

## 13b

 $\mathcal{NP}$ -Vollständigkeit von  
GRAPH-COLORING, (UN)DIRECTED-HAMILTON-CYCLE und  
TRAVELING-SALESPERSON

Prof. Dr. Jasmin Blanchette

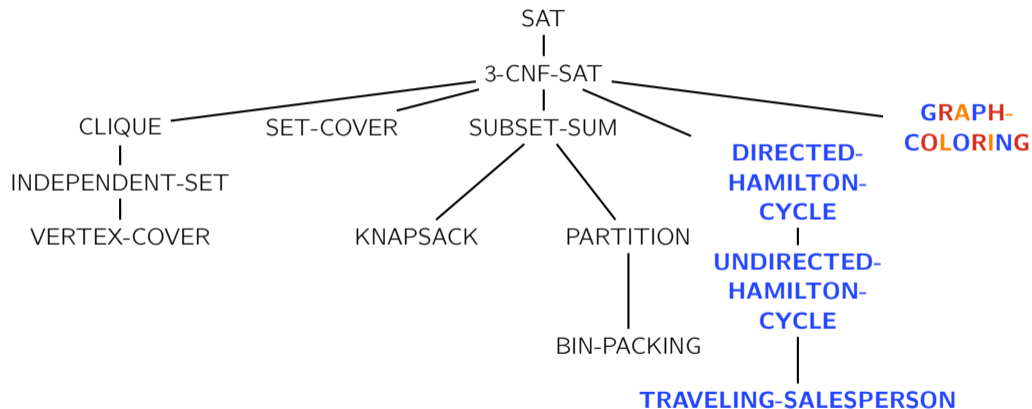
Lehr- und Forschungseinheit für  
Theoretische Informatik und Theorembeweisen

Stand: 10. Mai 2024

Basiert auf Folien von PD Dr. David Sabel



# Überblick über $\mathcal{NP}$ -Vollständigkeitsbeweise



## Definition

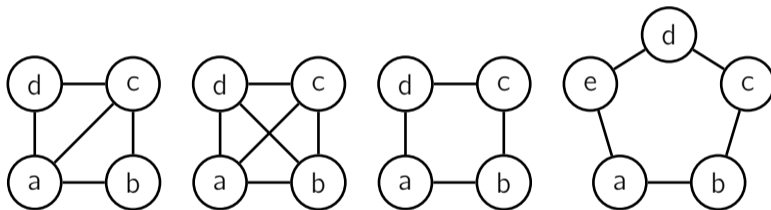
Das **GRAPH-COLORING-Problem** lässt sich wie folgt formulieren.

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

gefragt: Gibt es eine Färbung der Knoten in  $V$  mit höchstens  $k$  Farben (eine sogenannte  $k$ -Färbung), sodass keine zwei benachbarten Knoten in  $G$  die gleiche Farbe erhalten?

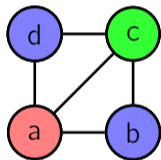
# Beispiele für Färbungen

Wie viele Farben sind jeweils minimal notwendig?

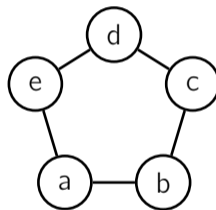
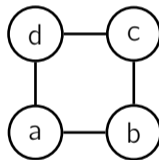
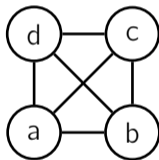


# Beispiele für Färbungen

Wie viele Farben sind jeweils minimal notwendig?

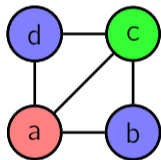


3

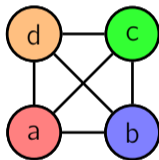


# Beispiele für Färbungen

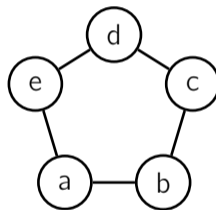
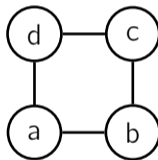
Wie viele Farben sind jeweils minimal notwendig?



3

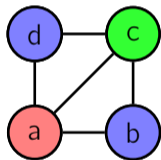


4

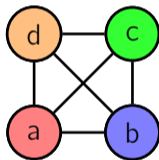


# Beispiele für Färbungen

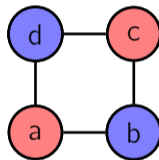
Wie viele Farben sind jeweils minimal notwendig?



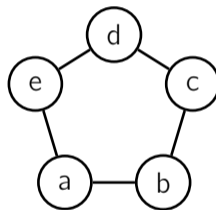
3



4

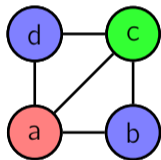


2

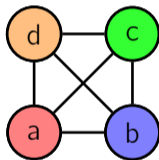


# Beispiele für Färbungen

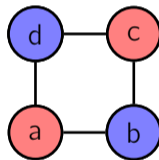
Wie viele Farben sind jeweils minimal notwendig?



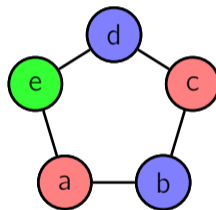
3



4



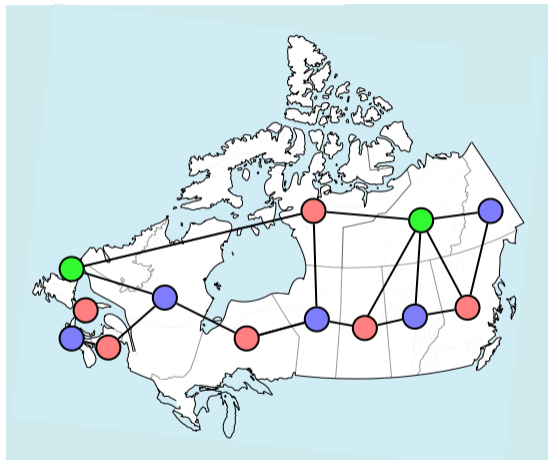
2



3



## Weiteres Beispiel für eine Färbung



Adanak

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

---

## Satz

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

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

## Satz

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

## Beweis

### 1. 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 Polynomialzeit.

Daher kann GRAPH-COLORING auf einer NTM in polynomieller Zeit entschieden werden.

