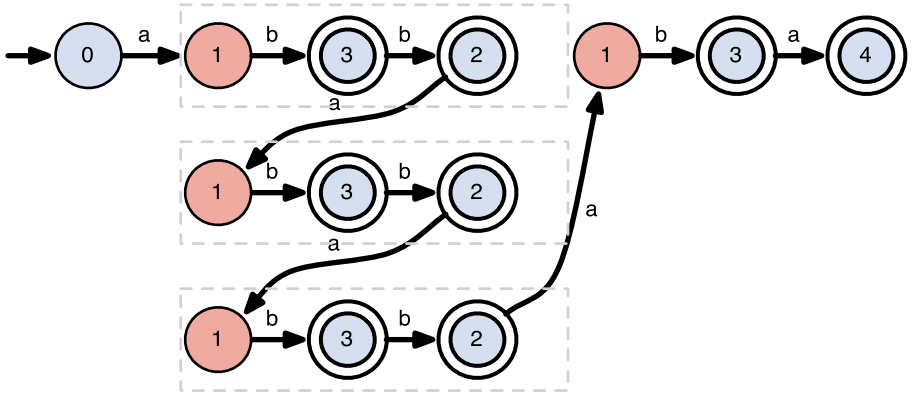
Berechenbarkeitstheorie

20. Vorlesung



Dr. Franziska Jahnke
Institut für Mathematische Logik und Grundlagenforschung
WWU Münster

Laufzeit einer Turingmaschine

Laufzeit einer Turingmaschine

- Laufzeit von M hängt von der Eingabe w ab

Laufzeit einer Turingmaschine

- Laufzeit von M hängt von der Eingabe w ab

$$T(w \in \Sigma^*) := \# \text{ Schritte für Lauf von } M(w)$$

(det. 1-Band TM)

Laufzeit einer Turingmaschine

- Laufzeit von M hängt von der Eingabe w ab

  $T(w \in \Sigma^*) := \#$ Schritte für Lauf von $M(w)$

  (det. 1-Band TM)

- Laufzeit wird meist in Bezug zur Eingabelänge gemessen

Laufzeit einer Turingmaschine

- Laufzeit von M hängt von der Eingabe w ab

$$T(w \in \Sigma^*) := \#\text{ Schritte für Lauf von } M(w)$$

(det. 1-Band TM)

- Laufzeit wird meist in Bezug zur Eingabelänge gemessen

$$t(n \in \mathbf{N}) := \max_{w \in \Sigma^n} T(w)$$

Laufzeit einer Turingmaschine

- Laufzeit von M hängt von der Eingabe w ab

$$T(w \in \Sigma^*) := \#\text{ Schritte für Lauf von } M(w)$$

(det. 1-Band TM)

- Laufzeit wird meist in Bezug zur Eingabelänge gemessen

$$t(n \in \mathbf{N}) := \max_{w \in \Sigma^n} T(w)$$

- genaue Angabe der Laufzeit oft schwierig (und unnötig)

Laufzeit einer Turingmaschine

- Laufzeit von M hängt von der Eingabe w ab

$$T(w \in \Sigma^*) := \#\text{ Schritte für Lauf von } M(w)$$

(det. 1-Band TM)

- Laufzeit wird meist in Bezug zur Eingabelänge gemessen

$$t(n \in \mathbf{N}) := \max_{w \in \Sigma^n} T(w)$$

- genaue Angabe der Laufzeit oft schwierig (und unnötig)
- wir greifen auf assymptotische Schranken zurück ($\rightarrow O$,Ω,Θ-Notation)

Laufzeit einer Turingmaschine

- Laufzeit von M hängt von der Eingabe w ab

$$T(w \in \Sigma^*) := \#\text{ Schritte für Lauf von } M(w)$$

(det. 1-Band TM)

- Laufzeit wird meist in Bezug zur Eingabelänge gemessen

$$t(n \in \mathbf{N}) := \max_{w \in \Sigma^n} T(w)$$

- genaue Angabe der Laufzeit oft schwierig (und unnötig)
- wir greifen auf assymptotische Schranken zurück ($\rightarrow O$,Ω,Θ-Notation)

Definition

Sei $f : \mathbf{N} \rightarrow \mathbf{N}$, dann bezeichnet

$$\text{TIME}(f(n)) = \{L \mid \exists \text{TM } M \text{ mit } t_M(n) = O(f(n))\}$$

die **Zeitkomplexitätsklasse** von $f(n)$.

