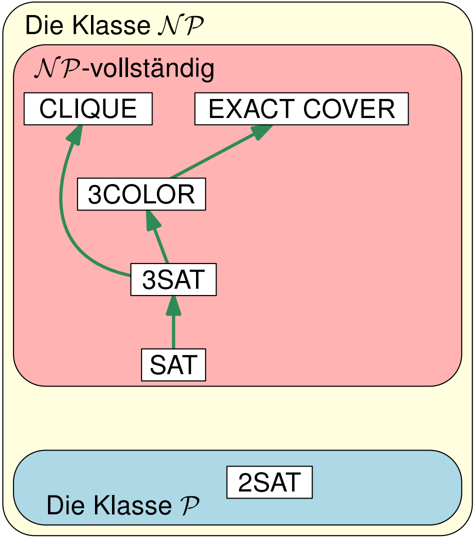
# Theoretische Grundlagen der Informatik
# Vorlesung am 04. Dezember 2018

INSTITUT FÜR THEORETISCHE INFORMATIK

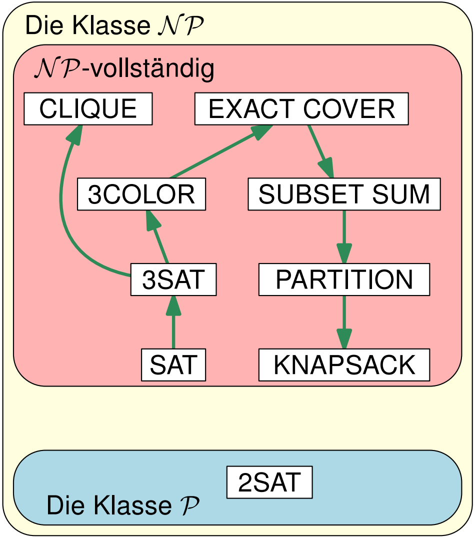
# neue Termine (Änderungen vorbehalten)

| Dienstags | | Donnerstags | |
|---|---|---|---|
| 16.10. | Vorlesung | 18.10. | Vorlesung |
| 23.10. | Vorlesung | 25.10. | Übung |
| 30.10. | Vorlesung | 01.11. | Feiertag |
| 06.11. | Vorlesung | 08.11. | Übung |
| 13.11. | Vorlesung | 15.11. | Vorlesung |
| 20.11. | Übung | 22.11. | Vorlesung |
| 27.11. | Vorlesung | 29.11. | Übung |
| 04.12. | Vorlesung | 06.12. | Übung |
| 11.12. | Vorlesung | 13.12. | Vorlesung |
| 18.12. | Übung | 20.12. | Vorlesung |
| 08.01. | Vorlesung | 10.01. | — |
| 15.01. | Übung | 17.01. | Vorlesung |
| 22.01. | Vorlesung | 24.01. | Übung |
| 29.01. | Vorlesung | 31.01. | Vorlesung |
| 05.02. | Übung | 07.02. | Vorlesung |

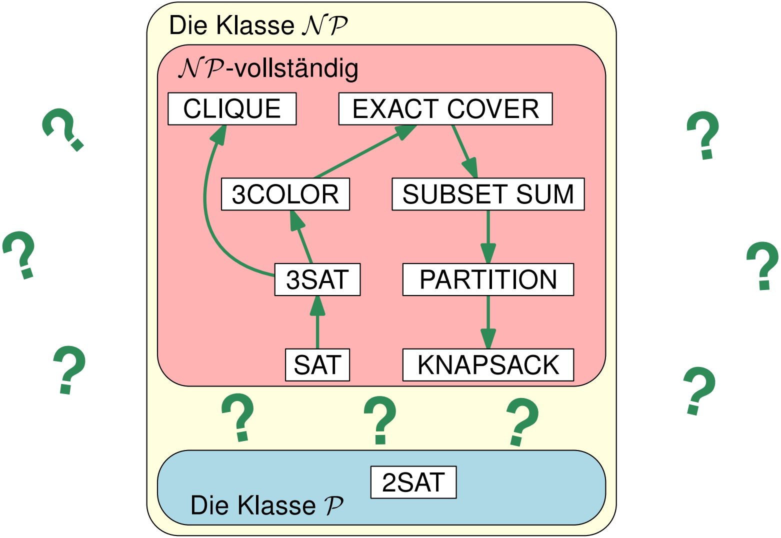
# Letzte / Diese Vorlesung



# Letzte / Diese Vorlesung



# Letzte / Diese Vorlesung



# Das Problem SUBSET SUM

**Problem SUBSET SUM**

**Gegeben:** Eine endliche Menge $M$,
eine Gewichtsfunktion $w\colon M \to \mathbb{N}_0$,
eine Zahl $K \in \mathbb{N}_0$

**Frage:** Existiert eine Teilmenge $M' \subseteq M$ mit $\sum_{a \in M'} w(a) = K$ ?

**Satz**:
Problem SUBSET SUM ist $\mathcal{NP}$-vollständig.

# Das Problem SUBSET SUM

**Problem SUBSET SUM**

**Gegeben:** Eine endliche Menge $M$,
eine Gewichtsfunktion $w\colon M \to \mathbb{N}_0$,
eine Zahl $K \in \mathbb{N}_0$

**Frage:** Existiert eine Teilmenge $M' \subseteq M$ mit $\sum_{a \in M'} w(a) = K$ ?

**Satz**:
Problem SUBSET SUM ist $\mathcal{NP}$-vollständig.

**Notation:**

$$w(X) := \sum_{a \in X} w(a)$$

# Das Problem SUBSET SUM

**Problem SUBSET SUM**

**Gegeben:** Eine endliche Menge $M$,
eine Gewichtsfunktion $w\colon M \to \mathbb{N}_0$,
eine Zahl $K \in \mathbb{N}_0$

**Frage:** Existiert eine Teilmenge $M' \subseteq M$ mit $w(M') = K$ ?

**Satz**:
Problem SUBSET SUM ist $\mathcal{NP}$-vollständig.

**Notation:**

$$w(X) := \sum_{a \in X} w(a)$$

# $\mathcal{NP}$-Vollständigkeit von SUBSET SUM

SUBSET SUM $\in \mathcal{NP}$.

- Es kann für eine gegebene Teilmenge $M' \subseteq M$ in Polynomialzeit der Wert $w(M') = \sum_{a \in M'} w(a)$ ausgerechnet und mit $K$ verglichen werden.

# Beweis: $\mathcal{NP}$-Vollständigkeit von SUBSET SUM

EXACT COVER $\propto$ SUBSET SUM

- Sei $(X = \{0, 1, \ldots, m-1\}, \mathcal{S})$ eine beliebige EXACT COVER-Instanz.
- Konstruiere SUBSET SUM Instanz $(M, w, K)$

$$
\begin{aligned}
M &:= \mathcal{S} \\
\#x &:= \#\{Y \in \mathcal{S} \mid x \in Y\} \\
p &:= \max\{\#x + 1 \mid x \in X\} \\
w(x) &:= p^x \\
K &:= w(M) = \sum_{x=0}^{m-1} p^x
\end{aligned}
$$

- Die Konstruktion benötigt nur Polynomialzeit.

