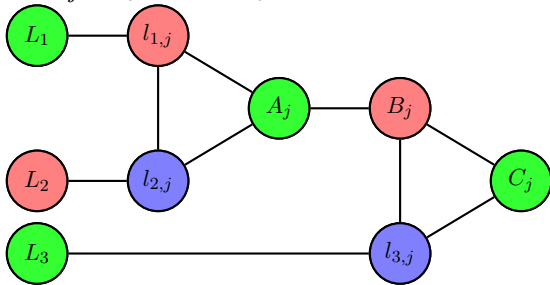
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.
- Wenn mind. ein  $L_i$  mit  $T$ , kann  $C_j$  mit  $T$  gefärbt werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:

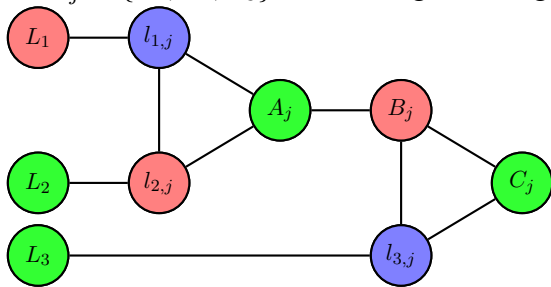


- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.
- Wenn mind. ein  $L_i$  mit  $T$ , kann  $C_j$  mit  $T$  gefärbt werden.



## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

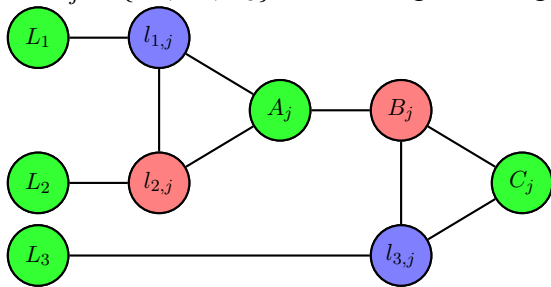
Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.
- Wenn mind. ein  $L_i$  mit  $T$ , kann  $C_j$  mit  $T$  gefärbt werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (3)

Für jede Klausel  $K_j = \{L_1, L_2, L_3\}$  wird der folgende Teilgraph  $G_j$  erzeugt:



- Idee:  $G_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.
- Verbinde  $l_{i,j}$  mit  $x_k$  oder  $\neg x_k$  aus dem anderen Graphen.
- Farben der  $L_i$  daher  $F$  oder  $T$ .
- Wenn alle  $L_i$  mit  $F$  gefärbt, kann  $C_j$  nicht mit  $T$  (sondern nur mit  $F$ ) gefärbt werden.
- Wenn mind. ein  $L_i$  mit  $T$ , kann  $C_j$  mit  $T$  gefärbt werden.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (4)

---

- Erzeuge alle Teilgraphen  $G_j$  für  $K_1, \dots, K_m$
- Verbinde  $l_{i,j}$  mit  $x_k$  wenn  $L_{i,j} = x_k$  und mit  $\neg x_k$  wenn  $L_{i,j} = \neg x_k$
- Verbinde jeweils  $C_j$  mit  $N$  und  $C_j$  mit  $F$ .  
(Zulässige Färbung muss  $C_j$  auf  $T$  setzen)

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (4)

- Erzeuge alle Teilgraphen  $G_j$  für  $K_1, \dots, K_m$
- Verbinde  $l_{i,j}$  mit  $x_k$  wenn  $L_{i,j} = x_k$  und mit  $\neg x_k$  wenn  $L_{i,j} = \neg x_k$
- Verbinde jeweils  $C_j$  mit  $N$  und  $C_j$  mit  $F$ .  
(Zulässige Färbung muss  $C_j$  auf  $T$  setzen)

Korrektheit:

- Sei Graph 3-färbbar
- Dann sind alle  $C_j$  mit  $T$  gefärbt
- $x_i$  und  $\neg x_i$  sind mit  $T$  und  $F$  oder umgekehrt gefärbt
- Belegung  $I(x_i) = 1$  wenn  $x_i$  mit  $T$  und  $I(x_i) = 0$  sonst
- $I$  macht Formel  $F$  wahr, da in jeder Klausel ein Literal durch  $I$  wahr gemacht wird.

## $\mathcal{NP}$ -Vollständigkeit von GRAPH-COLORING (5)

---

- Analog kann aus erfüllendem  $I$  eine 3-Färbung erzeugt werden.
- Berechnung des Graph geht in Polynomialzeit.
- Daher  $3\text{-CNF-SAT} \leq_p \text{GRAPH-COLORING}$ .
- Da  $3\text{-CNF-SAT}$   $\mathcal{NP}$ -hart, folgt:

GRAPH-COLORING ist  $\mathcal{NP}$ -hart

# Zusammenfassung: Polynomielle Reduktionen

