4.1.1 Die Klasse P

Worst-Case Laufzeit $t_M(n)$ einer Turingmaschine M auf Eingaben der Länge n:

$$t_M(n) = \max_{w \in \Sigma^n} t_M(w).$$

Definition 4.1

Entscheidungsproblem L gehört zu der **Komplexitätsklasse P**, wenn es eine TM M gibt, die L entscheidet, und eine Konstante $k \in \mathbb{N}$, für die $t_M(n) = O(n^k)$ gilt.

Beobachtung: Die Klasse P ändert sich nicht, wenn **Registermaschinen im logarithmischen Kostenmaß** statt Turingmaschinen eingesetzt werden.

Idee: P enthält die **effizient lösbaren Probleme**.

4.1.1 Die Klasse P

Clique in einem Graphen $G = (V, E)$ ist
$V' \subseteq V$ mit $\{u, v\} \in E$ für alle $u, v \in V'$

Varianten des Cliquenproblems

- **Optimierungsvariante**
  Eingabe: ungerichteter Graph $G = (V, E)$
  Aufgabe: Berechne eine Clique von G mit maximaler Kardinalität.
- **Wertvariante**
  Eingabe: ungerichteter Graph $G = (V, E)$
  Aufgabe: Berechne das größte $k^\star \in \mathbb{N}$, für das es eine $k^\star$-Clique in G gibt.
- **Entscheidungsvariante**
  Eingabe: ungerichteter Graph $G = (V, E)$ und ein Wert $k \in \mathbb{N}$
  Aufgabe: Entscheide, ob es in G eine Clique der Größe mindestens k gibt.

Theorem 4.2

Entweder gibt es für alle drei Varianten des Cliquenproblems polynomielle Algorithmen oder für gar keine.

Beweisidee:

Optimierungsvariante polynomiell lösbar.

$\Leftrightarrow$ Wertvariante polynomiell lösbar.

$\Leftrightarrow$ Entscheidungsvariante polynomiell lösbar.

□

4.1.1 Die Klasse P

Beispiele:

- Das Zusammenhangsproblem in ungerichteten Graphen

4.1.1 Die Klasse P

Beispiele:

- Das Zusammenhangsproblem in ungerichteten Graphen $\Rightarrow$ gehört zu P.

4.1.1 Die Klasse P

Beispiele:

- Das Zusammenhangsproblem in ungerichteten Graphen $\Rightarrow$ gehört zu P.
- Das Spannbaumproblem

4.1.1 Die Klasse P

Beispiele:

- Das Zusammenhangsproblem in ungerichteten Graphen $\Rightarrow$ gehört zu P.
- Das Spannbaumproblem $\Rightarrow$ gehört zu P.

4.1.1 Die Klasse P

Beispiele:

- Das Zusammenhangsproblem in ungerichteten Graphen $\Rightarrow$ gehört zu P.
- Das Spannbaumproblem $\Rightarrow$ gehört zu P.
- Das Cliquenproblem

4.1.1 Die Klasse P

Beispiele:

- Das Zusammenhangsproblem in ungerichteten Graphen $\Rightarrow$ gehört zu P.
- Das Spannbaumproblem $\Rightarrow$ gehört zu P.
- Das Cliquenproblem $\Rightarrow$ kein polynomieller Algorithmus bekannt.

4.1.1 Die Klasse P

Beispiele:

- Das Zusammenhangsproblem in ungerichteten Graphen $\Rightarrow$ gehört zu P.
- Das Spannbaumproblem $\Rightarrow$ gehört zu P.
- Das Cliquenproblem $\Rightarrow$ kein polynomieller Algorithmus bekannt.
- Das Rucksackproblem

4.1.1 Die Klasse P

Beispiele:

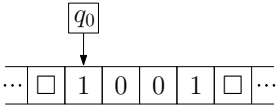
- Das Zusammenhangsproblem in ungerichteten Graphen $\Rightarrow$ gehört zu P.
- Das Spannbaumproblem $\Rightarrow$ gehört zu P.
- Das Cliquenproblem $\Rightarrow$ kein polynomieller Algorithmus bekannt.
- Das Rucksackproblem $\Rightarrow$ kein polynomieller Algorithmus bekannt.

4.1.2 Die Klasse NP

Definition 2.1

Eine **Turingmaschine (TM)** M ist ein 7-Tupel $(Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, das aus den folgenden Komponenten besteht.

- Q, die **Zustandsmenge**, ist eine endliche Menge von **Zuständen**.
- $\Sigma \supseteq \{0, 1\}$, das **Eingabealphabet**, ist eine endliche Menge von Zeichen.
- $\Gamma \supseteq \Sigma$, das **Bandalphabet**, ist eine endliche Menge von Zeichen.
- $\square \in \Gamma \setminus \Sigma$ ist das **Leerzeichen**.
- $q_0 \in Q$ ist der **Startzustand**.
- $\bar{q}$ ist der **Endzustand**.
- $\delta\colon (Q \setminus \{\bar{q}\}) \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$ ist die **Zustandsüberführungsfunktion**.

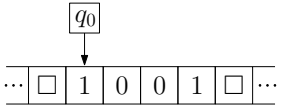


$\delta(q_0, 1) = (q, 1, R)$

Definition 4.3

Eine **nichtdeterministische Turingmaschine (NTM)** M ist ein 7-Tupel $(Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, das aus den folgenden Komponenten besteht.

- Q, die **Zustandsmenge**, ist eine endliche Menge von **Zuständen**.
- $\Sigma \supseteq \{0, 1\}$, das **Eingabealphabet**, ist eine endliche Menge von Zeichen.
- $\Gamma \supseteq \Sigma$, das **Bandalphabet**, ist eine endliche Menge von Zeichen.
- $\square \in \Gamma \setminus \Sigma$ ist das **Leerzeichen**.
- $q_0 \in Q$ ist der **Startzustand**.
- $\bar{q}$ ist der **Endzustand**.
- $\delta \subseteq ((Q \setminus \{\bar{q}\}) \times \Gamma) \times (Q \times \Gamma \times \{L, N, R\})$ ist die **Zustandsüberführungsrelation**.