**Beweis** (Fortsetzung)

2. GRAPH-COLORING ist  $\mathcal{NP}$ -schwer:

Wir zeigen  $3\text{-CNF-SAT} \leq_p \text{GRAPH-COLORING}$ .

## **Beweis** (Fortsetzung)

### 2. GRAPH-COLORING ist $\mathcal{NP}$ -schwer:

Wir zeigen  $3\text{-CNF-SAT} \leq_p \text{GRAPH-COLORING}$ .

Sei  $H = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, sodass jede Klausel  $K_j$  genau drei Literale enthält.

## **Beweis** (Fortsetzung)

### 2. GRAPH-COLORING ist $\mathcal{NP}$ -schwer:

Wir zeigen  $3\text{-CNF-SAT} \leq_p \text{GRAPH-COLORING}$ .

Sei  $H = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, sodass jede Klausel  $K_j$  genau drei Literale enthält.

Seien  $x_1, \dots, x_n$  die in  $H$  vorkommenden aussagenlogischen Variablen.

## **Beweis** (Fortsetzung)

### 2. GRAPH-COLORING ist $\mathcal{NP}$ -schwer:

Wir zeigen  $3\text{-CNF-SAT} \leq_p \text{GRAPH-COLORING}$ .

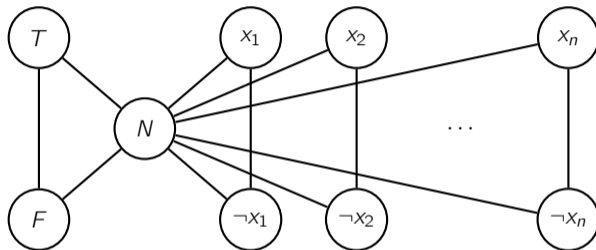
Sei  $H = K_1 \wedge \dots \wedge K_m$  eine 3-CNF, sodass jede Klausel  $K_j$  genau drei Literale enthält.

Seien  $x_1, \dots, x_n$  die in  $H$  vorkommenden aussagenlogischen Variablen.

Wir erzeugen ein GRAPH-COLORING-Problem  $f(H) = (G, k)$  mit  $k = 3$ .

# Konstruktion der GRAPH-COLORING-Instanz

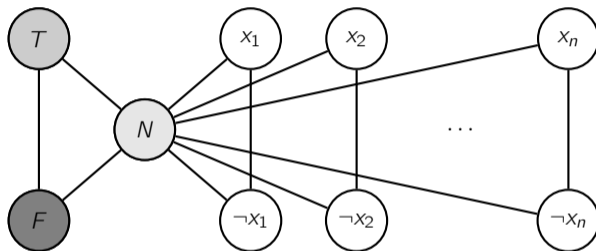
Konstruiere zunächst den folgenden Teilgraphen (die „Palette“):





# Konstruktion der GRAPH-COLORING-Instanz

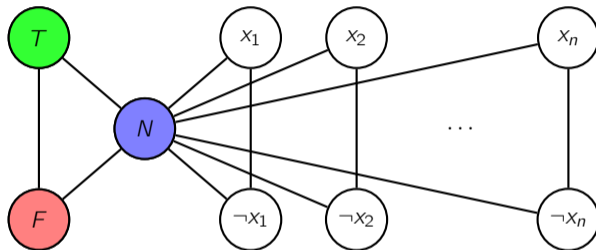
Konstruiere zunächst den folgenden Teilgraphen (die „Palette“):



Wenn der Graph 3-färbbar ist, dann müssen  $T, N, F$  verschiedene Farben haben.

# Konstruktion der GRAPH-COLORING-Instanz

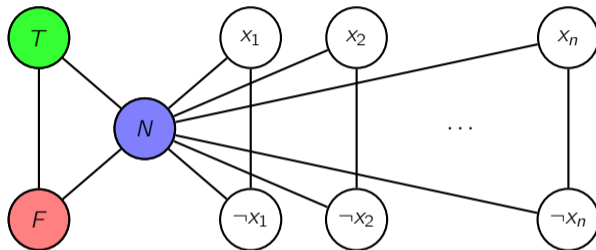
Konstruiere zunächst den folgenden Teilgraphen (die „Palette“):



Wenn der Graph 3-färbbar ist, dann müssen  $T, N, F$  verschiedene Farben haben. O.B.d.A. seien dies genau die Farben  $T, N, F$  mit genau den Knoten.

## Konstruktion der GRAPH-COLORING-Instanz

Konstruiere zunächst den folgenden Teilgraphen (die „Palette“):

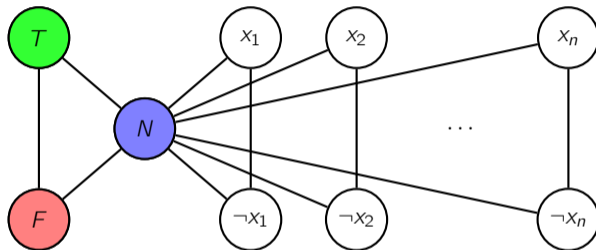


Wenn der Graph 3-färbbar ist, dann müssen  $T, N, F$  verschiedene Farben haben. O.B.d.A. seien dies genau die Farben  $T, N, F$  mit genau den Knoten.

Für 3-Färbung muss  $(x_i, \neg x_i)$  mit  $(T, F)$  oder  $(F, T)$  gefärbt werden.

# Konstruktion der GRAPH-COLORING-Instanz

Konstruiere zunächst den folgenden Teilgraphen (die „Palette“):



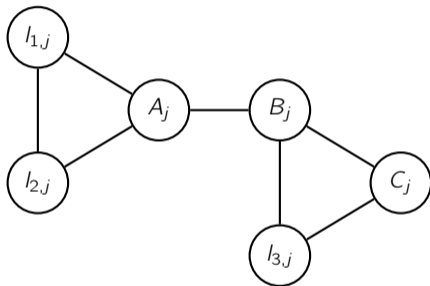
Wenn der Graph 3-färbbar ist, dann müssen  $T$ ,  $N$ ,  $F$  verschiedene Farben haben. O.B.d.A. seien dies genau die Farben  $T$ ,  $N$ ,  $F$  mit genau den Knoten.

Für 3-Färbung muss  $(x_i, \neg x_i)$  mit  $(T, F)$  oder  $(F, T)$  gefärbt werden.