Die Klasse P

Die Klasse P

Definition

Die Komplexitätsklasse P bezeichnet alle Probleme, die in Polynomialzeit von einer deterministischen TM entschieden werden können. Das heißt:

$$\mathsf{P} := \bigcup_{k=1}^{\infty} \mathsf{TIME}(n^k)$$

Die Klasse P

Definition

Die Komplexitätsklasse P bezeichnet alle Probleme, die in Polynomialzeit von einer deterministischen TM entschieden werden können. Das heißt:

$$\mathsf{P} := \bigcup_{k=1}^{\infty} \mathsf{TIME}(n^k)$$

- Bsp.: $\{0^k 1^k \mid k \geq 0\} \in \mathsf{P}$

Die Klasse P

Definition

Die Komplexitätsklasse P bezeichnet alle Probleme, die in Polynomialzeit von einer deterministischen TM entschieden werden können. Das heißt:

$$\mathsf{P} := \bigcup_{k=1}^{\infty} \mathsf{TIME}(n^k)$$

- Bsp.: $\{0^k 1^k \mid k \geq 0\} \in \mathsf{P}$
- **Anmerkung** Simulation eines TM-äquivalenten Berechnungsmodells durch ein anderes kostet i.A. einen polynomiellen Faktor

Die Klasse P

Definition

Die Komplexitätsklasse P bezeichnet alle Probleme, die in Polynomialzeit von einer deterministischen TM entschieden werden können. Das heißt:

$$\mathsf{P} := \bigcup_{k=1}^{\infty} \mathsf{TIME}(n^k)$$

- Bsp.: $\{0^k 1^k \mid k \geq 0\} \in \mathsf{P}$
- **Anmerkung** Simulation eines TM-äquivalenten Berechnungsmodells durch ein anderes kostet i.A. einen polynomiellen Faktor
  $\rightarrow$ Beschreibung von P relativ unabhängig vom Berechnungsmodell

Die Klasse P

Definition

Die Komplexitätsklasse P bezeichnet alle Probleme, die in Polynomialzeit von einer deterministischen TM entschieden werden können. Das heißt:

$$\mathsf{P} := \bigcup_{k=1}^{\infty} \mathsf{TIME}(n^k)$$

- Bsp.: $\{0^k 1^k \mid k \geq 0\} \in \mathsf{P}$
- **Anmerkung** Simulation eines TM-äquivalenten Berechnungsmodells durch ein anderes kostet i.A. einen polynomiellen Faktor
  $\rightarrow$ Beschreibung von P relativ unabhängig vom Berechnungsmodell
- **Konvention:** Probleme aus P sind effizient lösbar

Die Klasse P

Definition

Die Komplexitätsklasse P bezeichnet alle Probleme, die in Polynomialzeit von einer deterministischen TM entschieden werden können. Das heißt:

$$\mathsf{P} := \bigcup_{k=1}^{\infty} \mathsf{TIME}(n^k)$$

- Bsp.: $\{0^k 1^k \mid k \geq 0\} \in \mathsf{P}$
- **Anmerkung** Simulation eines TM-äquivalenten Berechnungsmodells durch ein anderes kostet i.A. einen polynomiellen Faktor
  $\rightarrow$ Beschreibung von P relativ unabhängig vom Berechnungsmodell
- **Konvention:** Probleme aus P sind effizient lösbar
- Probleme müssen sinnvoll kodiert sein

Die Klasse P

Definition

Die Komplexitätsklasse P bezeichnet alle Probleme, die in Polynomialzeit von einer deterministischen TM entschieden werden können. Das heißt:

$$\mathsf{P} := \bigcup_{k=1}^{\infty} \mathsf{TIME}(n^k)$$

- Bsp.: $\{0^k 1^k \mid k \geq 0\} \in \mathsf{P}$
- **Anmerkung** Simulation eines TM-äquivalenten Berechnungsmodells durch ein anderes kostet i.A. einen polynomiellen Faktor
  $\rightarrow$ Beschreibung von P relativ unabhängig vom Berechnungsmodell
- **Konvention:** Probleme aus P sind effizient lösbar
- Probleme müssen sinnvoll kodiert sein
  - $\langle k \rangle = \mathsf{bin}(k) \leftrightarrow \langle k \rangle = 1^k$

Die Klasse P

Definition

Die Komplexitätsklasse P bezeichnet alle Probleme, die in Polynomialzeit von einer deterministischen TM entschieden werden können. Das heißt:

$$\mathsf{P} := \bigcup_{k=1}^{\infty} \mathsf{TIME}(n^k)$$

- Bsp.: $\{0^k 1^k \mid k \geq 0\} \in \mathsf{P}$
- **Anmerkung** Simulation eines TM-äquivalenten Berechnungsmodells durch ein anderes kostet i.A. einen polynomiellen Faktor
  $\rightarrow$ Beschreibung von P relativ unabhängig vom Berechnungsmodell
- **Konvention:** Probleme aus P sind effizient lösbar
- Probleme müssen sinnvoll kodiert sein
  - $\langle k \rangle = \mathsf{bin}(k) \leftrightarrow \langle k \rangle = 1^k$
  - exponentieller Unterschied der Länge