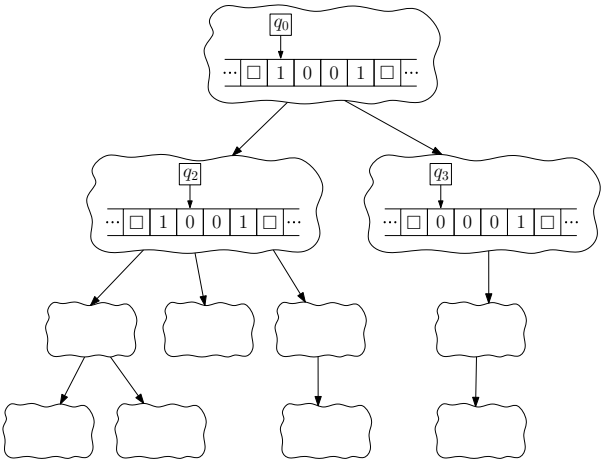


$((q_0, 1), (q, 1, R)) \in \delta$

oder

$((q_0, 1), (q, 0, L)) \in \delta$

Rechenbaum einer Turingmaschine:



Konfiguration = Zustand, Bandinhalt, Kopfposition

Wurzel

= Startkonfiguration

Kante

= erlaubter Übergang

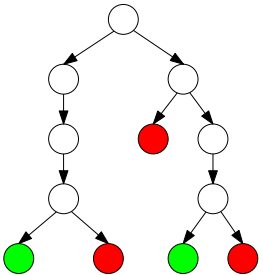
Blatt

= Konfiguration ohne erlaubten Übergang in δ

Rechenweg = Weg von der Wurzel zu einem Blatt

Definition 4.4

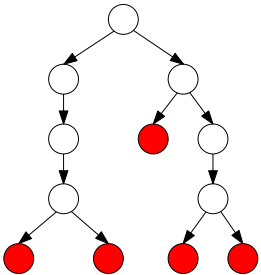
Eine NTM M **akzeptiert** eine Eingabe $w \in \Sigma^*$, wenn es Rechenweg von M gibt, der bei Eingabe w zu einer akzeptierenden Endkonfiguration führt.



Eingabe wird akzeptiert

Definition 4.4

Eine NTM M **akzeptiert** eine Eingabe $w \in \Sigma^*$, wenn es Rechenweg von M gibt, der bei Eingabe w zu einer akzeptierenden Endkonfiguration führt.

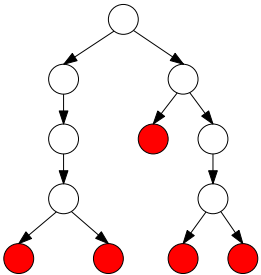


Eingabe wird nicht akzeptiert

Definition 4.4

Eine NTM M **akzeptiert** eine Eingabe $w \in \Sigma^*$, wenn es Rechenweg von M gibt, der bei Eingabe w zu einer akzeptierenden Endkonfiguration führt.

Sei $L(M) \subseteq \Sigma^*$ die Menge der von M **akzeptierten Eingaben**. M **entscheidet** die Sprache $L(M)$, wenn sie für jede Eingabe auf jedem Rechenweg hält.



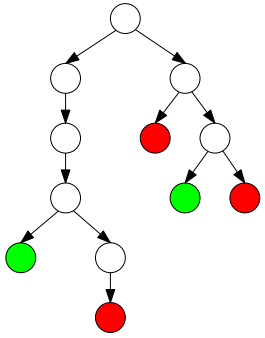
Eingabe wird nicht akzeptiert

Definition 4.5

Die **Laufzeit** $t_M(w)$ einer nichtdeterministischen Turingmaschine M auf einer Eingabe $w \in \Sigma^*$ ist definiert als die **Länge des längsten Rechenweges** von M bei Eingabe w.

Gibt es bei Eingabe w einen Rechenweg, auf dem M nicht terminiert, so ist die Laufzeit unendlich.

Sei $t_M(n) = \max_{w \in \Sigma^n} t_M(w)$ die **Worst-Case-Laufzeit** für Eingaben der Länge $n \in \mathbb{N}$.



Laufzeit = 5

4.1.2 Die Klasse NP

Definition 4.6

Ein Entscheidungsproblem L gehört genau dann zu der **Komplexitätsklasse NP**, wenn es eine nichtdeterministische Turingmaschine M gibt, die L entscheidet, und eine Konstante $k \in \mathbb{N}$, für die $t_M(n) = O(n^k)$ gilt.

NP wurde nicht mit dem Ziel definiert, ein physikalisch realisierbares Rechnermodell zu finden, sondern als **theoretisches Hilfsmittel**.

Theorem 4.7

Die Entscheidungsvarianten des Cliquenproblems und des Rucksackproblems gehören zu NP.

Beweis: Wir starten mit dem Cliquenproblem.

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

Frage: Existiert in G eine k-Clique?

4.1.2 Die Klasse NP

Beweis: Wir starten mit dem Cliquenproblem.

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

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

4.1.2 Die Klasse NP

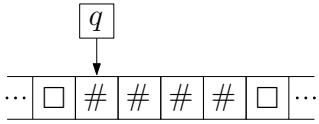
Beweis: Wir starten mit dem Cliquenproblem.

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

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$

$((q, \#), (q, 1, R)) \in \delta$

4.1.2 Die Klasse NP

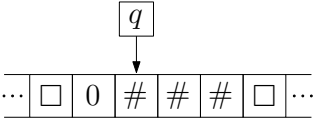
Beweis: Wir starten mit dem Cliquenproblem.

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

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$

$((q, \#), (q, 1, R)) \in \delta$

4.1.2 Die Klasse NP

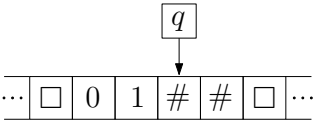
Beweis: Wir starten mit dem Cliquenproblem.