Die Palette gibt eine Belegung an: Die 3-Färbung erzeugt  $I(x_i) = 1$ , wenn  $x_i$  mit  $T$  gefärbt ist, und  $I(x_i) = 0$ , wenn  $x_i$  mit  $F$  gefärbt ist.

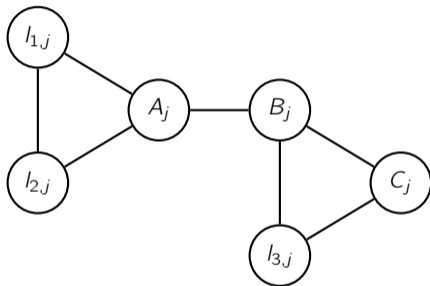
## Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



## Konstruktion der GRAPH-COLORING-Instanz

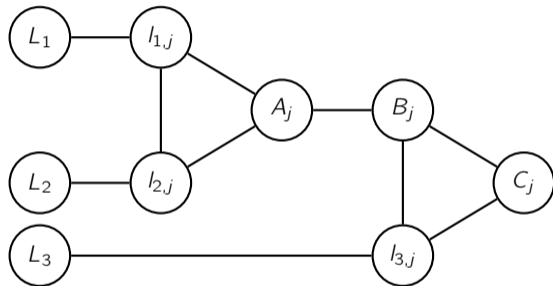
Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

## Konstruktion der GRAPH-COLORING-Instanz

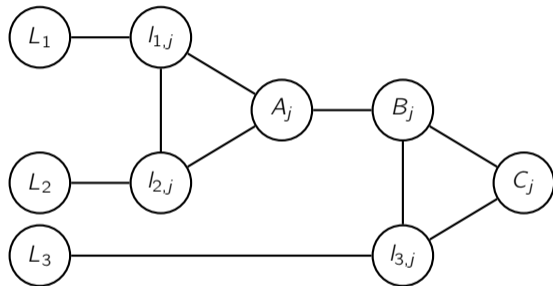
Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“. Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette. Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

## Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

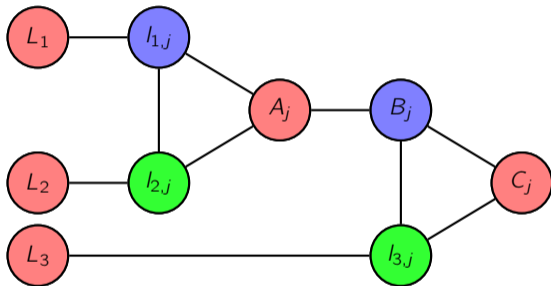
Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.



# Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

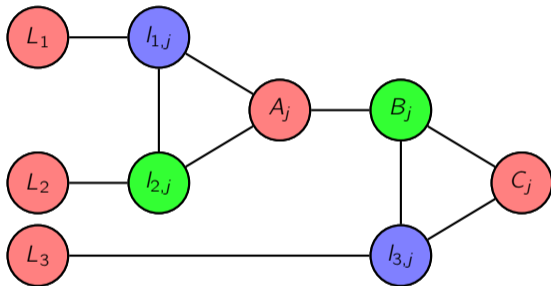
Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.

# Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

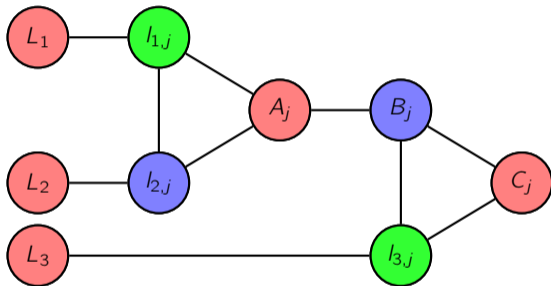
Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.

## Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

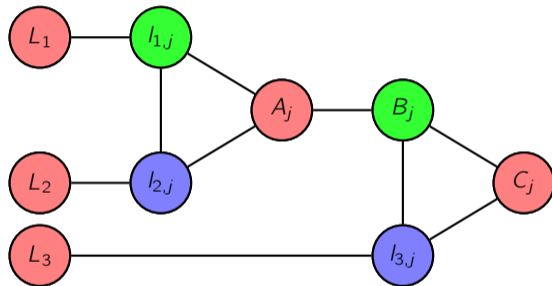
Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.

# Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

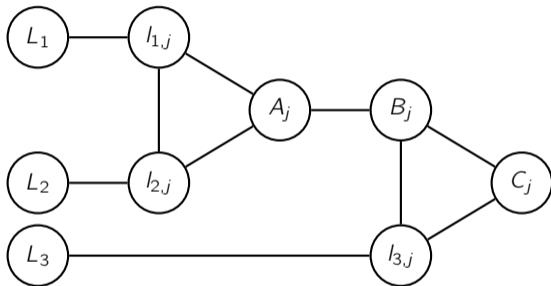
Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.

## Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

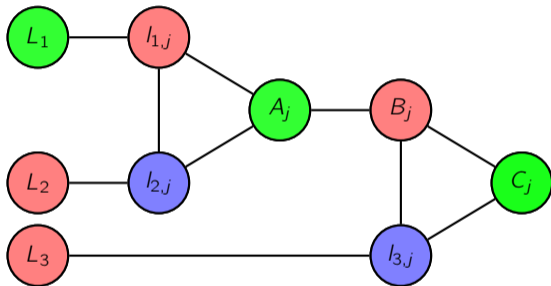
Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.

# Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

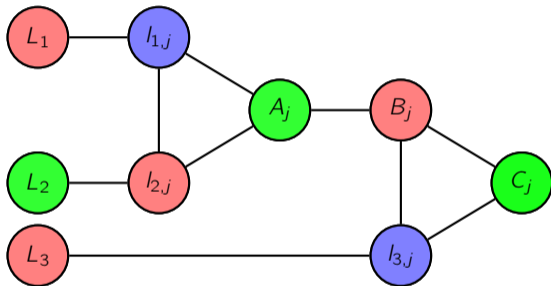
Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.

# Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

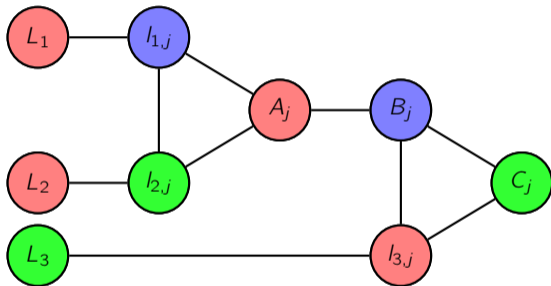
Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.

# Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

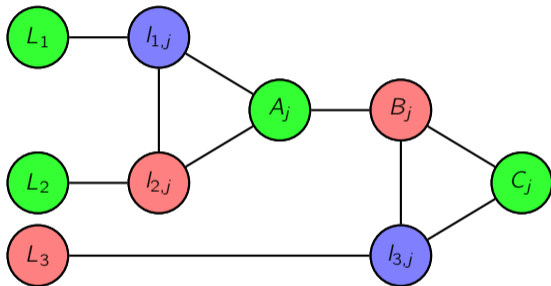
Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.



# Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

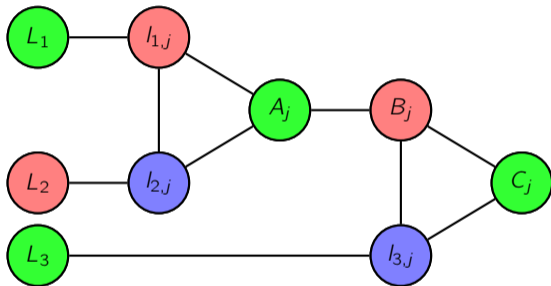
Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.

# Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

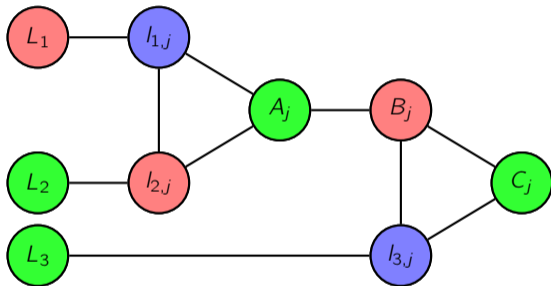
Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.

# Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

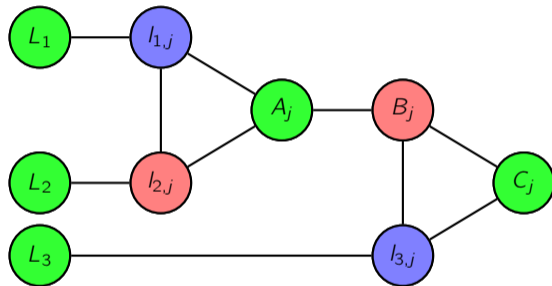
Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.

# Konstruktion der GRAPH-COLORING-Instanz

Erzeuge für jede Klausel  $K_j = L_1 \vee L_2 \vee L_3$  den folgenden Teilgraphen  $O_j$  (ein „Oder-Gatter“):



Grundgedanke:  $O_j$  soll das logische Oder der drei Literale in  $K_j$  „berechnen“.

Verbinde die Eingänge mit  $L_1$ ,  $L_2$  und  $L_3$  aus der Palette.

Die Farben der  $L_i$  sind daher  $F$  oder  $T$ .

$C_j$  kann mit  $T$  gefärbt werden g.d.w. ein  $L_i$  mit  $T$  gefärbt ist.

# Konstruktion der GRAPH-COLORING-Instanz

---

Erzeuge die Oder-Gatter  $O_1, \dots, O_m$  für jeweils  $K_1, \dots, K_m$ .

# Konstruktion der GRAPH-COLORING-Instanz

---

Erzeuge die Oder-Gatter  $O_1, \dots, O_m$  für jeweils  $K_1, \dots, K_m$ .

Für jedes  $j \in \{1, \dots, m\}$ , sei  $K_j = L_1 \vee L_2 \vee L_3$ .

Verbinde jedes  $l_{i,j}$  von  $O_j$  mit  $L_i$  aus der „Palette“.

# Konstruktion der GRAPH-COLORING-Instanz

---

Erzeuge die Oder-Gatter  $O_1, \dots, O_m$  für jeweils  $K_1, \dots, K_m$ .

Für jedes  $j \in \{1, \dots, m\}$ , sei  $K_j = L_1 \vee L_2 \vee L_3$ .

Verbinde jedes  $l_{i,j}$  von  $O_j$  mit  $L_i$  aus der „Palette“.

Verbinde jeweils  $C_j$  mit  $N$  und  $C_j$  mit  $F$ .

Eine zulässige Färbung muss daher  $C_j$  auf  $T$  setzen.

# Konstruktion der GRAPH-COLORING-Instanz

---

Erzeuge die Oder-Gatter  $O_1, \dots, O_m$  für jeweils  $K_1, \dots, K_m$ .

Für jedes  $j \in \{1, \dots, m\}$ , sei  $K_j = L_1 \vee L_2 \vee L_3$ .

Verbinde jedes  $l_{i,j}$  von  $O_j$  mit  $L_i$  aus der „Palette“.

Verbinde jeweils  $C_j$  mit  $N$  und  $C_j$  mit  $F$ .

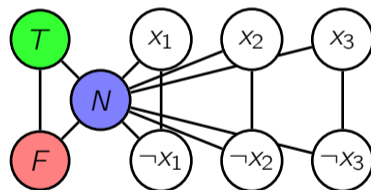
Eine zulässige Färbung muss daher  $C_j$  auf  $T$  setzen.

Damit haben wir den Graphen  $G$  konstruiert.



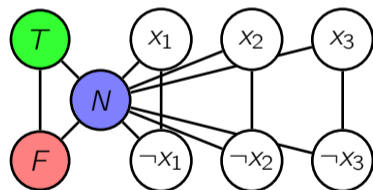
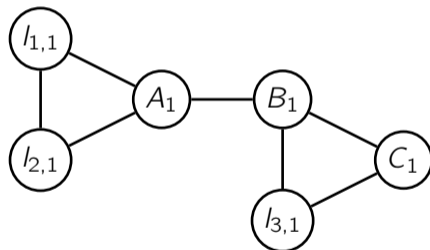
# Konstruktion der GRAPH-COLORING-Instanz

Beispiel: Sei  $H = K_1$ , wobei  $K_1 = x_1 \vee \neg x_3 \vee \neg x_2$ .