# Beweis: $\mathcal{NP}$-Vollständigkeit von SUBSET SUM

$$
\begin{aligned}
M &:= \mathcal{S} \\
\#x &:= \#\{Y \in \mathcal{S} \mid x \in Y\} \\
p &:= \max\{\#x + 1 \mid x \in X\} \\
w(x) &:= p^x \\
K &:= w(M) = \sum_{x=0}^{m-1} p^x
\end{aligned}
$$

**Beispiel:**

$$
\begin{aligned}
X &= \{0, \ldots, 6\} \\
Y &= \{0, 2, 5\} \in \mathcal{S} \\
&\downarrow \\
w(Y) &= p^0 + p^2 + p^5 \\
&\downarrow \\
w(Y) &= 1010010_p \\
K &= 1111111_p
\end{aligned}
$$

## Veranschaulichung:

- Wir stellen die Mengenzugehörigkeiten als Zahlen zur Basis $p$ dar.
- Kodiere $w(Y)$ für $Y \in \mathcal{S}$ als String aus Nullen und Einsen der Länge $m$, wobei an $i$-ter Stelle eine 1 steht genau dann, wenn $i \in Y$;
- entsprechend ist $K$ ein String der Länge $m$ aus Einsen.

# Beweis: $\mathcal{NP}$-Vollständigkeit von SUBSET SUM

$$
\begin{aligned}
M &:= \mathcal{S} \\
\#x &:= \#\{Y \in \mathcal{S} \mid x \in Y\} \\
p &:= \max\{\#x + 1 \mid x \in X\} \\
w(x) &:= p^x \\
K &:= w(M) = \sum_{x=0}^{m-1} p^x
\end{aligned}
$$

**Beispiel:**

$$
\begin{aligned}
X &= \{0, \dots, 6\} \\
Y &= \{0, 2, 5\} \in \mathcal{S} \\
&\downarrow \\
w(Y) &= p^0 + p^2 + p^5 \\
&\downarrow \\
w(Y) &= 1010010_p \\
K &= 1111111_p
\end{aligned}
$$

**Veranschaulichung:**

- Komponentenweise Addition der Strings $w(Y_1), \dots, w(Y_n)$ ergibt einen String der Länge $m$, an dessen $i$-ter Stelle steht in wievielen der $Y_j$, $j = 1, \dots, n$, das Element $i$ vorkommt.
- $\sum_{Y \in \mathcal{S}'} w(Y) = K$ bedeutet also, dass jedes $x \in X$ in genau einem $Y \in \mathcal{S}'$ vorkommt.

# Beweis: $\mathcal{NP}$-Vollständigkeit von SUBSET SUM

$$
\begin{aligned}
M &:= \mathcal{S}\\
\#x &:= \#\{Y \in \mathcal{S} \mid x \in Y\}\\
p &:= \max\{\#x + 1 \mid x \in X\}\\
w(x) &:= p^x\\
K &:= w(M) = \sum_{x=0}^{m-1} p^x
\end{aligned}
$$

**Beispiel:**

$$
\begin{aligned}
X &= \{0, \ldots, 6\}\\
Y &= \{0, 2, 5\} \in \mathcal{S}\\
&\downarrow\\
w(Y) &= p^0 + p^2 + p^5\\
&\downarrow\\
w(Y) &= 1010010_p\\
K &= 1111111_p
\end{aligned}
$$

**Beispiel:**

$\mathcal{S} = \{Y_1 = \{0, 2, 5\}, Y_2 = \{0, 1, 2\}, Y_3 = \{2, 4, 5\}, Y_4 = \{3, 4, 6\}\}$

$\#0 = 2$, $\#1 = 1$, $\#2 = 3$, $\#3 = 1$, $\#4 = 2$, $\#5 = 1$, $\#6 = 1$, $p = 4$

$w(Y_1) = 1010010_4$ , $w(Y_2) = 1110000_4$

$w(Y_3) = 0010110_4$, $w(Y_4) = 0001101_4$.

# Beweis: $\mathcal{NP}$-Vollständigkeit von SUBSET SUM

$$
\begin{aligned}
M &:= \mathcal{S} \\
\#x &:= \#\{Y \in \mathcal{S} \mid x \in Y\} \\
p &:= \max\{\#x + 1 \mid x \in X\} \\
w(x) &:= p^x \\
K &:= w(M) = \sum_{x=0}^{m-1} p^x
\end{aligned}
$$

**Beispiel:**

$$
\begin{aligned}
X &= \{0, \ldots, 6\} \\
Y &= \{0, 2, 5\} \in \mathcal{S} \\
&\downarrow \\
w(Y) &= p^0 + p^2 + p^5 \\
&\downarrow \\
w(Y) &= 1010010_p \\
K &= 1111111_p
\end{aligned}
$$

- $(X, \mathcal{S})$ lösbar $\Rightarrow$ $(M, w, K)$ lösbar.

Sei $\mathcal{S}' \subseteq \mathcal{S}$ exakte Überdeckung von $(X, \mathcal{S})$. Dann gilt

$$\sum_{Y \in \mathcal{S}'} w(Y) = \sum_{Y \in \mathcal{S}'} \sum_{x \in Y} p^x = \sum_{x=0}^{m-1} p^x = K,$$

da jedes $x \in X$ genau einmal überdeckt wird.
$\rightsquigarrow$ $\mathcal{S}'$ erfüllt die Bedingung für SUBSET SUM.

# Beweis: $\mathcal{NP}$-Vollständigkeit von SUBSET SUM

$$
\begin{aligned}
M &:= \mathcal{S} \\
\#x &:= \#\{Y \in \mathcal{S} \mid x \in Y\} \\
p &:= \max\{\#x + 1 \mid x \in X\} \\
w(x) &:= p^x \\
K &:= w(M) = \sum_{x=0}^{m-1} p^x
\end{aligned}
$$

**Beispiel:**

$$
\begin{aligned}
X &= \{0, \ldots, 6\} \\
Y &= \{0, 2, 5\} \in \mathcal{S} \\
&\downarrow \\
w(Y) &= p^0 + p^2 + p^5 \\
&\downarrow \\
w(Y) &= 1010010_p \\
K &= 1111111_p
\end{aligned}
$$

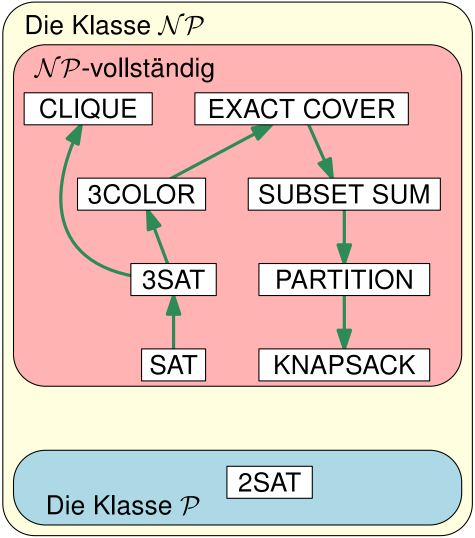
- $(X, \mathcal{S})$ lösbar $\Leftarrow (M, w, K)$ lösbar.

Ist $\mathcal{S}' \subseteq M = \mathcal{S}$ eine geeignete Menge für SUBSET SUM, so gilt

$$\sum_{Y \in \mathcal{S}'} w(Y) = K = \sum_{x=0}^{m-1} p^x.$$

Nach Wahl von $p$ kommt jedes $x \in X$ in genau einem $Y \in \mathcal{S}'$ vor.
$\rightsquigarrow \mathcal{S}'$ ist damit eine exakte Überdeckung von $(X, \mathcal{S})$.