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

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$

$((q, \#), (q, 1, R)) \in \delta$

4.1.2 Die Klasse NP

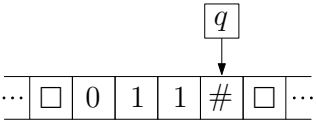
Beweis: Wir starten mit dem Cliquenproblem.

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

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$

$((q, \#), (q, 1, R)) \in \delta$

4.1.2 Die Klasse NP

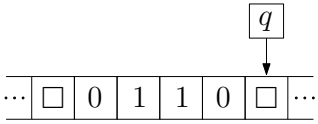
Beweis: Wir starten mit dem Cliquenproblem.

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

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$

$((q, \#), (q, 1, R)) \in \delta$

4.1.2 Die Klasse NP

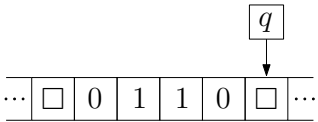
Beweis: Wir starten mit dem Cliquenproblem.

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

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$
$((q, \#), (q, 1, R)) \in \delta$

NTM kann jede Zeichenkette aus $x \in \{0, 1\}^n$ nichtdeterministisch schreiben.

Interpretiere x als Knotenauswahl $V' \subseteq V$.

4.1.2 Die Klasse NP

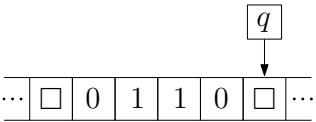
Beweis: Wir starten mit dem Cliquenproblem.

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

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$
$((q, \#), (q, 1, R)) \in \delta$

NTM kann jede Zeichenkette aus $x \in \{0, 1\}^n$ nichtdeterministisch schreiben.

Interpretiere x als Knotenauswahl $V' \subseteq V$.

Phase 2: Akzeptiere genau dann, wenn V' die Größe k besitzt und eine Clique in G ist.

4.1.2 Die Klasse NP

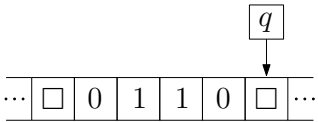
Beweis: Wir starten mit dem Cliquenproblem.

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

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$
$((q, \#), (q, 1, R)) \in \delta$

NTM kann jede Zeichenkette aus $x \in \{0, 1\}^n$ nichtdeterministisch schreiben.

Interpretiere x als Knotenauswahl $V' \subseteq V$.

Phase 2: Akzeptiere genau dann, wenn V' die Größe k besitzt und eine Clique in G ist.

Laufzeit ist polynomiell.

4.1.2 Die Klasse NP

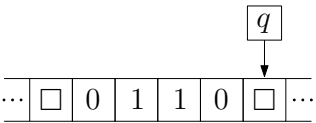
Beweis: Wir starten mit dem Cliquenproblem.

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

Frage: Existiert in G eine k-Clique?

Konstruiere polynomielle NTM M, die CLIQUE entscheidet.

Phase 1: Schreibe $n = |V|$ Rauten, bewege Kopf auf erste Raute, wechsel in Zustand q.



$((q, \#), (q, 0, R)) \in \delta$
$((q, \#), (q, 1, R)) \in \delta$

NTM kann jede Zeichenkette aus $x \in \{0, 1\}^n$ nichtdeterministisch schreiben.

Interpretiere x als Knotenauswahl $V' \subseteq V$.

Phase 2: Akzeptiere genau dann, wenn V' die Größe k besitzt und eine Clique in G ist.

Laufzeit ist polynomiell.

Es gibt k-Clique in G. $\iff$ Es gibt akzeptierenden Rechenweg von M.

4.1.2 Die Klasse NP

Wir betrachten nun das Rucksackproblem.

Eingabe: Nutzenwerte $p_1, \ldots, p_n \in \mathbb{N}$, Gewichte $w_1, \ldots, w_n \in \mathbb{N}$,
Kapazität $t \in \mathbb{N}$, Schranke $z \in \mathbb{N}$.

Frage: Gibt es Teilmenge $I \subseteq \{1, \ldots, n\}$ der Objekte mit $\sum_{i \in I} w_i \leq t$ und $\sum_{i \in I} p_i \geq z$.

4.1.2 Die Klasse NP

Wir betrachten nun das Rucksackproblem.

Eingabe: Nutzenwerte $p_1, \ldots, p_n \in \mathbb{N}$, Gewichte $w_1, \ldots, w_n \in \mathbb{N}$,
Kapazität $t \in \mathbb{N}$, Schranke $z \in \mathbb{N}$.

Frage: Gibt es Teilmenge $I \subseteq \{1, \ldots, n\}$ der Objekte mit $\sum_{i \in I} w_i \leq t$ und $\sum_{i \in I} p_i \geq z$.

Analoges Vorgehen zum Cliquenproblem:

Erzeuge nichtdeterministisch eine Auswahl $I \subseteq \{1, \ldots, n\}$ und teste, ob sie die beiden Bedingungen erfüllt. □

4.1.2 Die Klasse NP

Die NTMs für Clique und das Rucksackproblem haben eine spezielle Struktur:

Nichtdeterminismus wird nur am Anfang benötigt.

4.1.2 Die Klasse NP

Die NTMs für Clique und das Rucksackproblem haben eine spezielle Struktur:
Nichtdeterminismus wird nur am Anfang benötigt.

Theorem 4.8

Eine Sprache $L \subseteq \Sigma^*$ ist genau dann in der Klasse NP enthalten, wenn es eine deterministische Turingmaschine V (einen **Verifizierer**) gibt, deren Worst-Case-Laufzeit polynomiell beschränkt ist, und ein Polynom p, sodass für jede Eingabe $x \in \Sigma^*$ gilt

$$x \in L \iff \exists y \in \{0,1\}^* : |y| \leq p(|x|) \text{ und } V \text{ akzeptiert } x\#y.$$

Dabei sei $\#$ ein beliebiges Zeichen, das zum Eingabealphabet des Verifizierers, aber nicht zu Σ gehört.