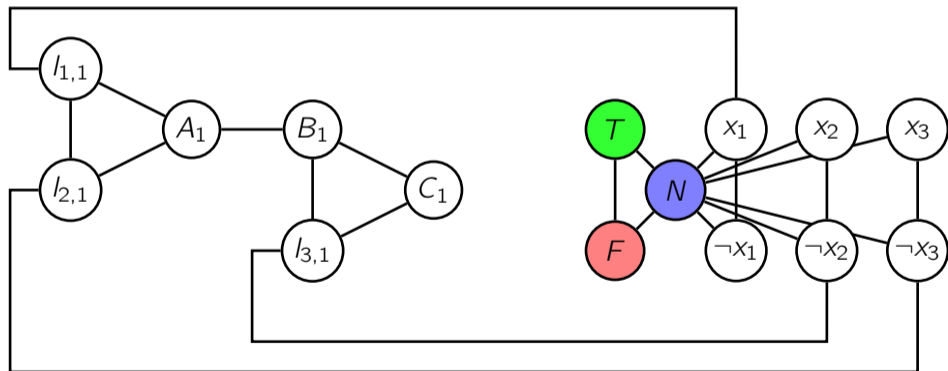
# Konstruktion der GRAPH-COLORING-Instanz

Beispiel: Sei  $H = K_1$ , wobei  $K_1 = x_1 \vee \neg x_3 \vee \neg x_2$ .



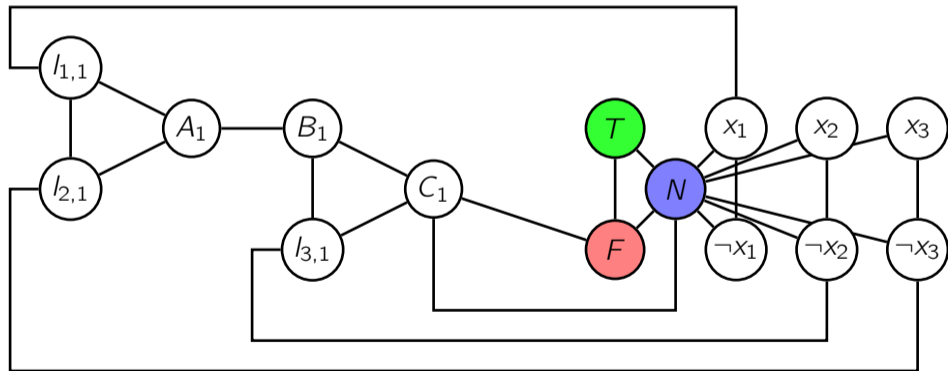
# Konstruktion der GRAPH-COLORING-Instanz

Beispiel: Sei  $H = K_1$ , wobei  $K_1 = x_1 \vee \neg x_3 \vee \neg x_2$ .



# Konstruktion der GRAPH-COLORING-Instanz

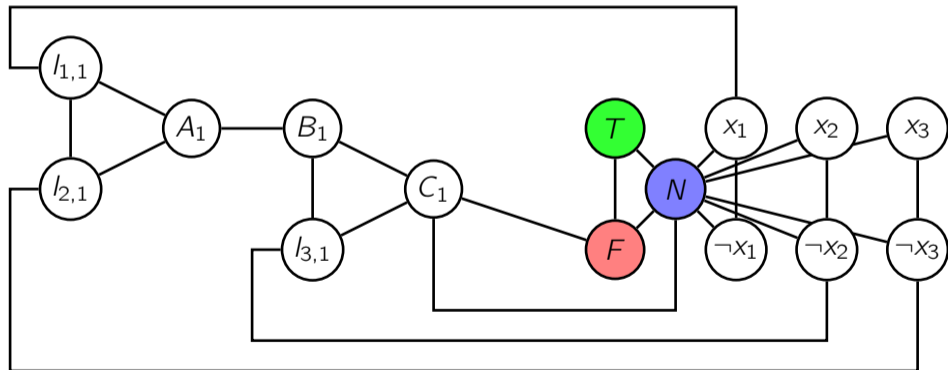
Beispiel: Sei  $H = K_1$ , wobei  $K_1 = x_1 \vee \neg x_3 \vee \neg x_2$ .



# Konstruktion der GRAPH-COLORING-Instanz

Beispiel: Sei  $H = K_1$ , wobei  $K_1 = x_1 \vee \neg x_3 \vee \neg x_2$ .

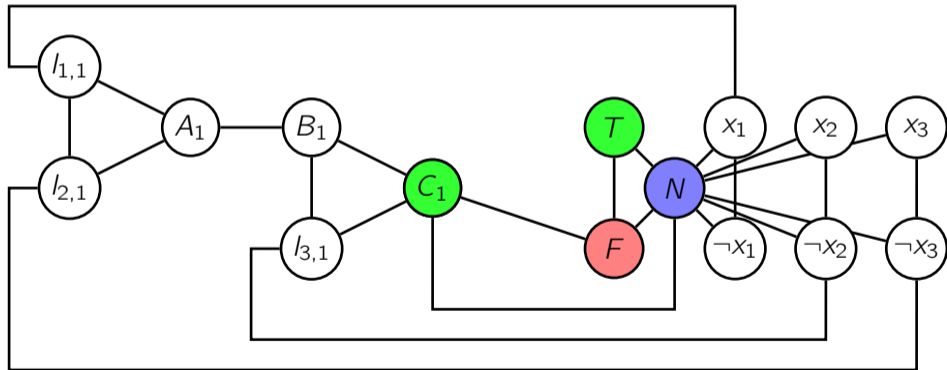
$H$  ist erfüllbar g.d.w. folgender Graph  $G$  3-färbbar ist.



# Konstruktion der GRAPH-COLORING-Instanz

Beispiel: Sei  $H = K_1$ , wobei  $K_1 = x_1 \vee \neg x_3 \vee \neg x_2$ .