# Diese Vorlesung



# Das Problem PARTITION

**Problem PARTITION**

**Gegeben:** Eine endliche Menge $M$,
eine Gewichtsfunktion $w\colon M \to \mathbb{N}_0$

**Frage:** Existiert eine Teilmenge $M' \subseteq M$ mit
$\sum_{a \in M'} w(a) = \sum_{a \in M \setminus M'} w(a)$, d.h. $w(M') = w(M \setminus M')$?

**Satz**:
Problem PARTITION ist $\mathcal{NP}$-vollständig.

# Beweis: $\mathcal{NP}$-Vollständigkeit von PARTITION

PARTITION $\in \mathcal{NP}$.

- Für eine Menge $M'$ können in Polynomialzeit die Werte $w(M') = \sum_{a \in M'} w(a)$ und $w(M \setminus M') = \sum_{a \in M \setminus M'} w(a)$ ausgerechnet und verglichen werden.

# Beweis: $\mathcal{NP}$-Vollständigkeit von PARTITION

SUBSET SUM $\propto$ PARTITION.

- Sei $(M, w, K)$ eine beliebige SUBSET SUM-Instanz.
- Konstruiere PARTITION-Instanz $(M^*, w^*)$

$$
\begin{aligned}
N &:= w(M) + 1 \\
M^* &:= M \cup \{b, c\} \\
w^*(a) &:= w(a) \qquad \text{für } a \in M \\
w^*(b) &:= N - K \\
w^*(c) &:= K + 1
\end{aligned}
$$

- Die Konstruktion benötigt nur Polynomialzeit.

# Beweis: $\mathcal{NP}$-Vollständigkeit von PARTITION

$$\begin{aligned} N &:= w(M) + 1 \\ M^* &:= M \cup \{b, c\} \\ w^*(a) &:= w(a) \qquad \text{für } a \in M \\ w^*(b) &:= N - K \\ w^*(c) &:= K + 1 \end{aligned}$$

**Beispiel:**

$$\begin{aligned} M &= \{i, j, k, \ell\} \\ w(j) &= 13 \\ &\downarrow \\ M^* &= \{i, j, k, \ell, b, c\} \\ w^*(j) &= 13 \end{aligned}$$

- $(M^*, w^*)$ Ja-Instanz genau dann, wenn $(M, w, K)$ Ja-Instanz:

$$\boxed{\exists M' \subseteq M^* \colon w^*(M') = w^*(M^* \setminus M')} \iff \boxed{\exists M'' \subseteq M \colon w(M'') = K}$$

- Es gilt:

$$\begin{aligned} w^*(b) + w^*(c) &= (N - K) + (K + 1) = N + 1 \\ w^*(M^*) &= (N - 1) + (N - K) + (K + 1) = 2N \end{aligned}$$

- Also können $b$ und $c$ nicht beide in $M'$ bzw. $M^* \setminus M'$ enthalten sein.
- o.B.d.A. sei also $b \in M'$ und $c \in M^* \setminus M'$

# Beweis: $\mathcal{NP}$-Vollständigkeit von PARTITION

$$\begin{aligned} N &:= w(M)+1 \\ M^* &:= M \cup \{b, c\} \\ w^*(a) &:= w(a) \qquad \text{für } a \in M \\ w^*(b) &:= N - K \\ w^*(c) &:= K + 1 \end{aligned}$$

**Beispiel:**

$$\begin{aligned} M &= \{i, j, k, \ell\} \\ w(j) &= 13 \\ &\downarrow \\ M^* &= \{i, j, k, \ell, b, c\} \\ w^*(j) &= 13 \end{aligned}$$

- $(M^*, w^*)$ Ja-Instanz genau dann, wenn $(M, w, K)$ Ja-Instanz:

$$\boxed{\exists M' \subseteq M^* \colon w^*(M') = w^*(M^* \setminus M')} \iff \boxed{\exists M'' \subseteq M \colon w(M'') = K}$$

"$\Longrightarrow$"

- Sei $M' \subseteq M^*$, so dass $w^*(M') = w^*(M^* \setminus M')$ mit $b \in M'$.
- Dann gilt $w^*(M') = N$, da $w^*(M^*) = 2N$.
- Damit erfüllt $M'' := M' \setminus \{b\}$ die Bedingung für SUBSET SUM, denn $w(M'') = w^*(M') - w^*(b) = N - (N - K) = K$.

# Beweis: $\mathcal{NP}$-Vollständigkeit von PARTITION

$$
\begin{aligned}
N &:= w(M) + 1 \\
M^* &:= M \cup \{b, c\} \\
w^*(a) &:= w(a) \qquad \text{für } a \in M \\
w^*(b) &:= N - K \\
w^*(c) &:= K + 1
\end{aligned}
$$

**Beispiel:**

$$
\begin{aligned}
M &= \{i, j, k, \ell\} \\
w(j) &= 13 \\
&\downarrow \\
M^* &= \{i, j, k, \ell, b, c\} \\
w^*(j) &= 13
\end{aligned}
$$

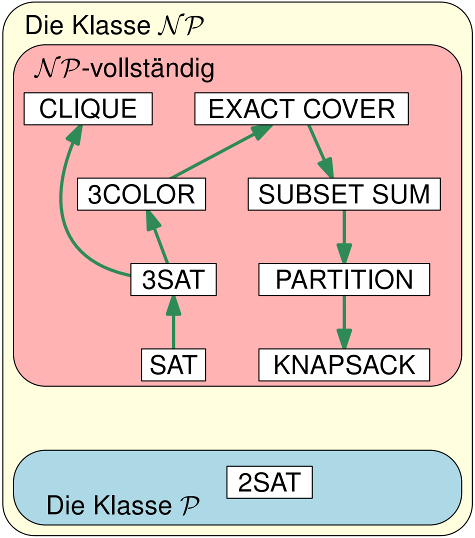
- $(M^*, w^*)$ Ja-Instanz genau dann, wenn $(M, w, K)$ Ja-Instanz:

$$\boxed{\exists M' \subseteq M^* \colon w^*(M') = w^*(M^* \setminus M')} \iff \boxed{\exists M'' \subseteq M \colon w(M'') = K}$$

“$\Longleftarrow$”

- Sei $M''$, so dass $w(M'') = K$.
- Dann erfüllt $M' := M'' \cup \{b\}$ die Bedingung für PARTITION, denn $w^*(M') = w(M') + w^*(b) = K + (N - K) = N$.

# Diese Vorlesung



# Das Problem KNAPSACK

**Problem KNAPSACK**

**Gegeben:** Eine endliche Menge $M$,
eine Gewichtsfunktion $w\colon M \to \mathbb{N}_0$,
eine Kostenfunktion $c\colon M \to \mathbb{N}_0$
Zahlen $W, C \in \mathbb{N}_0$

**Frage:** Existiert eine Teilmenge $M' \subseteq M$ mit $w(M') \leq W$ und $c(M') \geq C$?

**Satz**:
Problem KNAPSACK ist $\mathcal{NP}$-vollständig.

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

KNAPSACK $\in \mathcal{NP}$.

- Für eine Menge $M'$ kann in Polynomialzeit überprüft werden, ob
  - $w(M') = \sum_{a \in M'} w(a) \leq W$ und
  - $c(M') = \sum_{a \in M'} c(a) \geq C$

  gilt.