Die Klasse P

Definition

Die Komplexitätsklasse P bezeichnet alle Probleme, die in Polynomialzeit von einer deterministischen TM entschieden werden können. Das heißt:

$$\mathsf{P} := \bigcup_{k=1}^{\infty} \mathsf{TIME}(n^k)$$

- Bsp.: $\{0^k 1^k \mid k \geq 0\} \in \mathsf{P}$
- **Anmerkung** Simulation eines TM-äquivalenten Berechnungsmodells durch ein anderes kostet i.A. einen polynomiellen Faktor
  $\rightarrow$ Beschreibung von P relativ unabhängig vom Berechnungsmodell
- **Konvention:** Probleme aus P sind effizient lösbar
- Probleme müssen sinnvoll kodiert sein
  - $\langle k \rangle = \mathsf{bin}(k) \leftrightarrow \langle k \rangle = 1^k$
  - exponentieller Unterschied der Länge
  - Laufzeit wird relativ zur Eingabelänge gemessen!

Die Klasse NP

- NP beinhaltet die Probleme, deren Lösung sich leicht verifizieren lässt

Die Klasse NP

- NP beinhaltet die Probleme, deren Lösung sich leicht verifizieren lässt
- **Def.:** Ein **Verifizierer für eine Sprache** L ist eine TM für die gilt: $L = \{w \mid \exists z \in \Sigma^* : \langle w, z \rangle \text{ wird von } V \text{ akzeptiert}\}$

Die Klasse NP

- NP beinhaltet die Probleme, deren Lösung sich leicht verifizieren lässt
- **Def.:** Ein **Verifizierer für eine Sprache** L ist eine TM für die gilt: $L = \{w \mid \exists z \in \Sigma^* : \langle w, z \rangle$ wird von V akzeptiert$\}$

z heißt Zeuge oder Zertifikat

Die Klasse NP

- NP beinhaltet die Probleme, deren Lösung sich leicht verifizieren lässt
- **Def.:** Ein **Verifizierer für eine Sprache** L ist eine TM für die gilt: $L = \{w \mid \exists z \in \Sigma^* : \langle w, z \rangle$ wird von V akzeptiert$\}$

z heißt Zeuge oder Zertifikat

Laufzeit von V wird bzgl. $\langle w, z \rangle$ gemessen

Die Klasse NP

- NP beinhaltet die Probleme, deren Lösung sich leicht verifizieren lässt
- **Def.:** Ein **Verifizierer für eine Sprache** L ist eine TM für die gilt:
  $L = \{w \mid \exists z \in \Sigma^* : \langle w, z \rangle$ wird von V akzeptiert$\}$

z heißt Zeuge oder Zertifikat

Laufzeit von V wird bzgl. $\langle w, z \rangle$ gemessen

- **Def.:** Ein Verifizierer V für L heißt **polynomiell**, gdw.
  (1) $\exists k$, sodass $L = \{w \mid \exists z \in \Sigma^* : |z| \leq |w|^k$ und $\langle w, z \rangle \in L(V)\}$
  (2) V hat polynomielle Laufzeit

Die Klasse NP

- NP beinhaltet die Probleme, deren Lösung sich leicht verifizieren lässt
- **Def.:** Ein **Verifizierer für eine Sprache** L ist eine TM für die gilt:
  $L = \{w \mid \exists z \in \Sigma^* : \langle w, z \rangle$ wird von V akzeptiert$\}$

z heißt Zeuge oder Zertifikat

Laufzeit von V wird bzgl. $\langle w, z \rangle$ gemessen

- **Def.:** Ein Verifizierer V für L heißt **polynomiell**, gdw.
  (1) $\exists k$, sodass $L = \{w \mid \exists z \in \Sigma^* : |z| \leq |w|^k$ und $\langle w, z \rangle \in L(V)\}$
  (2) V hat polynomielle Laufzeit

Definition

Die Komplexitätsklasse NP umfasst alle Sprachen, für die es einen polynomiellen Verifizierer gibt.

Die Klasse NP

- NP beinhaltet die Probleme, deren Lösung sich leicht verifizieren lässt
- **Def.:** Ein **Verifizierer für eine Sprache** L ist eine TM für die gilt: $L = \{w \mid \exists z \in \Sigma^* : \langle w, z \rangle$ wird von V akzeptiert$\}$

z heißt Zeuge oder Zertifikat

Laufzeit von V wird bzgl. $\langle w, z \rangle$ gemessen

- **Def.:** Ein Verifizierer V für L heißt **polynomiell**, gdw.
  (1) $\exists k$, sodass $L = \{w \mid \exists z \in \Sigma^* : |z| \leq |w|^k$ und $\langle w, z \rangle \in L(V)\}$
  (2) V hat polynomielle Laufzeit

Definition

Die Komplexitätsklasse NP umfasst alle Sprachen, für die es einen polynomiellen Verifizierer gibt.