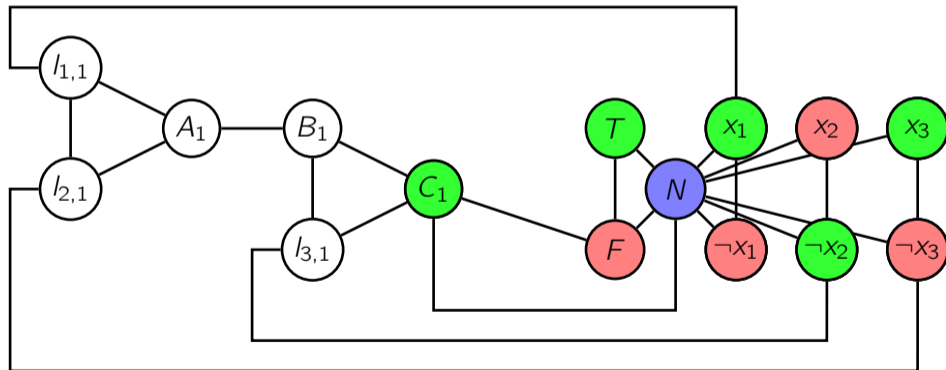
$H$  ist erfüllbar g.d.w. folgender Graph  $G$  3-färbbar ist.



# Konstruktion der GRAPH-COLORING-Instanz

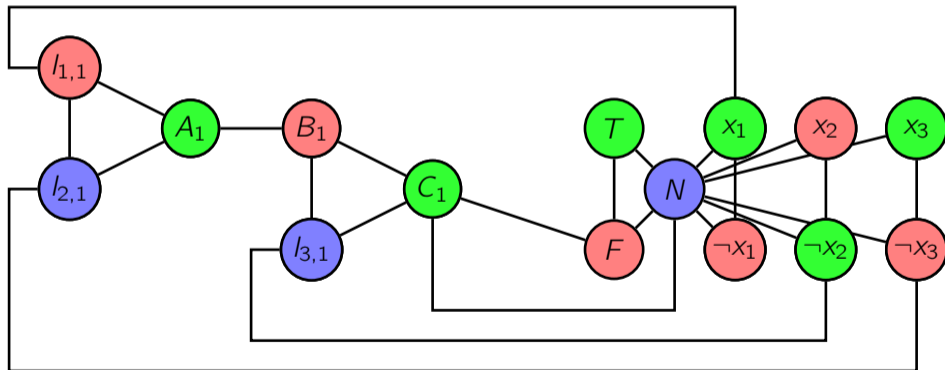
Beispiel: Sei  $H = K_1$ , wobei  $K_1 = x_1 \vee \neg x_3 \vee \neg x_2$ .

$H$  ist erfüllbar g.d.w. folgender Graph  $G$  3-färbbar ist.



# Konstruktion der GRAPH-COLORING-Instanz

Beispiel: Sei  $H = K_1$ , wobei  $K_1 = x_1 \vee \neg x_3 \vee \neg x_2$ .  
 $H$  ist erfüllbar g.d.w. folgender Graph  $G$  3-färbbar ist.





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

---

Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

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

---

Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

Wir zeigen:  $H$  ist erfüllbar g.d.w. der Graph  $G$  3-färbbar ist.

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

---

Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

Wir zeigen:  $H$  ist erfüllbar g.d.w. der Graph  $G$  3-färbbar ist.

$\Leftarrow$  Sei  $G$  3-färbbar.

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

---

Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

Wir zeigen:  $H$  ist erfüllbar g.d.w. der Graph  $G$  3-färbbar ist.

$\Leftarrow$  Sei  $G$  3-färbbar.

Dann sind alle  $C_j$  mit  $T$  gefärbt.

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

---

Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

Wir zeigen:  $H$  ist erfüllbar g.d.w. der Graph  $G$  3-färbbar ist.

← Sei  $G$  3-färbbar.

Dann sind alle  $C_j$  mit  $T$  gefärbt.

$x_j$  und  $\neg x_j$  sind mit  $T$  und  $F$  oder umgekehrt gefärbt.

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

---

Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

Wir zeigen:  $H$  ist erfüllbar g.d.w. der Graph  $G$  3-färbbar ist.

← Sei  $G$  3-färbbar.

Dann sind alle  $C_j$  mit  $T$  gefärbt.

$x_j$  und  $\neg x_j$  sind mit  $T$  und  $F$  oder umgekehrt gefärbt.

Wir definieren die Belegung  $I$  mit  $I(x_j) = 1$  wenn  $x_j$  mit  $T$  und  $I(x_j) = 0$  sonst.

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

---

Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

Wir zeigen:  $H$  ist erfüllbar g.d.w. der Graph  $G$  3-färbbar ist.

← Sei  $G$  3-färbbar.

Dann sind alle  $C_j$  mit  $T$  gefärbt.

$x_i$  und  $\neg x_i$  sind mit  $T$  und  $F$  oder umgekehrt gefärbt.

Wir definieren die Belegung  $I$  mit  $I(x_i) = 1$  wenn  $x_i$  mit  $T$  und  $I(x_i) = 0$  sonst.

Die Belegung  $I$  macht die Formel  $H$  wahr, da in jeder Klausel ein Literal durch  $I$  wahr gemacht wird.

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

---

Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

Wir zeigen:  $H$  ist erfüllbar g.d.w. der Graph  $G$  3-färbbar ist.

$\Leftarrow$  Sei  $G$  3-färbbar.

Dann sind alle  $C_j$  mit  $T$  gefärbt.

$x_i$  und  $\neg x_i$  sind mit  $T$  und  $F$  oder umgekehrt gefärbt.

Wir definieren die Belegung  $I$  mit  $I(x_i) = 1$  wenn  $x_i$  mit  $T$  und  $I(x_i) = 0$  sonst.

Die Belegung  $I$  macht die Formel  $H$  wahr, da in jeder Klausel ein Literal durch  $I$  wahr gemacht wird.

$\Rightarrow$  Analog kann aus einer erfüllenden Belegung  $I$  eine 3-Färbung erzeugt werden. □



## 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  $(v_{\pi(n)}, v_{\pi(1)}) \in E$ .

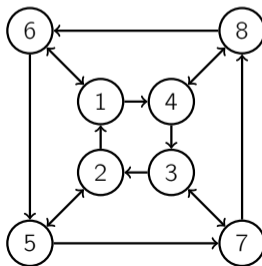
# Gerichtete Hamilton-Kreise

## 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:



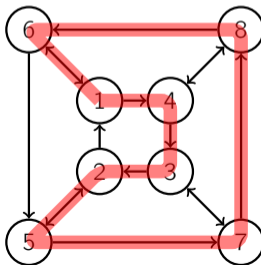
# Gerichtete Hamilton-Kreise

## 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:



# Das DIRECTED-HAMILTON-CYCLE-Problem

---

## Definition

Das **DIRECTED-HAMILTON-CYCLE-Problem** lässt sich wie folgt formulieren.