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

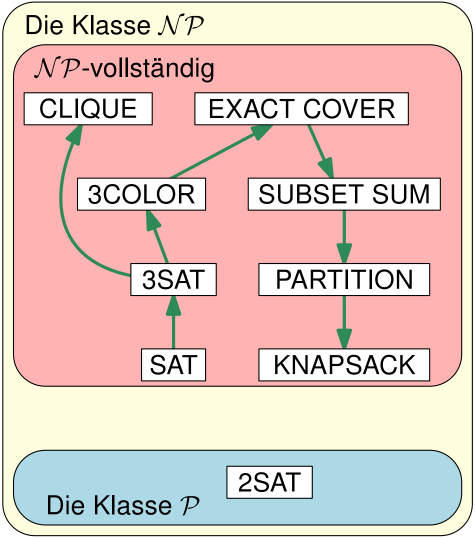
PARTITION $\propto$ KNAPSACK.

- Sei $(M, w)$ eine PARTITION-Instanz.
- Konstruiere KNAPSACK-Instanz $(M, w', c, W, C)$

$$
\begin{aligned}
w' &:= 2w \\
c &:= 2w \\
W = C &:= w(M) = \sum_{a \in M} w(a)
\end{aligned}
$$

- Die Konstruktion benötigt nur Polynomialzeit.
- Es ist $(M, w)$ genau dann eine Ja-Instanz, wenn $(M, w', c, W, C)$ eine Ja-Instanz ist (ohne Beweis).

# Diese Vorlesung



# Auswirkung auf die Frage $\mathcal{P} = \mathcal{NP}$

- Wir haben gesehen, dass es für je zwei $\mathcal{NP}$-vollständige Probleme eine polynomiale Transformation von einem zum anderen Problem gibt.
- Deshalb sind alle $\mathcal{NP}$-vollständigen Probleme im wesentlichen gleich schwer.
- Dies hat Auswirkungen auf die Frage, ob $\mathcal{P} = \mathcal{NP}$ ist.

**Satz**:
Sei $L$ $\mathcal{NP}$-vollständig, dann gilt:

- $L \in \mathcal{P} \Longrightarrow \mathcal{P} = \mathcal{NP}$
- $L \notin \mathcal{P} \Longrightarrow$ für jede $\mathcal{NP}$-vollständige Sprache $L'$ gilt $L' \notin \mathcal{P}$

# Auswirkung auf die Frage $\mathcal{P} = \mathcal{NP}$

**Satz**:
Sei $L$ $\mathcal{NP}$-vollständig, dann gilt:

- $L \in \mathcal{P} \Longrightarrow \mathcal{P} = \mathcal{NP}$
- $L \notin \mathcal{P} \Longrightarrow$ für jede $\mathcal{NP}$-vollständige Sprache $L'$ gilt $L' \notin \mathcal{P}$

**Beweis Teil 1:**

- Sei $L \in \mathcal{P}$ und $L$ $\mathcal{NP}$-vollständig.
- Dann existiert eine polynomiale deterministische TM $M$ für $L$.
- Sei $L' \in \mathcal{NP}$:
- Es gibt polynomiale Transformation $L' \propto L$.
- Hintereinanderausführung von $L' \propto L$ und $M$ liefert deterministische polynomielle TM-Berechnung für $L'$.
- Damit ist $L' \in \mathcal{P}$.

# Auswirkung auf die Frage $\mathcal{P} = \mathcal{NP}$

**Satz**:
Sei $L$ $\mathcal{NP}$-vollständig, dann gilt:

- $L \in \mathcal{P} \Longrightarrow \mathcal{P} = \mathcal{NP}$
- $L \notin \mathcal{P} \Longrightarrow$ für jede $\mathcal{NP}$-vollständige Sprache $L'$ gilt $L' \notin \mathcal{P}$

**Beweis Teil 2:**

- Sei $L \notin \mathcal{P}$ und $L$ $\mathcal{NP}$-vollständig.
- Angenommen für eine $\mathcal{NP}$-vollständige Sprache $L'$ gilt: $L' \in \mathcal{P}$
- Dann folgt aus Teil 1 des Satzes $\mathcal{P} = \mathcal{NP}$.
- Dies ist aber ein Widerspruch zur Voraussetzung $L \notin \mathcal{P}$.

# Zusammenfassung

- Die Klasse $\mathcal{P}$ ist die Klasse aller Entscheidungsprobleme/Sprachen die mit einer **deterministischen** Turingmaschine in polynomieller Zeit gelöst werden können.

- Die Klasse $\mathcal{NP}$ ist die Klasse aller Entscheidungsprobleme/Sprachen die mit einer **nicht-deterministischen** Turingmaschine in polynomieller Zeit gelöst werden können.

- Informell ausgedrückt gehört $\Pi$ zu $\mathcal{NP}$, falls $\Pi$ folgende Eigenschaft hat:
  Ist die Antwort bei Eingabe eines Instanz *I* von $\Pi$ Ja, dann kann die Korrektheit eines Beweises (Zeugen) dafür in polynomialer Zeit überprüft werden.

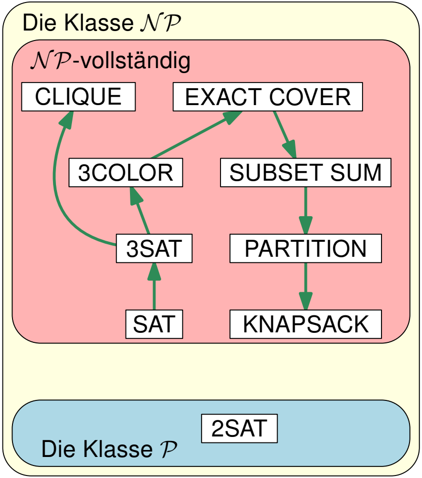
# Zusammenfassung

- Eine polynomiale Transformation einer Sprache $L_1 \subseteq \Sigma_1^*$ in eine Sprache $L_2 \subseteq \Sigma_2^*$ ist eine Funktion $f\colon \Sigma_1^* \to \Sigma_2^*$ mit den Eigenschaften:
  - es existiert eine polynomiale deterministische Turing-Maschine, die $f$ berechnet;
  - für alle $x \in \Sigma_1^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$.
- Eine Sprache $L$ heißt $\mathcal{NP}$-vollständig, falls gilt:
  - $L \in \mathcal{NP}$ und
  - für alle $L' \in \mathcal{NP}$ gilt $L' \propto L$ ($\mathcal{NP}$-Schwere).
- **Bedeutung**:
  Unter der Annahme $\mathcal{P} \neq \mathcal{NP}$ gibt es kein polynomielles Lösungsverfahren für ein $\mathcal{NP}$-vollständiges Problem.