- Wenn L aus P dann ist der polynomielle Entscheider für L auch ein polynomieller Verifizierer für L (er ignoriert den Zeugen)

Die Klasse NP

- NP beinhaltet die Probleme, deren Lösung sich leicht verifizieren lässt
- **Def.:** Ein **Verifizierer für eine Sprache** L ist eine TM für die gilt: $L = \{w \mid \exists z \in \Sigma^* : \langle w, z \rangle$ wird von V akzeptiert$\}$

z heißt Zeuge oder Zertifikat

Laufzeit von V wird bzgl. $\langle w, z \rangle$ gemessen

- **Def.:** Ein Verifizierer V für L heißt **polynomiell**, gdw.
  (1) $\exists k$, sodass $L = \{w \mid \exists z \in \Sigma^* : |z| \leq |w|^k$ und $\langle w, z \rangle \in L(V)\}$
  (2) V hat polynomielle Laufzeit

Definition

Die Komplexitätsklasse NP umfasst alle Sprachen, für die es einen polynomiellen Verifizierer gibt.

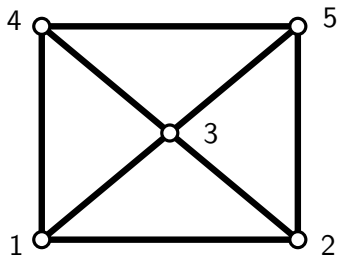
- Wenn L aus P dann ist der polynomielle Entscheider für L auch ein polynomieller Verifizierer für L (er ignoriert den Zeugen)
- Es folgt: P $\subseteq$ NP

- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$

- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

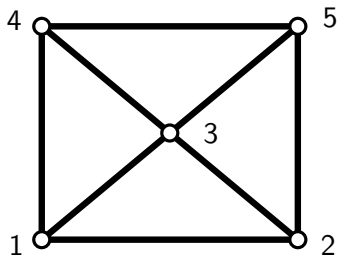
- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

Bsp.1



- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

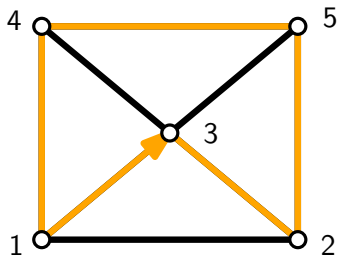
Bsp.1



- Graph hat einen Hamiltonkreis

- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

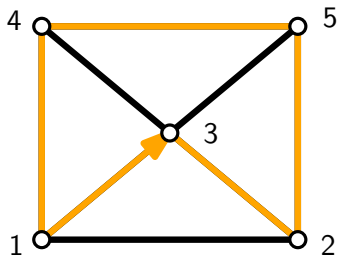
Bsp.1



- Graph hat einen Hamiltonkreis

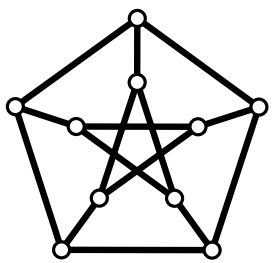
- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

Bsp.1



- Graph hat einen Hamiltonkreis

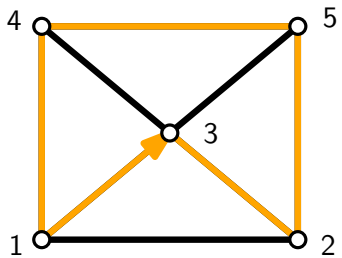
Bsp.2



- Graph hat keinen Kreis der Länge ≤ 4

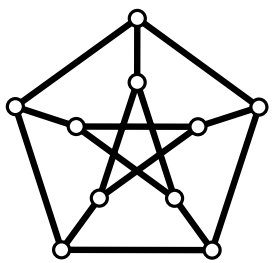
- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

Bsp.1

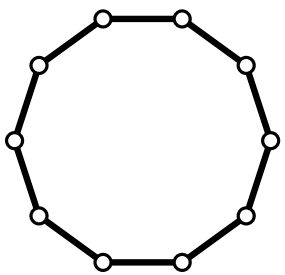


- Graph hat einen Hamiltonkreis

Bsp.2

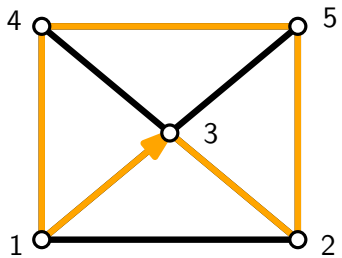


- Graph hat keinen Kreis der Länge ≤ 4
- Angenommen es gibt Hamiltonkreis



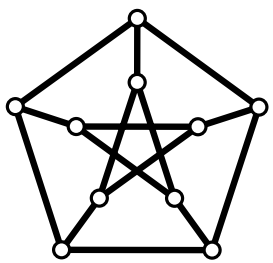
- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