Beweis:

„$\Rightarrow$“: Sei $L \in$ NP. Dann gibt es NTM $M = (Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, die L entscheidet, und ein Polynom r mit $t_M(n) \leq r(n)$.

Beweis:

„$\Rightarrow$“: Sei $L \in \text{NP}$. Dann gibt es NTM $M = (Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, die L entscheidet, und ein Polynom r mit $t_M(n) \leq r(n)$.

Idee: Konstruiere Verifizierer V für L, der M simuliert. Dabei gibt das Zertifikat y die nichtdeterministischen Entscheidungen vor.

Beweis:

„$\Rightarrow$“: Sei $L \in$ NP. Dann gibt es NTM $M = (Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, die L entscheidet, und ein Polynom r mit $t_M(n) \leq r(n)$.

Idee: Konstruiere Verifizierer V für L, der M simuliert. Dabei gibt das Zertifikat y die nichtdeterministischen Entscheidungen vor.

In jeder Konfiguration maximal $\ell := 3|Q||\Gamma|$ mögliche Rechenschritte.
Codiere jeden Rechenschritt mit $\ell^\star := \lceil \log_2 \ell \rceil$ Bits.

Beweis:

„⇒“: Sei $L \in \text{NP}$. Dann gibt es NTM $M = (Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, die L entscheidet, und ein Polynom r mit $t_M(n) \leq r(n)$.

Idee: Konstruiere Verifizierer V für L, der M simuliert. Dabei gibt das Zertifikat y die nichtdeterministischen Entscheidungen vor.

In jeder Konfiguration maximal $\ell := 3|Q||\Gamma|$ mögliche Rechenschritte.
Codiere jeden Rechenschritt mit $\ell^\star := \lceil \log_2 \ell \rceil$ Bits.

Der Verifizierer V erhält die Eingabe $x\#y$ für ein Zertifikat $y \in \{0,1\}^m$ mit $m \leq \ell^\star r(|x|)$.

Beweis:

„⇒“: Sei $L \in \text{NP}$. Dann gibt es NTM $M = (Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, die L entscheidet, und ein Polynom r mit $t_M(n) \leq r(n)$.

Idee: Konstruiere Verifizierer V für L, der M simuliert. Dabei gibt das Zertifikat y die nichtdeterministischen Entscheidungen vor.

In jeder Konfiguration maximal $\ell := 3|Q||\Gamma|$ mögliche Rechenschritte.
Codiere jeden Rechenschritt mit $\ell^\star := \lceil \log_2 \ell \rceil$ Bits.

Der Verifizierer V erhält die Eingabe $x\#y$ für ein Zertifikat $y \in \{0,1\}^m$ mit $m \leq \ell^\star r(|x|)$.

V zerlegt das Zertifikat y in eine Folge von Zeichenketten der Länge jeweils $\ell^\star$ und interpretiert diese als Folge von Rechenschritten.

Beweis:

„⇒“: Sei $L \in \text{NP}$. Dann gibt es NTM $M = (Q, \Sigma, \Gamma, \square, q_0, \bar{q}, \delta)$, die L entscheidet, und ein Polynom r mit $t_M(n) \leq r(n)$.

Idee: Konstruiere Verifizierer V für L, der M simuliert. Dabei gibt das Zertifikat y die nichtdeterministischen Entscheidungen vor.

In jeder Konfiguration maximal $\ell := 3|Q||\Gamma|$ mögliche Rechenschritte.
Codiere jeden Rechenschritt mit $\ell^\star := \lceil \log_2 \ell \rceil$ Bits.

Der Verifizierer V erhält die Eingabe $x\#y$ für ein Zertifikat $y \in \{0,1\}^m$ mit $m \leq \ell^\star r(|x|)$.

V zerlegt das Zertifikat y in eine Folge von Zeichenketten der Länge jeweils $\ell^\star$ und interpretiert diese als Folge von Rechenschritten.

V simuliert den entsprechenden Rechenweg von M und akzeptiert, wenn dieser zu einer akzeptierenden Endkonfiguration führt.

„$\Leftarrow$“: Sei nun eine Sprache L gegeben, für die es ein Polynom p und einen Verifizierer V mit den geforderten Eigenschaften gibt.

Wir konstruieren NTM M für die Sprache L.

„$\Leftarrow$“: Sei nun eine Sprache L gegeben, für die es ein Polynom p und einen Verifizierer V mit den geforderten Eigenschaften gibt.

Wir konstruieren NTM M für die Sprache L.

Bei Eingabe x erzeugt M nichtdeterministisch ein beliebiges Zertifikat $y \in \{0,1\}^*$ mit $|y| \leq p(|x|)$.

„$\Leftarrow$“: Sei nun eine Sprache L gegeben, für die es ein Polynom p und einen Verifizierer V mit den geforderten Eigenschaften gibt.

Wir konstruieren NTM M für die Sprache L.

Bei Eingabe x erzeugt M nichtdeterministisch ein beliebiges Zertifikat $y \in \{0,1\}^*$ mit $|y| \leq p(|x|)$.

Dann simuliert M den Verifizierer V auf der Eingabe $x\#y$ und akzeptiert genau dann die Eingabe x, wenn V die Eingabe $x\#y$ akzeptiert. $\square$

4.1.3 P versus NP

Zusammenfassung:

P enthält alle Probleme, die in **polynomieller Zeit gelöst** werden können.

NP enthält alle Probleme, für die ein **Lösungskandidat in polynomieller Zeit überprüft** werden kann.

4.1.3 P versus NP

Zusammenfassung:

P enthält alle Probleme, die in **polynomieller Zeit gelöst** werden können.

NP enthält alle Probleme, für die ein **Lösungskandidat in polynomieller Zeit überprüft** werden kann.

Aus der Definition folgt direkt $P \subseteq NP$.
Wie mächtig ist die Klasse NP?

4.1.3 P versus NP

Theorem 4.9

Für jede Sprache $L \in \text{NP}$ gibt es eine deterministische Turingmaschine M, die L entscheidet, und ein Polynom r, für das $t_M(n) \leq 2^{r(n)}$ gilt.