# Zusammenfassung

- Mit dem Satz von Cook haben wir direkt gezeigt, dass das Problem SAT $\mathcal{NP}$-schwer ist.
- Bei allen anderen Problemen haben wir polynomielle Transformationen (Reduktionen) benutzt um die $\mathcal{NP}$-Schwere nachzuweisen:

$$\begin{aligned}\text{SAT} &\propto \text{3SAT}\\ &\propto \text{3COLOR}\\ &\propto \text{EXACT COVER}\\ &\propto \text{SUBSET SUM}\\ &\propto \text{PARTITION}\\ &\propto \text{KNAPSACK}\end{aligned}$$



# Ein Blick über den Tellerrand

- Entscheidungsprobleme außerhalb von $\mathcal{P}$ und $\mathcal{NP}$
- Probleme die nicht Entscheidungsprobleme sind

# Die Klassen $\mathcal{NPC}$ und $\mathcal{NPI}$

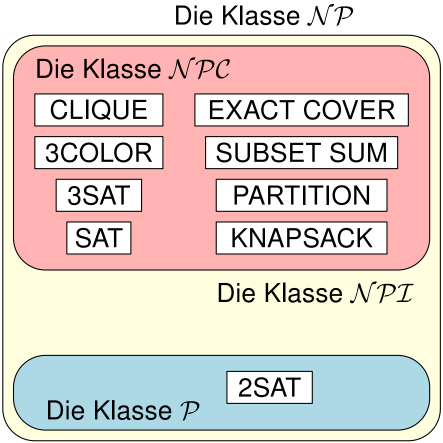
- Die Klasse $\mathcal{NPC}$ ($\mathcal{NP}$-complete) sei die Klasse der $\mathcal{NP}$-vollständigen Sprachen/Probleme.
- Die Klasse $\mathcal{NPI}$ ($\mathcal{NP}$-intermediate) ist definiert durch $\mathcal{NPI} := \mathcal{NP} \setminus (\mathcal{P} \cup \mathcal{NPC})$.

# Die Klassen $\mathcal{NPC}$ und $\mathcal{NPI}$

- Die Klasse $\mathcal{NPC}$ ($\mathcal{NP}$-complete) sei die Klasse der $\mathcal{NP}$-vollständigen Sprachen/Probleme.
- Die Klasse $\mathcal{NPI}$ ($\mathcal{NP}$-intermediate) ist definiert durch $\mathcal{NPI} := \mathcal{NP} \setminus (\mathcal{P} \cup \mathcal{NPC})$.

**Satz (Ladner (1975)):**
Falls $\mathcal{P} \neq \mathcal{NP}$, so folgt $\mathcal{NPI} \neq \emptyset$.

# Vermutete Situation



# Die Klassen co-$\mathcal{P}$ und co-$\mathcal{NP}$

## Klasse der Komplementsprachen

- Die Klasse **co-$\mathcal{P}$** ist die Klasse aller Sprachen

$$L' = \Sigma^* \setminus L \text{ mit } L \subseteq \Sigma^* \text{ und } L \in \mathcal{P}.$$

- Die Klasse **co-$\mathcal{NP}$** ist die Klasse aller Sprachen

$$L' = \Sigma^* \setminus L \text{ mit } L \subseteq \Sigma^* \text{ und } L \in \mathcal{NP}.$$

# Die Klassen co-$\mathcal{P}$ und co-$\mathcal{NP}$

## Klasse der Komplementsprachen

- Die Klasse **co-$\mathcal{P}$** ist die Klasse aller Sprachen

$$L' = \Sigma^* \setminus L \text{ mit } L \subseteq \Sigma^* \text{ und } L \in \mathcal{P}.$$

- Die Klasse **co-$\mathcal{NP}$** ist die Klasse aller Sprachen

$$L' = \Sigma^* \setminus L \text{ mit } L \subseteq \Sigma^* \text{ und } L \in \mathcal{NP}.$$

**Wir wissen:** $\mathcal{P} = \text{co-}\mathcal{P} \subseteq \text{co-}\mathcal{NP}$.

# Die Klassen co-$\mathcal{P}$ und co-$\mathcal{NP}$

## Klasse der Komplementsprachen

- Die Klasse **co-$\mathcal{P}$** ist die Klasse aller Sprachen

$$L' = \Sigma^* \setminus L \text{ mit } L \subseteq \Sigma^* \text{ und } L \in \mathcal{P}.$$

- Die Klasse **co-$\mathcal{NP}$** ist die Klasse aller Sprachen

$$L' = \Sigma^* \setminus L \text{ mit } L \subseteq \Sigma^* \text{ und } L \in \mathcal{NP}.$$

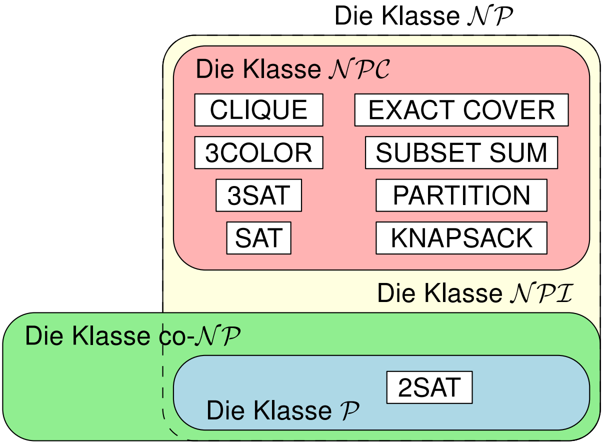
**Wir wissen:** $\mathcal{P} = \text{co-}\mathcal{P} \subseteq \text{co-}\mathcal{NP}$.

**Frage:** Gilt auch $\mathcal{NP} = \text{co-}\mathcal{NP}$?

- Sollte $\mathcal{NP} \neq \text{co-}\mathcal{NP}$ sein, dann würde auch $\mathcal{P} \neq \mathcal{NP}$ gelten.
- Vermutlich ist $\mathcal{NP} \neq \text{co-}\mathcal{NP}$

  $\rightsquigarrow$ Verschärfung der $\mathcal{P} \neq \mathcal{NP}$-Vermutung.
- Aber was würde aus $\mathcal{NP} = \text{co-}\mathcal{NP}$ folgen?

# Vermutete Situation



# Das TSP-Komplement-Problem

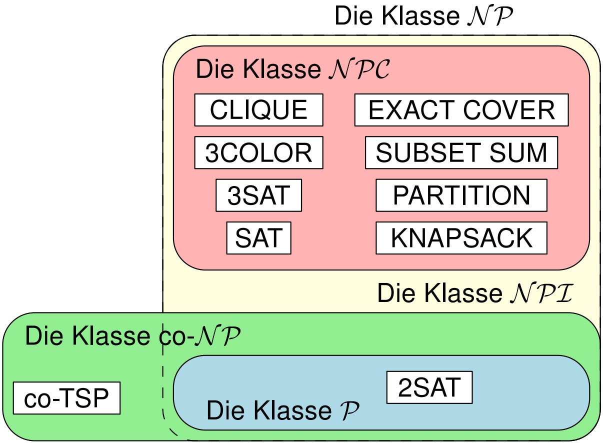
**Problem co-TSP**

**Gegeben:** Graph $G = (V, E)$,
Kantengewichtung $c \colon E \to \mathbb{Z}^+$,
Parameter $K \in \mathbb{Z}$.

**Frage:** Gibt es <u>keine</u> Tour der Länge $\leq K$?

- **Bemerkung:**
  Für ein vernünftiges Kodierungsschema von TSP ist es leicht nachzuweisen, ob ein gegebener String eine gültige TSP-Instanz repräsentiert.
- co-TSP ist in co-$\mathcal{NP}$, denn TSP ist in $\mathcal{NP}$.
- Frage: Ist co-TSP in $\mathcal{NP}$? Vermutung: Nein.

# Vermutete Situation



# $\mathcal{NP}$-vollständig vs. co-$\mathcal{NP}$

**Lemma.**
Falls $L$ $\mathcal{NP}$-vollständig ist und $L \in$ co-$\mathcal{NP}$, so ist $\mathcal{NP} =$ co-$\mathcal{NP}$.