Bsp.1

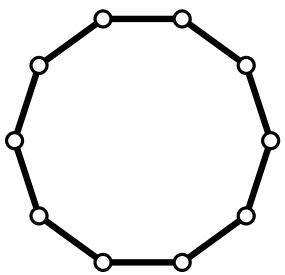


- Graph hat einen Hamiltonkreis

Bsp.2

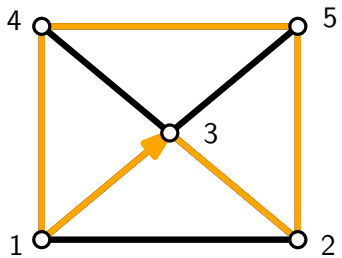


- Graph hat keinen Kreis der Länge ≤ 4
- Angenommen es gibt Hamiltonkreis
- nicht-kreuzende "Diagonalen" erzeugen Kreis der Länge ≤ 4



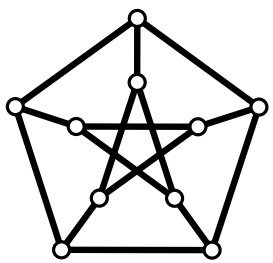
- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

Bsp.1

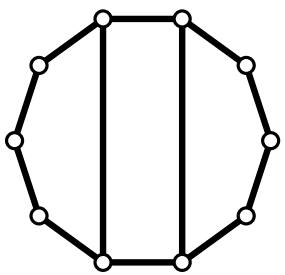


- Graph hat einen Hamiltonkreis

Bsp.2

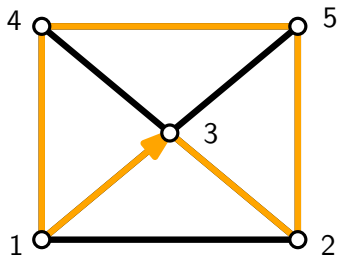


- Graph hat keinen Kreis der Länge ≤ 4
- Angenommen es gibt Hamiltonkreis
- nicht-kreuzende "Diagonalen" erzeugen Kreis der Länge ≤ 4



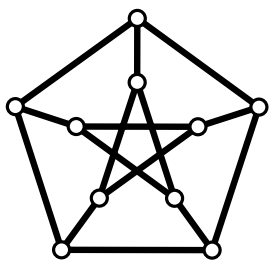
- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

Bsp.1

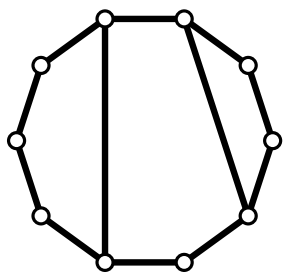


- Graph hat einen Hamiltonkreis

Bsp.2

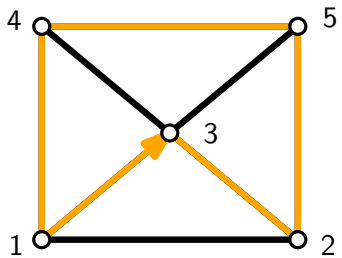


- Graph hat keinen Kreis der Länge ≤ 4
- Angenommen es gibt Hamiltonkreis
- nicht-kreuzende "Diagonalen" erzeugen Kreis der Länge ≤ 4



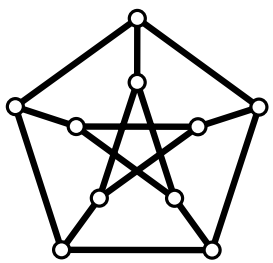
- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

Bsp.1

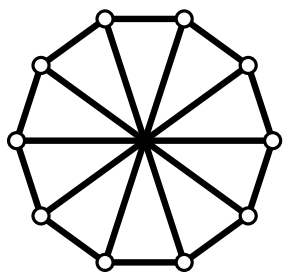


- Graph hat einen Hamiltonkreis

Bsp.2

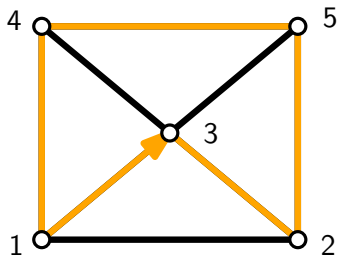


- Graph hat keinen Kreis der Länge ≤ 4
- Angenommen es gibt Hamiltonkreis
- nicht-kreuzende "Diagonalen" erzeugen Kreis der Länge ≤ 4
- verbleibenen Kanten kreuzen paarweise