4.1.3 P versus NP

Theorem 4.9

Für jede Sprache $L \in \text{NP}$ gibt es eine deterministische Turingmaschine M, die L entscheidet, und ein Polynom r, für das $t_M(n) \leq 2^{r(n)}$ gilt.

Beweis: Es sei $L \in \text{NP}$ beliebig. Dann gibt es Polynom p und einen polynomiellen Verifizierer V, sodass für jede Eingabe $x \in \Sigma^*$ gilt:

$$x \in L \iff \exists y \in \{0,1\}^* : |y| \leq p(|x|) \text{ und } V \text{ akzeptiert } x\#y.$$

4.1.3 P versus NP

Theorem 4.9

Für jede Sprache $L \in \text{NP}$ gibt es eine deterministische Turingmaschine M, die L entscheidet, und ein Polynom r, für das $\mathbf{t_M(n) \leq 2^{r(n)}}$ gilt.

Beweis: Es sei $L \in \text{NP}$ beliebig. Dann gibt es Polynom p und einen polynomiellen Verifizierer V, sodass für jede Eingabe $x \in \Sigma^*$ gilt:

$$x \in L \iff \exists y \in \{0,1\}^* : |y| \leq p(|x|) \text{ und } V \text{ akzeptiert } x\#y.$$

Konstruiere deterministische TM M für L: Bei Eingabe x **simuliere den Verifizierer V auf der Eingabe x#y für alle Zertifikate y** $\in \{0,1\}^*$ mit $|y| \leq p(|x|)$. Akzeptiere x genau dann, wenn der Verifizierer in **mindestens einer dieser Simulationen akzeptiert**.

4.1.3 P versus NP

Theorem 4.9

Für jede Sprache $L \in \text{NP}$ gibt es eine deterministische Turingmaschine M, die L entscheidet, und ein Polynom r, für das $\mathbf{t_M(n) \leq 2^{r(n)}}$ gilt.

Beweis: Es sei $L \in \text{NP}$ beliebig. Dann gibt es Polynom p und einen polynomiellen Verifizierer V, sodass für jede Eingabe $x \in \Sigma^*$ gilt:

$$x \in L \iff \exists y \in \{0,1\}^* : |y| \leq p(|x|) \text{ und } V \text{ akzeptiert } x\#y.$$

Konstruiere deterministische TM M für L: Bei Eingabe x **simuliere den Verifizierer V auf der Eingabe x#y für alle Zertifikate y** $\in \{0,1\}^*$ mit $|y| \leq p(|x|)$. Akzeptiere x genau dann, wenn der Verifizierer in **mindestens einer dieser Simulationen akzeptiert**.

Korrektheit: M akzeptiert x genau dann, wenn ein gültiges Zertifikat y existiert.

Theorem 4.9

Für jede Sprache $L \in \text{NP}$ gibt es eine deterministische Turingmaschine M, die L entscheidet, und ein Polynom r, für das $\mathbf{t_M(n) \leq 2^{r(n)}}$ gilt.

Beweis: Es sei $L \in \text{NP}$ beliebig. Dann gibt es Polynom p und einen polynomiellen Verifizierer V, sodass für jede Eingabe $x \in \Sigma^*$ gilt:

$$x \in L \iff \exists y \in \{0,1\}^* : |y| \leq p(|x|) \text{ und } V \text{ akzeptiert } x\#y.$$

Konstruiere deterministische TM M für L: Bei Eingabe x **simuliere den Verifizierer V auf der Eingabe x#y für alle Zertifikate y** $\in \{0,1\}^*$ mit $|y| \leq p(|x|)$. Akzeptiere x genau dann, wenn der Verifizierer in **mindestens einer dieser Simulationen akzeptiert**.

Korrektheit: M akzeptiert x genau dann, wenn ein gültiges Zertifikat y existiert.

Laufzeit: Anzahl Zertifikate $\sum_{i=0}^{p(|x|)} 2^i = 2^{p(|x|)+1} - 1 = O(2^{p(|x|)})$
Simulation von V für konkrete Eingabe $x\#y$ benötigt polynomielle Zeit. □

Zusammenfassung

- Es gilt P $\subseteq$ NP.
- Alle Probleme aus NP können in **exponentieller Zeit** auf deterministischen Turingmaschinen gelöst werden.

Zusammenfassung

- Es gilt P $\subseteq$ NP.
- Alle Probleme aus NP können in **exponentieller Zeit** auf deterministischen Turingmaschinen gelöst werden.

Offene Frage

Gilt P $=$ NP oder gibt es $L \in$ NP mit $L \notin P$?

Frage: Warum ist die Klasse NP überhaupt interessant?

4.2 NP-Vollständigkeit

Frage: Warum ist die Klasse NP überhaupt interessant?

Wir lernen zunächst, wie wir die Komplexität von Problemen in Relation setzen können.

Frage: Warum ist die Klasse NP überhaupt interessant?

Wir lernen zunächst, wie wir die Komplexität von Problemen in Relation setzen können.

Definition 4.10

Eine **polynomielle Reduktion** einer Sprache $A \subseteq \Sigma_1^*$ auf eine Sprache $B \subseteq \Sigma_2^*$ ist eine Many-One-Reduktion $f\colon \Sigma_1^* \to \Sigma_2^*$, die in polynomieller Zeit berechnet werden kann. Existiert eine solche Reduktion, so heißt A auf B **polynomiell reduzierbar** und wir schreiben $A \leq_p B$.

Frage: Warum ist die Klasse NP überhaupt interessant?

Wir lernen zunächst, wie wir die Komplexität von Problemen in Relation setzen können.

Definition 4.10

Eine **polynomielle Reduktion** einer Sprache $A \subseteq \Sigma_1^*$ auf eine Sprache $B \subseteq \Sigma_2^*$ ist eine Many-One-Reduktion $f\colon \Sigma_1^* \to \Sigma_2^*$, die in polynomieller Zeit berechnet werden kann. Existiert eine solche Reduktion, so heißt A auf B **polynomiell reduzierbar** und wir schreiben $A \leq_p B$.