# $\mathcal{NP}$-vollständig vs. co-$\mathcal{NP}$

**Lemma.**
Falls $L$ $\mathcal{NP}$-vollständig ist und $L \in$ co-$\mathcal{NP}$, so ist $\mathcal{NP} =$ co-$\mathcal{NP}$.

**Beweis:**

- Sei $L \in$ co-$\mathcal{NP}$ und $L$ $\mathcal{NP}$-vollständig.
- Dann existiert eine polynomiale nichtdet. Berechnung für $L^c$.
- Für alle $L' \in \mathcal{NP}$ gilt: $L' \propto L$.
- Also existiert eine det. poly. Transformation ${L'}^c \propto L^c$.
- Deshalb existiert eine poly. nichtdet. Berechnung für ${L'}^c$.
- Also $L' \in$ co-$\mathcal{NP}$.

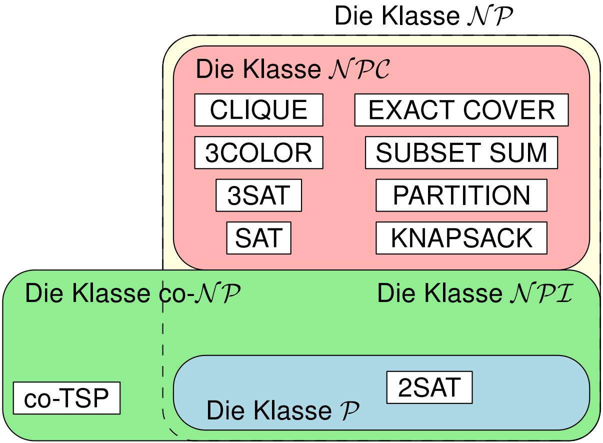
# $\mathcal{NP}$-vollständig vs. co-$\mathcal{NP}$

**Lemma.**
Falls $L$ $\mathcal{NP}$-vollständig ist und $L \in$ co-$\mathcal{NP}$, so ist $\mathcal{NP} =$ co-$\mathcal{NP}$.

**Bemerkung:**

- Mit der Vermutung $\mathcal{NP} \neq$ co-$\mathcal{NP}$ folgt also $\mathcal{NPC} \cap$ co-$\mathcal{NP} = \emptyset$.
- Unter dieser Annahme:
  Wenn ein Problem in $\mathcal{NP}$ und co-$\mathcal{NP}$ ist, aber nicht in $\mathcal{P}$, so ist es in $\mathcal{NPI}$.

# Vermutete Situation



# Das Problem Subgraphisomorphie

**Problem Subgraphisomorphie**

**Gegeben:** Graph $G = (V, E)$,
Graph $H = (V', E')$ mit $|V'| < |V|$

**Frage:** Gibt es eine Menge $U \subseteq V$ mit $|U| = |V'|$ und
eine bijektive Abbildung $\mathrm{Iso}\colon V' \to U$,
so dass für alle $x, y \in V'$ gilt:
$\{x, y\} \in E' \iff \{\mathrm{Iso}(x), \mathrm{Iso}(y)\} \in E$

- **Frage anschaulich:** Ist $H$ isomorph zu einem Subgraphen von $G$?

# Das Problem Subgraphisomorphie

**Problem Subgraphisomorphie**

**Gegeben:** Graph $G = (V, E)$,
Graph $H = (V', E')$ mit $|V'| < |V|$

**Frage:** Gibt es eine Menge $U \subseteq V$ mit $|U| = |V'|$ und
eine bijektive Abbildung $\text{Iso}\colon V' \to U$,
so dass für alle $x, y \in V'$ gilt:
$\{x, y\} \in E' \iff \{\text{Iso}(x), \text{Iso}(y)\} \in E$

- **Frage anschaulich:** Ist $H$ isomorph zu einem Subgraphen von $G$?
- Problem Subgraphisomorphie ist $\mathcal{NP}$-vollständig (ohne Beweis).

# Das Problem Graphisomorphie

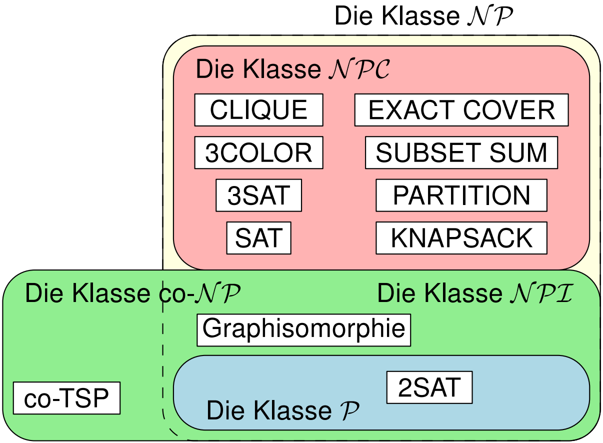
**Problem Graphisomorphie**

**Gegeben:** Graph $G = (V, E)$,
Graph $H = (V', E')$ mit $|V| = |V'|$

**Frage:** Existiert eine bijektive Abbildung $\mathrm{Iso}\colon V' \to V$ mit
$\{x, y\} \in E' \iff \{\mathrm{Iso}(x), \mathrm{Iso}(y)\} \in E$?

- **Frage anschaulich:** Sind $G$ und $H$ isomorph?
- Graphisomorphie ist ein Kandidat für ein Problem aus $\mathcal{NPI}$.
- Graphisomorphie liegt in $\mathcal{NP}$ und co-$\mathcal{NP}$.

# Vermutete Situation



# Die Polynomielle Hierarchie

- Betrachte SAT Instanz $(U, C)$ und eine Aufteilung der Variablen in disjunkte Mengen $U_1$ und $U_2$.
- Die Instanz heiße lösbar wenn
  - es eine Wahrheitsbelegung $t_1$ von $U_1$ gibt,
  - so dass für alle Wahrheitsbelegungen $t_2$ von $U_2$
  - alle Klauseln in $C$ erfüllt sind.
- $\rightsquigarrow$ bezeichnet als QSAT$_2$ oder $\exists\forall$SAT

# Die Polynomielle Hierarchie

- Betrachte SAT Instanz $(U, C)$ und eine Aufteilung der Variablen in disjunkte Mengen $U_1$ und $U_2$.
- Die Instanz heiße lösbar wenn
  - es eine Wahrheitsbelegung $t_1$ von $U_1$ gibt,
  - so dass für alle Wahrheitsbelegungen $t_2$ von $U_2$
  - alle Klauseln in $C$ erfüllt sind.
- $\rightsquigarrow$ bezeichnet als $\mathrm{QSAT}_2$ oder $\exists\forall$SAT
- Analog betrachtet man $\mathrm{QSAT}_3$ oder $\exists\forall\exists$SAT.

# Die Polynomielle Hierarchie

- Betrachte SAT Instanz $(U, C)$ und eine Aufteilung der Variablen in disjunkte Mengen $U_1$ und $U_2$.
- Die Instanz heiße lösbar wenn
  - es eine Wahrheitsbelegung $t_1$ von $U_1$ gibt,
  - so dass für alle Wahrheitsbelegungen $t_2$ von $U_2$
  - alle Klauseln in $C$ erfüllt sind.
- $\rightsquigarrow$ bezeichnet als $\text{QSAT}_2$ oder $\exists\forall$SAT
- Analog betrachtet man $\text{QSAT}_3$ oder $\exists\forall\exists$SAT.
- Analog betrachtet man $\text{QSAT}_4$ oder $\exists\forall\exists\forall$SAT.