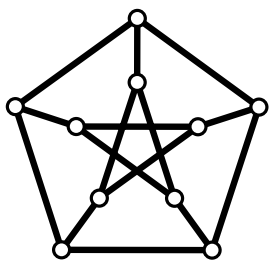
- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

Bsp.1

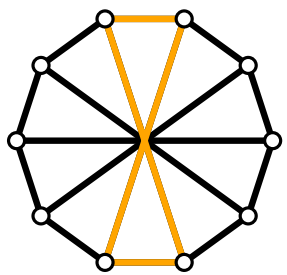


- Graph hat einen Hamiltonkreis

Bsp.2

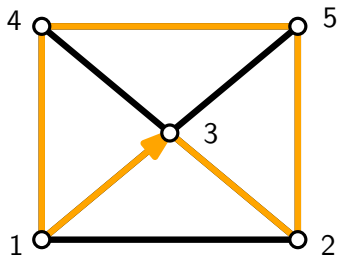


- Graph hat keinen Kreis der Länge ≤ 4
- Angenommen es gibt Hamiltonkreis
- nicht-kreuzende "Diagonalen" erzeugen Kreis der Länge ≤ 4
- verbleibenen Kanten kreuzen paarweise
- einzige Möglichkeit hat Kreis der Länge 4



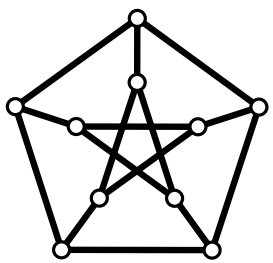
- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

Bsp.1

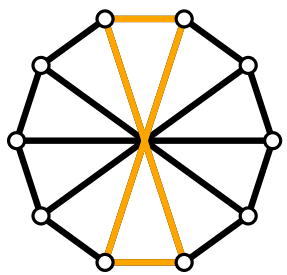


- Graph hat einen Hamiltonkreis

Bsp.2



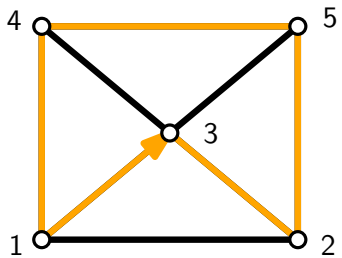
- Graph hat keinen Kreis der Länge ≤ 4
- Angenommen es gibt Hamiltonkreis
- nicht-kreuzende "Diagonalen" erzeugen Kreis der Länge ≤ 4
- verbleibenen Kanten kreuzen paarweise
- einzige Möglichkeit hat Kreis der Länge 4



- HC $\in$ NP

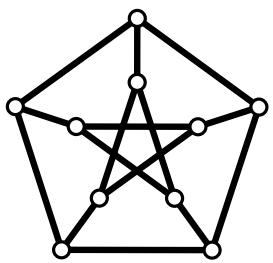
- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

Bsp.1

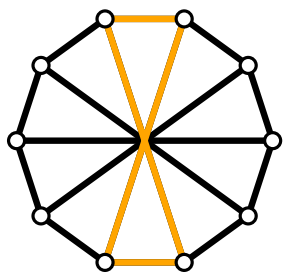


- Graph hat einen Hamiltonkreis
  Zeuge: (3,2,5,4,1)

Bsp.2



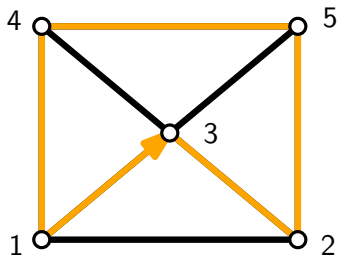
- Graph hat keinen Kreis der Länge ≤ 4
- Angenommen es gibt Hamiltonkreis
- nicht-kreuzende "Diagonalen" erzeugen Kreis der Länge ≤ 4
- verbleibenen Kanten kreuzen paarweise
- einzige Möglichkeit hat Kreis der Länge 4



- HC $\in$ NP
- Zeuge: Permutation der Knoten die Hamiltonkreis beschreibt

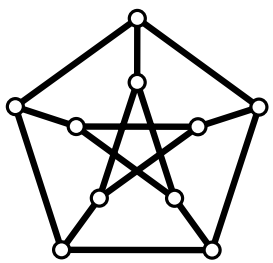
- **Bsp.:** HC $= \{\langle G \rangle \mid G$ ist Graph mit Hamiltonkreis$\}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