Erinnerung: Many-One-Reduktion $f : \Sigma_1^* \to \Sigma_2^*$ erfüllt für alle $x \in \Sigma_1^*$:

$$x \in A \iff f(x) \in B.$$

Frage: Warum ist die Klasse NP überhaupt interessant?

Wir lernen zunächst, wie wir die Komplexität von Problemen in Relation setzen können.

Definition 4.10

Eine **polynomielle Reduktion** einer Sprache $A \subseteq \Sigma_1^*$ auf eine Sprache $B \subseteq \Sigma_2^*$ ist eine Many-One-Reduktion $f\colon \Sigma_1^* \to \Sigma_2^*$, die in polynomieller Zeit berechnet werden kann. Existiert eine solche Reduktion, so heißt A auf B **polynomiell reduzierbar** und wir schreiben $A \leq_p B$.

Erinnerung: Many-One-Reduktion $f : \Sigma_1^* \to \Sigma_2^*$ erfüllt für alle $x \in \Sigma_1^*$:

$$x \in A \iff f(x) \in B.$$

Polynomielle Berechenbarkeit:

$\exists k \in \mathbb{N}$: $\exists$ TM M: $\forall x \in \Sigma_1^*$: M berechnet $f(x)$ in Zeit $t_M(|x|) = O(|x|^k)$.

Theorem 4.11

Es seien $A \subseteq \Sigma_1^*$ und $B \subseteq \Sigma_2^*$ zwei Sprachen, für die $A \leq_p B$ gilt.

Ist $B \in \mathrm{P}$, so ist auch $A \in \mathrm{P}$. Ist $A \notin \mathrm{P}$, so ist auch $B \notin \mathrm{P}$.

Theorem 4.11

Es seien $A \subseteq \Sigma_1^*$ und $B \subseteq \Sigma_2^*$ zwei Sprachen, für die $A \leq_p B$ gilt.

Ist $B \in \mathrm{P}$, so ist auch $A \in \mathrm{P}$. Ist $A \notin \mathrm{P}$, so ist auch $B \notin \mathrm{P}$.

Beweis: Sei $A \leq_p B$ mit polynomieller Reduktion $f\colon \Sigma_1^* \to \Sigma_2^*$ und sei $B \in \mathrm{P}$.

Sei M_B die TM, die B in polynomieller Zeit entscheidet.

Sei M_f die TM, die f in polynomieller Zeit berechnet.

Theorem 4.11

Es seien $A \subseteq \Sigma_1^*$ und $B \subseteq \Sigma_2^*$ zwei Sprachen, für die $A \leq_p B$ gilt.
Ist $B \in \mathrm{P}$, so ist auch $A \in \mathrm{P}$. Ist $A \notin \mathrm{P}$, so ist auch $B \notin \mathrm{P}$.

Beweis: Sei $A \leq_p B$ mit polynomieller Reduktion $f\colon \Sigma_1^* \to \Sigma_2^*$ und sei $B \in \mathrm{P}$.

Sei M_B die TM, die B in polynomieller Zeit entscheidet.
Sei M_f die TM, die f in polynomieller Zeit berechnet.

Konstruktion einer TM M_A für A:

1. Berechne bei einer Eingabe x zunächst $f(x)$ mittels M_f.
2. Simuliere anschließend M_B auf $f(x)$.

Theorem 4.11

Es seien $A \subseteq \Sigma_1^*$ und $B \subseteq \Sigma_2^*$ zwei Sprachen, für die $A \leq_p B$ gilt.
Ist $B \in \mathrm{P}$, so ist auch $A \in \mathrm{P}$. Ist $A \notin \mathrm{P}$, so ist auch $B \notin \mathrm{P}$.

Beweis: Sei $A \leq_p B$ mit polynomieller Reduktion $f\colon \Sigma_1^* \to \Sigma_2^*$ und sei $B \in \mathrm{P}$.

Sei M_B die TM, die B in polynomieller Zeit entscheidet.
Sei M_f die TM, die f in polynomieller Zeit berechnet.

Konstruktion einer TM M_A für A:

1. Berechne bei einer Eingabe x zunächst $f(x)$ mittels M_f.
2. Simuliere anschließend M_B auf $f(x)$.

Korrektheit: Folgt direkt aus der Definition von $\leq_p$.

Laufzeit: Es gilt

- $t_{M_B}(n) \leq p(n)$ für ein Polynom p,
- $t_{M_f}(n) \leq q(n)$ für ein Polynom q.

Laufzeit: Es gilt

- $t_{M_B}(n) \leq p(n)$ für ein Polynom p,
- $t_{M_f}(n) \leq q(n)$ für ein Polynom q.

Laufzeit von M_A bei einer Eingabe der Länge n:
$O(q(n) + p(q(n) + n))$.

Laufzeit: Es gilt

- $t_{M_B}(n) \leq p(n)$ für ein Polynom p,
- $t_{M_f}(n) \leq q(n)$ für ein Polynom q.

Laufzeit von M_A bei einer Eingabe der Länge n:
$O(q(n) + p(q(n) + n))$.

Die **Verschachtelung zweier Polynome** ist wieder ein Polynom. □

Vertex-Cover-Problem (VC)

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

Frage: Gibt es $V' \subseteq V$ mit $|V'| \leq k$, sodass jede Kante aus E zu mindestens einem Knoten aus V' inzident ist?

Vertex-Cover-Problem (VC)

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

Frage: Gibt es $V' \subseteq V$ mit $|V'| \leq k$, sodass jede Kante aus E zu mindestens einem Knoten aus V' inzident ist?

Theorem 4.12

Es gilt CLIQUE $\leq_p$ VC.

Vertex-Cover-Problem (VC)

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

Frage: Gibt es $V' \subseteq V$ mit $|V'| \leq k$, sodass jede Kante aus E zu mindestens einem Knoten aus V' inzident ist?