# Die Polynomielle Hierarchie

- Betrachte SAT Instanz $(U, C)$ und eine Aufteilung der Variablen in disjunkte Mengen $U_1$ und $U_2$.
- Die Instanz heiße lösbar wenn
  - es eine Wahrheitsbelegung $t_1$ von $U_1$ gibt,
  - so dass für alle Wahrheitsbelegungen $t_2$ von $U_2$
  - alle Klauseln in $C$ erfüllt sind.
- $\rightsquigarrow$ bezeichnet als $\text{QSAT}_2$ oder $\exists\forall$SAT
- Analog betrachtet man $\text{QSAT}_3$ oder $\exists\forall\exists$SAT.
- Analog betrachtet man $\text{QSAT}_4$ oder $\exists\forall\exists\forall$SAT.

**polynomielle Hierarchie**

**PH**: $\quad \text{SAT} \subseteq \text{QSAT}_2 \subseteq \text{QSAT}_3 \subseteq \text{QSAT}_4 \subseteq \text{QSAT}_5 \subseteq \cdots$

# Die Polynomielle Hierarchie

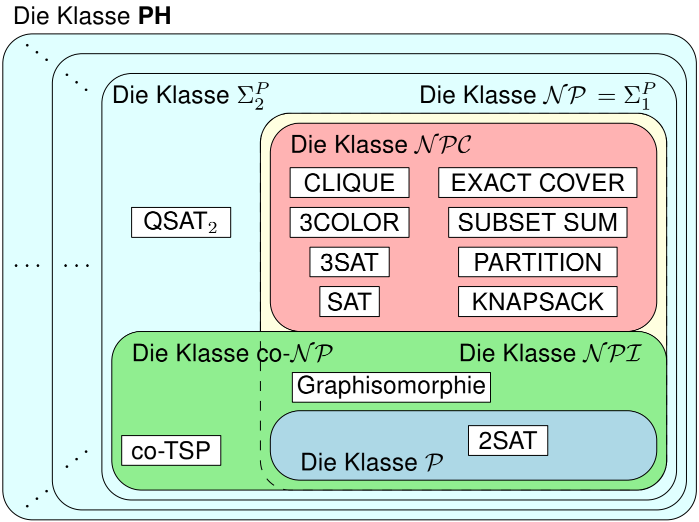
- Betrachte SAT Instanz $(U, C)$ und eine Aufteilung der Variablen in disjunkte Mengen $U_1$ und $U_2$.
- Die Instanz heiße lösbar wenn
  - es eine Wahrheitsbelegung $t_1$ von $U_1$ gibt,
  - so dass für alle Wahrheitsbelegungen $t_2$ von $U_2$
  - alle Klauseln in $C$ erfüllt sind.
- $\rightsquigarrow$ bezeichnet als $\text{QSAT}_2$ oder $\exists\forall$SAT
- Analog betrachtet man $\text{QSAT}_3$ oder $\exists\forall\exists$SAT.
- Analog betrachtet man $\text{QSAT}_4$ oder $\exists\forall\exists\forall$SAT.

**polynomielle Hierarchie**

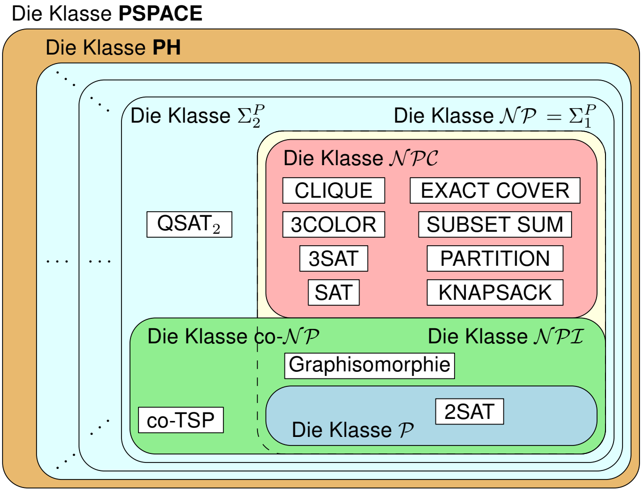
**PH**: $\quad \text{SAT} \subseteq \text{QSAT}_2 \subseteq \text{QSAT}_3 \subseteq \text{QSAT}_4 \subseteq \text{QSAT}_5 \subseteq \cdots$

- Es gilt $\exists$SAT ist vollständig für $\mathcal{NP}$.
- Es gilt $\forall$SAT ist vollständig für co-$\mathcal{NP}$.