Bsp.1

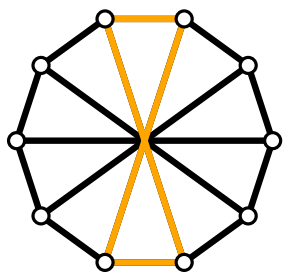


- Graph hat einen Hamiltonkreis
  Zeuge: (3,2,5,4,1)

Bsp.2



- Graph hat keinen Kreis der Länge ≤ 4
- Angenommen es gibt Hamiltonkreis
- nicht-kreuzende "Diagonalen" erzeugen Kreis der Länge ≤ 4
- verbleibenen Kanten kreuzen paarweise
- einzige Möglichkeit hat Kreis der Länge 4



- HC $\in$ NP
- Zeuge: Permutation der Knoten die Hamiltonkreis beschreibt
- Verifikation in polynomieller Zeit möglich, Größe des Zeugen ist Polynom

- Verifizierer V für Hamiltonkreis

$V(\langle G, z \rangle)$

1. Prüfe ob G eine korrekte Kodierung eines Graphen ist
2. Prüfe ob z eine Permutation der Knoten von G ist
3. - Bearbeite Knoten in der zyklischen Reihenfolge der Permutation
   - Sind je zwei aufeinander folgende Knoten durch eine Kante verbunden – akzeptiere
   - Ansonsten verwerfe

- Verifizierer V für Hamiltonkreis

$V(\langle G, z \rangle)$

1. Prüfe ob G eine korrekte Kodierung eines Graphen ist
2. Prüfe ob z eine Permutation der Knoten von G ist
3. - Bearbeite Knoten in der zyklischen Reihenfolge der Permutation
   - Sind je zwei aufeinander folgende Knoten durch eine Kante verbunden – akzeptiere
   - Ansonsten verwerfe

- V ist polynomieller Verifizierer, was belegt das HC $\in$ NP

- Verifizierer V für Hamiltonkreis

$V(\langle G, z \rangle)$

1. Prüfe ob G eine korrekte Kodierung eines Graphen ist
2. Prüfe ob z eine Permutation der Knoten von G ist
3. - Bearbeite Knoten in der zyklischen Reihenfolge der Permutation
   - Sind je zwei aufeinander folgende Knoten durch eine Kante verbunden – akzeptiere
   - Ansonsten verwerfe

- V ist polynomieller Verifizierer, was belegt das HC $\in$ NP
- Anmerkung: Gibt es für folgende Sprache einen Verifizierer?
  $\overline{\text{HC}} = \{\langle G \rangle \mid G \text{ hat keinen Hamiltonkreis}\}$

- Verifizierer V für Hamiltonkreis

$V(\langle G, z \rangle)$
1. Prüfe ob G eine korrekte Kodierung eines Graphen ist
2. Prüfe ob z eine Permutation der Knoten von G ist
3. - Bearbeite Knoten in der zyklischen Reihenfolge der Permutation
   - Sind je zwei aufeinander folgende Knoten durch eine Kante verbunden – akzeptiere
   - Ansonsten verwerfe

- V ist polynomieller Verifizierer, was belegt das HC $\in$ NP
- Anmerkung: Gibt es für folgende Sprache einen Verifizierer? $\overline{\text{HC}} = \{\langle G \rangle \mid G \text{ hat keinen Hamiltonkreis}\}$
- Im Beispiel war es aufwendiger zu bezeugen, dass es keinen Hamiltonkreis gibt
- Existenz eines polynomiellen Verifizierers zu $\overline{\text{HC}}$ ist ein offenes Problem

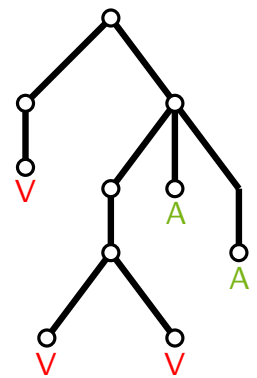
Alternative Definition von NP

Alternative Definition von NP

- Definition der Laufzeit einer nichtdeterministischen Turingmaschine

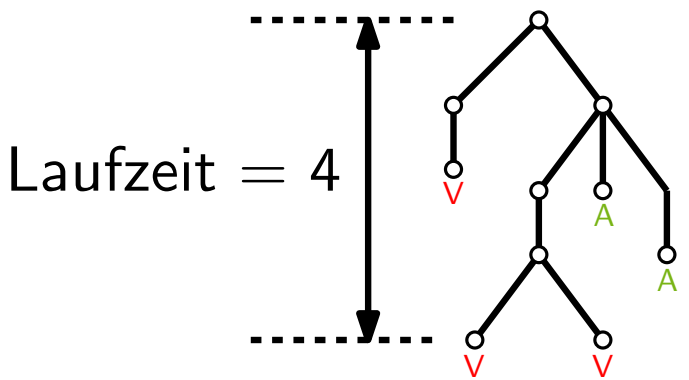
Alternative Definition von NP

- Definition der Laufzeit einer nichtdeterministischen Turingmaschine
- Für NTM M und Eingabe w existiert ein Berechnungsbaum