Theorem 4.12

Es gilt CLIQUE $\leq_p$ VC.

Beweis: Reduktion f mit $f((G, k)) = ((G', k'))$.

Vertex-Cover-Problem (VC)

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

Frage: Gibt es $V' \subseteq V$ mit $|V'| \leq k$, sodass jede Kante aus E zu mindestens einem Knoten aus V' inzident ist?

Theorem 4.12

Es gilt CLIQUE $\leq_p$ VC.

Beweis: Reduktion f mit $f((G, k)) = ((G', k'))$.

Sei $G' = (V', E')$ mit $V' = V$.

E' enthält genau die Kanten, die E nicht enthält, d. h.

$$E' = \{\{x, y\} \mid x, y \in V, x \neq y, \{x, y\} \notin E\}.$$

Außerdem sei $k' = n - k$ für $n = |V|$.

Vertex-Cover-Problem (VC)

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

Frage: Gibt es $V' \subseteq V$ mit $|V'| \leq k$, sodass jede Kante aus E zu mindestens einem Knoten aus V' inzident ist?

Theorem 4.12

Es gilt CLIQUE $\leq_p$ VC.

Beweis: Reduktion f mit $f((G, k)) = ((G', k'))$.

Sei $G' = (V', E')$ mit $V' = V$.

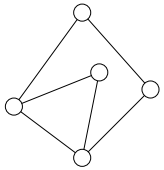
E' enthält genau die Kanten, die E nicht enthält, d. h.

$$E' = \{\{x, y\} \mid x, y \in V, x \neq y, \{x, y\} \notin E\}.$$

Außerdem sei $k' = n - k$ für $n = |V|$.

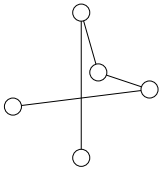
Reduktion f kann in **polynomieller Zeit berechnet** werden.

$G = (V, E)$



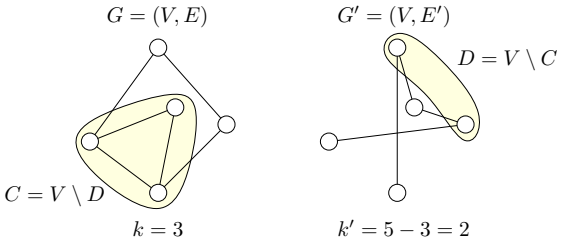
$k = 3$

$G' = (V, E')$



$k' = 5 - 3 = 2$

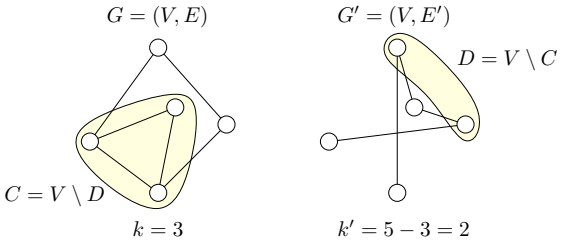
zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$



zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

„$\Rightarrow$“: Sei **C $\subseteq$ V eine k-Clique** in G.

Dann ist **D $=$ V $\setminus$ C ein Vertex Cover** in G' der Größe $k' = n - k = |V \setminus C|$.



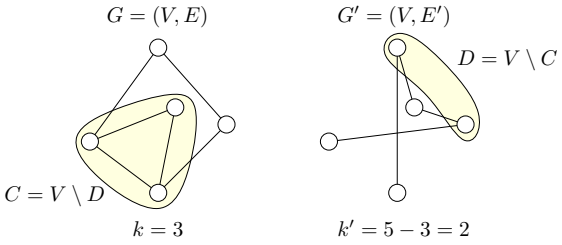
zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

„$\Rightarrow$“: Sei **$C \subseteq V$ eine k-Clique** in G.

Dann ist **$D = V \setminus C$ ein Vertex Cover** in G' der Größe $k' = n - k = |V \setminus C|$.

Annahme: D kein VC in G'.

$\Rightarrow$ Es existiert $\{x, y\} \in E'$ mit $x \notin D$ und $y \notin D$, also $x, y \in C$.



zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

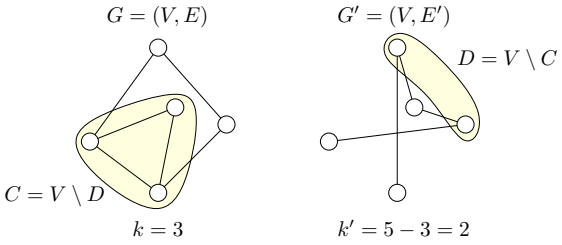
„$\Rightarrow$“: Sei **$C \subseteq V$ eine k-Clique** in G.

Dann ist **$D = V \setminus C$ ein Vertex Cover** in G' der Größe $k' = n - k = |V \setminus C|$.

Annahme: D kein VC in G'.

$\Rightarrow$ Es existiert $\{x, y\} \in E'$ mit $x \notin D$ und $y \notin D$, also $x, y \in C$.

$\Rightarrow \{x, y\} \in E$ (da C Clique in G)



zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

„$\Rightarrow$“: Sei **$C \subseteq V$ eine k-Clique** in G.

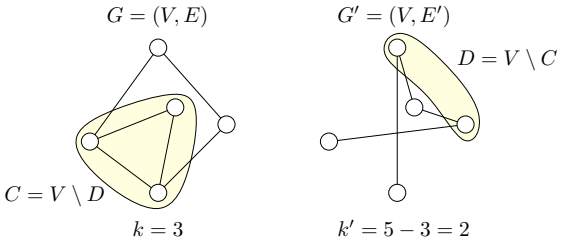
Dann ist **$D = V \setminus C$ ein Vertex Cover** in G' der Größe $k' = n - k = |V \setminus C|$.

Annahme: D kein VC in G'.

$\Rightarrow$ Es existiert $\{x, y\} \in E'$ mit $x \notin D$ und $y \notin D$, also $x, y \in C$.