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

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

# Das DIRECTED-HAMILTON-CYCLE-Problem

---

## Definition

Das **DIRECTED-HAMILTON-CYCLE-Problem** lässt sich 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.

# Das DIRECTED-HAMILTON-CYCLE-Problem

## Definition

Das **DIRECTED-HAMILTON-CYCLE-Problem** lässt sich 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** Siehe Skript.

# Ungerichtete Hamilton-Kreise

## Definition

In einem ungerichteten 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  $\{v_{\pi(n)}, v_{\pi(1)}\} \in E$ .

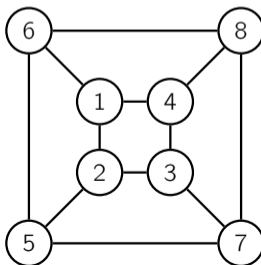
# Ungerichtete Hamilton-Kreise

## Definition

In einem ungerichteten 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

Das **UNDIRECTED-HAMILTON-CYCLE-Problem** lässt sich wie folgt formulieren.

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

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

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

## Satz

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

## Beweis

### 1. UNDIRECTED-HAMILTON-CYCLE $\in \mathcal{NP}$ :

Rate die Permutation  $\pi$  nichtdeterministisch.

Prüfe, ob  $\{v_{\pi(i)}, v_{\pi(i+1)}\} \in E$  für jedes  $i$  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

---

**Beweis** (Fortsetzung)

2. UNDIRECTED-HAMILTON-CYCLE ist  $\mathcal{NP}$ -schwer:

Wir zeigen DIRECTED-HAMILTON-CYCLE  $\leq_p$   
UNDIRECTED-HAMILTON-CYCLE.

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

**Beweis** (Fortsetzung)

2. UNDIRECTED-HAMILTON-CYCLE ist  $\mathcal{NP}$ -schwer:

Wir zeigen DIRECTED-HAMILTON-CYCLE  $\leq_p$   
UNDIRECTED-HAMILTON-CYCLE.

Sei  $f$  die Funktion, die aus einem gerichteten Graphen  $(V, E)$  einen ungerichteten Graphen macht, sodass

$$f(V) := \bigcup \{ \{v_{in}, v, v_{out}\} \mid v \in V \}$$

$$f(E) := \{ \{u_{out}, v_{in}\} \mid (u, v) \in E \} \cup \bigcup \{ \{ \{v_{in}, v\}, \{v, v_{out}\} \} \mid v \in V \}$$

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

**Beweis** (Fortsetzung)

2. UNDIRECTED-HAMILTON-CYCLE ist  $\mathcal{NP}$ -schwer:

Wir zeigen DIRECTED-HAMILTON-CYCLE  $\leq_p$   
UNDIRECTED-HAMILTON-CYCLE.

Sei  $f$  die Funktion, die aus einem gerichteten Graphen  $(V, E)$  einen ungerichteten Graphen macht, sodass

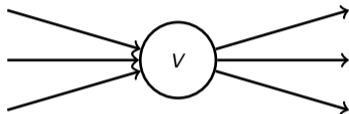
$$f(V) := \bigcup \{ \{v_{in}, v, v_{out}\} \mid v \in V \}$$

$$f(E) := \{ \{u_{out}, v_{in}\} \mid (u, v) \in E \} \cup \bigcup \{ \{ \{v_{in}, v\}, \{v, v_{out}\} \} \mid v \in V \}$$

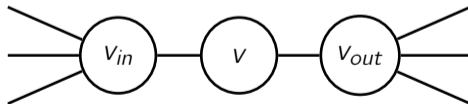
Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

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

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



wird ersetzt durch



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

---

Wir zeigen:  $G$  hat einen gerichteten Hamilton-Kreis g.d.w.  
 $f(G)$  einen ungerichteten Hamilton-Kreis hat.



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

Wir zeigen:  $G$  hat einen gerichteten Hamilton-Kreis g.d.w.

$f(G)$  einen ungerichteten Hamilton-Kreis hat.

$\implies$  Falls  $v_{\pi(n)}, \dots, v_{\pi(1)}, v_{\pi(n)}$  ein gerichteter Hamilton-Kreis ist,  
dann ist  $v_{in,\pi(1)}, v_{\pi(1)}, v_{out,\pi(1)}, \dots, v_{in,\pi(n)}, v_{\pi(n)}, v_{out,\pi(n)}, v_{in,\pi(1)}$   
ein ungerichteter Hamilton-Kreis.

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

Wir zeigen:  $G$  hat einen gerichteten Hamilton-Kreis g.d.w.  
 $f(G)$  einen ungerichteten Hamilton-Kreis hat.

$\implies$  Falls  $v_{\pi(n)}, \dots, v_{\pi(1)}, v_{\pi(n)}$  ein gerichteter Hamilton-Kreis ist,  
dann ist  $v_{in,\pi(1)}, v_{\pi(1)}, v_{out,\pi(1)}, \dots, v_{in,\pi(n)}, v_{\pi(n)}, v_{out,\pi(n)}, v_{in,\pi(1)}$   
ein ungerichteter Hamilton-Kreis.

$\impliedby$  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.

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

Wir zeigen:  $G$  hat einen gerichteten Hamilton-Kreis g.d.w.  
 $f(G)$  einen ungerichteten Hamilton-Kreis hat.

- ⇒ Falls  $v_{\pi(n)}, \dots, v_{\pi(1)}, v_{\pi(n)}$  ein gerichteter Hamilton-Kreis ist,  
dann ist  $v_{in,\pi(1)}, v_{\pi(1)}, v_{out,\pi(1)}, \dots, v_{in,\pi(n)}, v_{\pi(n)}, v_{out,\pi(n)}, v_{in,\pi(1)}$   
ein ungerichteter Hamilton-Kreis.
- ⇐ 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  $v_{in,\pi(1)}, v_{\pi(1)}, v_{out,\pi(1)}, \dots, v_{in,\pi(n)}, v_{\pi(n)}, v_{out,\pi(n)}, v_{in,\pi(1)}$   
ein ungerichteter Hamilton-Kreis ist,  
dann ist  $v_{\pi(1)}, \dots, v_{\pi(n)}, v_{\pi(1)}$  ein gerichteter Hamilton-Kreis.