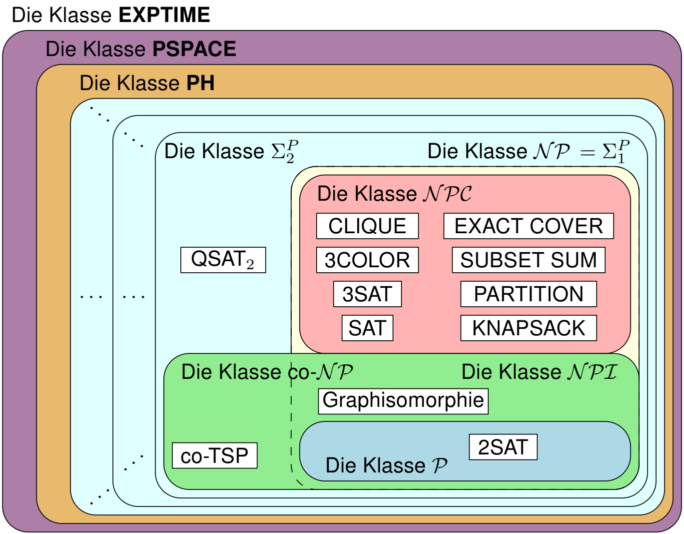
# Vermutete Situation



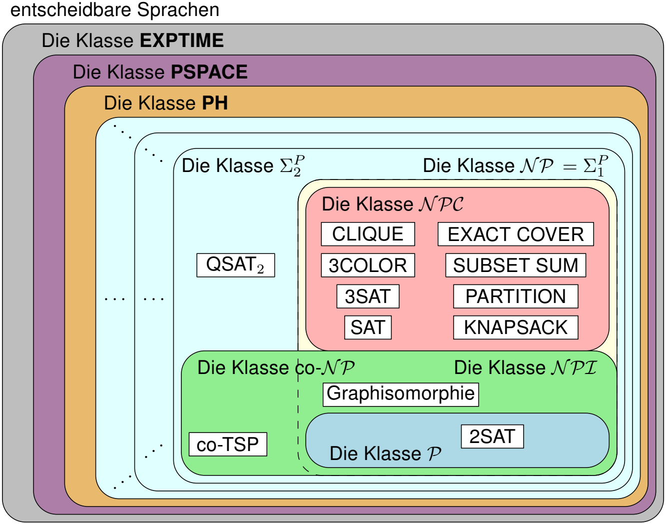
# Vermutete Situation



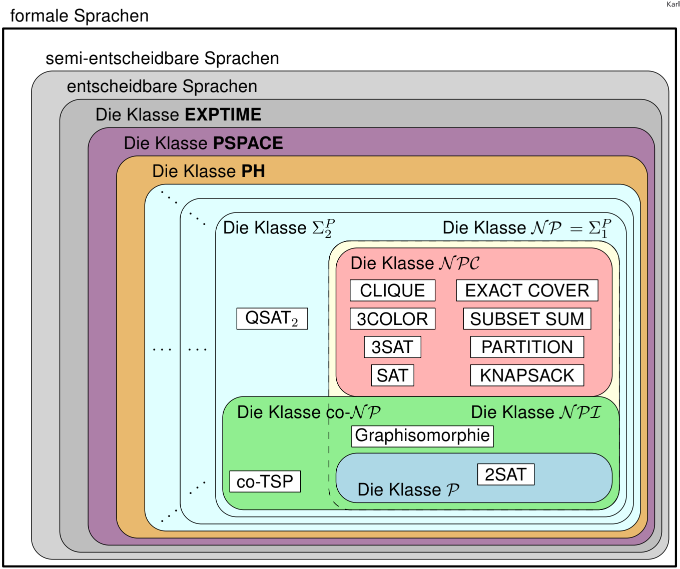
# Vermutete Situation



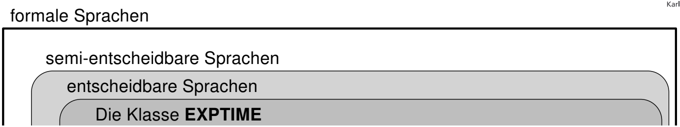
# Vermutete Situation



# Vermutete Situation



# Vermutete Situation



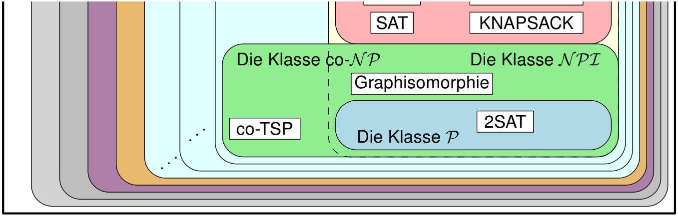

**Testen Sie sich:**

Lesen Sie den Artikel

A Short Guide to Hard Problems

auf Quanta Magazine

https://www.quantamagazine.org/a-short-guide-to-hard-problems-20180716/



# Ein Blick über den Tellerrand

- Entscheidungsprobleme außerhalb von $\mathcal{P}$ und $\mathcal{NP}$
- Probleme die nicht Entscheidungsprobleme sind

# Suchprobleme

Ein **Suchproblem** $\Pi$ wird beschrieben durch

- die Menge $D_\Pi$ der Instanzen und
- für $I \in D_\Pi$ die Menge $S_\Pi(I)$ <u>aller</u> Lösungen von $I$.

Die **Lösung** eines Suchproblems für eine Instanz $I \in D_\Pi$ ist

- ein beliebiges Element aus $S_\Pi(I)$ falls $S_\Pi(I) \neq \emptyset$
- $\emptyset$ sonst

# Beispiel: TSP-Suchproblem

**TSP-Suchproblem (Variante 1)**

**Gegeben:** Graph $G = (V, E)$,
Gewichtsfunktion $c \colon E \to \mathbb{Q}$

**Aufgabe:** Gib eine optimale Tour zu $G$ bezüglich $c$ an.

- Bemerkung: $S_\Pi(G)$ ist die Menge aller optimalen Touren zu $G$.

**TSP-Suchproblem (Variante 2)**

**Gegeben:** Graph $G = (V, E)$,
Gewichtsfunktion $c \colon E \to \mathbb{Q}$,
Parameter $k \in \mathbb{Q}$

**Aufgabe:** Gib eine Tour zu $G$ bzgl. $c$ mit Länge höchstens $k$ an.

- Bemerkung: $S_\Pi(G)$ ist die Menge aller Touren der Länge höchstens $k$.

# Beispiel: Hamilton-Kreis Suchproblem

Gegeben ist ein Graph $G = (V, E)$.

Ein Hamilton-Kreis in $G$ ist eine zyklische Permutation $\pi$ auf $V$, so dass

$$\{\pi(i), \pi(i+1)\} \in E \text{ für } 1 \leq i \leq n \text{ ist.}$$

**Hamilton-Kreis Suchproblem**

**Gegeben:** Ein ungerichteter, ungewichteter Graph $G = (V, E)$

**Aufgabe:** Gib einen Hamilton-Kreis in $G$ an, falls einer existiert.

- Bemerkung: $S_\Pi(G)$ ist die Menge aller Hamilton-Kreise in $G$.

# $\mathcal{NP}$-Schwere bei Suchproblemen

Karlsruher Institut für Technologie

Für Suchprobleme gibt es (ähnlich wie zu Entscheidungsproblemen):

- Eine Variante der Orakel-Turing-Maschine.
- Eine Klasse $\mathcal{NP}$ der Suchprobleme, die in polynomieller Zeit von einer OTM gelöst werden können.
- Eine Klasse $\mathcal{P}$ der Suchprobleme, die in polynomieller Zeit von einer deterministischen TM gelöst werden können.
- Den Begriff der $\mathcal{NP}$-Schwere, und sogenannte Turingreduktionen $\propto_T$.
- Die Frage ob $\mathcal{P} = \mathcal{NP}$?

# $\mathcal{NP}$-Schwere bei Suchproblemen

Für Suchprobleme gibt es (ähnlich wie zu Entscheidungsproblemen):

- Eine Variante der Orakel-Turing-Maschine.
- Eine Klasse $\mathcal{NP}$ der Suchprobleme, die in polynomieller Zeit von einer OTM gelöst werden können.
- Eine Klasse $\mathcal{P}$ der Suchprobleme, die in polynomieller Zeit von einer deterministischen TM gelöst werden können.
- Den Begriff der $\mathcal{NP}$-Schwere, und sogenannte Turingreduktionen $\propto_T$.
- Die Frage ob $\mathcal{P} = \mathcal{NP}$?

**Bemerkung:**

- Die Suchprobleme für Hamilton-Kreis und TSP sind $\mathcal{NP}$-schwer.

# Aufzählungsprobleme

Ein **Aufzählungsproblem** $\Pi$ ist gegeben durch

- die Menge $D_\Pi$ der Problembeispiele und
- für $I \in D_\Pi$ die Menge $S_\Pi(I)$ aller Lösungen von $I$.

Die **Lösung** eines Aufzählungsproblems für eine Instanz $I \in D_\Pi$ ist

- die Angabe der Kardinalität von $S_\Pi(I)$, d.h. von $|S_\Pi(I)|$.

# Beispiel: Hamilton-Kreis Aufzählungsproblem

**Hamilton-Kreis Aufzählungsproblem**

**Gegeben:** Ein ungerichteter, ungewichteter Graph $G = (V, E)$

**Aufgabe:** Wieviele Hamilton-Kreise gibt es in $G$?

# Beispiel: Hamilton-Kreis Aufzählungsproblem

**Hamilton-Kreis Aufzählungsproblem**

**Gegeben:** Ein ungerichteter, ungewichteter Graph $G = (V, E)$

**Aufgabe:** Wieviele Hamilton-Kreise gibt es in $G$?

## Bemerkung:

- Aufzählprobleme sind sehr schwierig!
- Beispiel: Permanente einer Matrix (Anzahl der perfekten Matchings in einem bipartiten Graphen) ist $\#P$-vollständig.
- **Satz von Toda.**
  Jedes Problem in **PH** (z.B. $\text{QSAT}_k$) kann durch einen Aufruf eines $\#P$-vollständigen Problems gelöst werden.