$\Rightarrow \{x, y\} \in E$ (da C Clique in G)

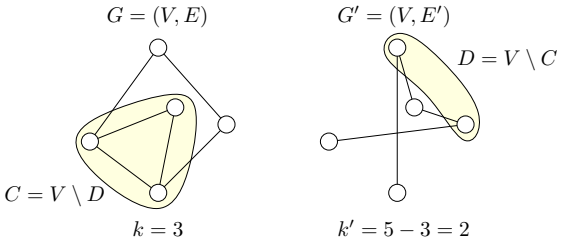
$\Rightarrow \{x, y\} \notin E'$



zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

„⇐“: Sei **$D \subseteq V' = V$ ein Vertex Cover der Größe k'** in G'.

Dann ist **$C = V \setminus D$ eine k-Clique** in G für $k = |V \setminus D| = n - k'$.



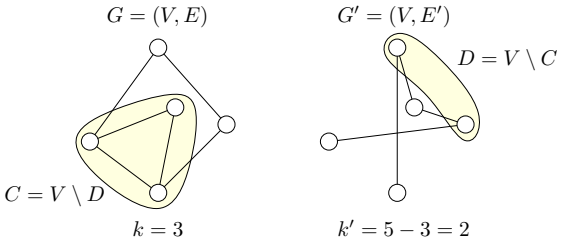
zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

„$\Leftarrow$“: Sei **$D \subseteq V' = V$ ein Vertex Cover der Größe k'** in G'.

Dann ist **$C = V \setminus D$ eine k-Clique** in G für $k = |V \setminus D| = n - k'$.

Annahme: C keine Clique in G.

$\Rightarrow$ Es existieren $x, y \in C$ mit $\{x, y\} \notin E$.



zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

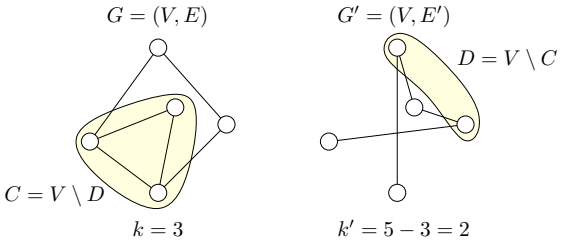
„⇐“: Sei **$D \subseteq V' = V$ ein Vertex Cover der Größe k'** in G'.

Dann ist **$C = V \setminus D$ eine k-Clique** in G für $k = |V \setminus D| = n - k'$.

Annahme: C keine Clique in G.

$\Rightarrow$ Es existieren $x, y \in C$ mit $\{x, y\} \notin E$.

$\Rightarrow \{x, y\} \in E'$.



zu zeigen: G enthält Clique der Größe $k \iff G'$ enthält VC der Größe $k' = n - k$

„⇐“: Sei **$D \subseteq V' = V$ ein Vertex Cover der Größe k'** in G'.

Dann ist **$C = V \setminus D$ eine k-Clique** in G für $k = |V \setminus D| = n - k'$.

Annahme: C keine Clique in G.

$\Rightarrow$ Es existieren $x, y \in C$ mit $\{x, y\} \notin E$.

$\Rightarrow \{x, y\} \in E'$.

$\Rightarrow x \in D$ oder $y \in D$ (da D Vertex Cover in G') □

Übung: $A \leq_p B$ und $B \leq_p C \Rightarrow A \leq_p C$.

Übung: $A \leq_p B$ und $B \leq_p C \Rightarrow A \leq_p C$.

Definition 4.14

Eine Sprache L heißt **NP-schwer**, wenn $L' \leq_p L$ für jede Sprache $L' \in \text{NP}$ gilt. Ist eine Sprache L NP-schwer und gilt zusätzlich $L \in \text{NP}$, so heißt L **NP-vollständig**.

Übung: $A \leq_p B$ und $B \leq_p C \Rightarrow A \leq_p C$.

Definition 4.14

Eine Sprache L heißt **NP-schwer**, wenn $L' \leq_p L$ für jede Sprache $L' \in \text{NP}$ gilt. Ist eine Sprache L NP-schwer und gilt zusätzlich $L \in \text{NP}$, so heißt L **NP-vollständig**.

Theorem 4.15

Gibt es eine NP-schwere Sprache $L \in \text{P}$, so gilt $\text{P} = \text{NP}$.

Übung: $A \leq_p B$ und $B \leq_p C \Rightarrow A \leq_p C$.

Definition 4.14

Eine Sprache L heißt **NP-schwer**, wenn $L' \leq_p L$ für jede Sprache $L' \in \mathrm{NP}$ gilt. Ist eine Sprache L NP-schwer und gilt zusätzlich $L \in \mathrm{NP}$, so heißt L **NP-vollständig**.

Theorem 4.15

Gibt es eine NP-schwere Sprache $L \in \mathrm{P}$, so gilt $\mathrm{P} = \mathrm{NP}$.

Beweis: Sei $L' \in \mathrm{NP}$ beliebig. Dann gilt $L' \leq_p L$. Wegen $L \in \mathrm{P}$ folgt daraus $L' \in \mathrm{P}$. □

Übung: $A \leq_p B$ und $B \leq_p C \Rightarrow A \leq_p C$.

Definition 4.14

Eine Sprache L heißt **NP-schwer**, wenn $L' \leq_p L$ für jede Sprache $L' \in \mathrm{NP}$ gilt. Ist eine Sprache L NP-schwer und gilt zusätzlich $L \in \mathrm{NP}$, so heißt L **NP-vollständig**.

Theorem 4.15

Gibt es eine NP-schwere Sprache $L \in \mathrm{P}$, so gilt $\mathrm{P} = \mathrm{NP}$.

Beweis: Sei $L' \in \mathrm{NP}$ beliebig. Dann gilt $L' \leq_p L$. Wegen $L \in \mathrm{P}$ folgt daraus $L' \in \mathrm{P}$. □

Korollar 4.16

Es sei L eine NP-vollständige Sprache. Dann gilt $L \in \mathrm{P}$ genau dann, wenn $\mathrm{P} = \mathrm{NP}$ gilt.