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

Wir zeigen:  $G$  hat einen gerichteten Hamilton-Kreis g.d.w.  
 $f(G)$  einen ungerichteten Hamilton-Kreis hat.

$\implies$  Falls  $v_{\pi(n)}, \dots, v_{\pi(1)}, v_{\pi(n)}$  ein gerichteter Hamilton-Kreis ist,  
dann ist  $v_{in,\pi(1)}, v_{\pi(1)}, v_{out,\pi(1)}, \dots, v_{in,\pi(n)}, v_{\pi(n)}, v_{out,\pi(n)}, v_{in,\pi(1)}$   
ein ungerichteter Hamilton-Kreis.

$\impliedby$  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  $v_{in,\pi(1)}, v_{\pi(1)}, v_{out,\pi(1)}, \dots, v_{in,\pi(n)}, v_{\pi(n)}, v_{out,\pi(n)}, v_{in,\pi(1)}$   
ein ungerichteter Hamilton-Kreis ist,  
dann ist  $v_{\pi(1)}, \dots, v_{\pi(n)}, v_{\pi(1)}$  ein gerichteter Hamilton-Kreis.
- ▶ Falls  $v_{out,\pi(1)}, v_{\pi(1)}, v_{in,\pi(1)}, \dots, v_{out,\pi(n)}, v_{\pi(n)}, v_{in,\pi(n)}, v_{out,\pi(1)}$   
ein ungerichteter Hamilton-Kreis ist,  
dann ist  $v_{\pi(n)}, \dots, v_{\pi(1)}, v_{\pi(n)}$  ein gerichteter Hamilton-Kreis.

□

# Das TRAVELING-SALESPERSON-Problem

## Definition

Das **TRAVELING-SALESPERSON-Problem** lässt sich wie folgt formulieren.

gegeben: eine 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  
 $(\sum_{i=1}^{n-1} M_{\pi(i), \pi(i+1)}) + M_{\pi(n), \pi(1)} \leq k$ ?

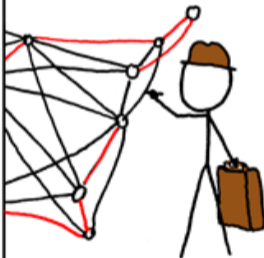
BRUTE-FORCE  
SOLUTION:

$$O(n!)$$



DYNAMIC  
PROGRAMMING  
ALGORITHMS:

$$O(n^2 2^n)$$



SELLING ON EBAY:  
 $O(1)$

STILL WORKING  
ON YOUR ROUTE?

SHUT THE  
HELL UP.



[xkcd.com/399/](http://xkcd.com/399/)

# $\mathcal{NP}$ -Vollständigkeit von TRAVELING-SALESPERSON

---

## Satz

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

# $\mathcal{NP}$ -Vollständigkeit von TRAVELING-SALESPERSON

## Satz

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

## Beweis

► TRAVELING-SALESPERSON  $\in \mathcal{NP}$ :

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 TRAVELING-SALESPERSON

---

**Beweis** (Fortsetzung)

- ▶ TRAVELING-SALESPERSON ist  $\mathcal{NP}$ -schwer:

Wir zeigen  $\text{UNDIRECTED-HAMILTON-CYCLE} \leq_p$   
TRAVELING-SALESPERSON.

# $\mathcal{NP}$ -Vollständigkeit von TRAVELING-SALESPERSON

**Beweis** (Fortsetzung)

- ▶ TRAVELING-SALESPERSON ist  $\mathcal{NP}$ -schwer:

Wir zeigen  $\text{UNDIRECTED-HAMILTON-CYCLE} \leq_p$   
TRAVELING-SALESPERSON.

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

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

# $\mathcal{NP}$ -Vollständigkeit von TRAVELING-SALESPERSON

**Beweis** (Fortsetzung)

- ▶ TRAVELING-SALESPERSON ist  $\mathcal{NP}$ -schwer:

Wir zeigen  $\text{UNDIRECTED-HAMILTON-CYCLE} \leq_p$   
TRAVELING-SALESPERSON.

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

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

Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

# $\mathcal{NP}$ -Vollständigkeit von TRAVELING-SALESPERSON

**Beweis** (Fortsetzung)

- ▶ TRAVELING-SALESPERSON ist  $\mathcal{NP}$ -schwer:

Wir zeigen  $\text{UNDIRECTED-HAMILTON-CYCLE} \leq_p$   
TRAVELING-SALESPERSON.

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

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

Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

Wir zeigen:  $G$  hat einen Hamilton-Kreis g.d.w.  
 $f(G)$  eine Rundreise der Länge  $n$  hat.

# $\mathcal{NP}$ -Vollständigkeit von TRAVELING-SALESPERSON

**Beweis** (Fortsetzung)

- ▶ TRAVELING-SALESPERSON ist  $\mathcal{NP}$ -schwer:

Wir zeigen  $\text{UNDIRECTED-HAMILTON-CYCLE} \leq_p$   
TRAVELING-SALESPERSON.

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

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

Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

Wir zeigen:  $G$  hat einen Hamilton-Kreis g.d.w.

$f(G)$  eine Rundreise der Länge  $n$  hat.

$\implies$  Wenn  $G$  einen Hamilton-Kreis hat, dann ist das eine Rundreise der Länge  $n$ .

# $\mathcal{NP}$ -Vollständigkeit von TRAVELING-SALESPERSON

**Beweis** (Fortsetzung)

► TRAVELING-SALESPERSON ist  $\mathcal{NP}$ -schwer:

Wir zeigen  $\text{UNDIRECTED-HAMILTON-CYCLE} \leq_p$   
TRAVELING-SALESPERSON.

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

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

Die Funktion  $f$  ist total und kann in Polynomialzeit berechnet werden.

Wir zeigen:  $G$  hat einen Hamilton-Kreis g.d.w.

$f(G)$  eine Rundreise der Länge  $n$  hat.

⇒ Wenn  $G$  einen 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 Hamilton-Kreis. □

# Zusammenfassung der $\mathcal{NP}$ -Vollständigkeitsbeweise