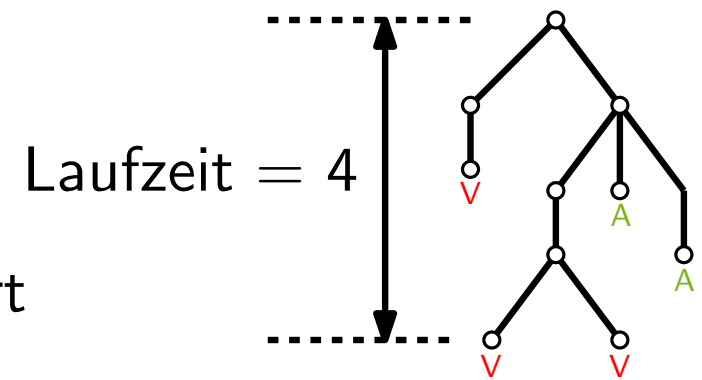
Alternative Definition von NP

- Definition der Laufzeit einer nichtdeterministischen Turingmaschine
- Für NTM M und Eingabe w existiert ein Berechnungsbaum
- Laufzeit von $M(w)$ ist Höhe des Berechnungsbaumes von $M(w)$



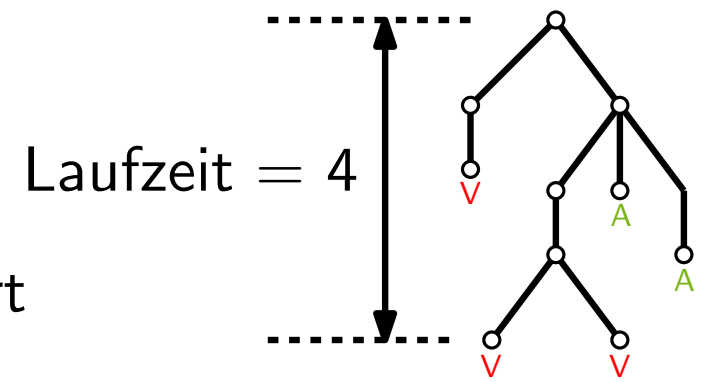
Alternative Definition von NP

- Definition der Laufzeit einer nichtdeterministischen Turingmaschine
- Für NTM M und Eingabe w existiert ein Berechnungsbaum
- Laufzeit von $M(w)$ ist Höhe des Berechnungsbaumes von $M(w)$
- Es ist egal ob, das Blatt zu einer akz. oder verw. Konfiguration gehört



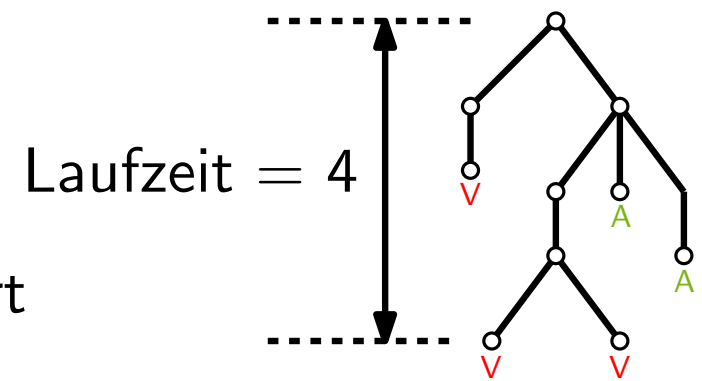
Alternative Definition von NP

- Definition der Laufzeit einer nichtdeterministischen Turingmaschine
- Für NTM M und Eingabe w existiert ein Berechnungsbaum
- Laufzeit von $M(w)$ ist Höhe des Berechnungsbaumes von $M(w)$
- Es ist egal ob, das Blatt zu einer akz. oder verw. Konfiguration gehört
- $T(w) =$ Laufzeit von $M(w)$



Alternative Definition von NP

- Definition der Laufzeit einer nichtdeterministischen Turingmaschine
- Für NTM M und Eingabe w existiert ein Berechnungsbaum
- Laufzeit von $M(w)$ ist Höhe des Berechnungsbaumes von $M(w)$
- Es ist egal ob, das Blatt zu einer akz. oder verw. Konfiguration gehört
- $T(w) =$ Laufzeit von $M(w)$
- $t(n) = \max_{w \in \Sigma^n} T(w)$



Definition

Sei $f : \mathbf{N} \to \mathbf{N}$, dann bezeichnet

$$\text{NTIME}(f(n)) = \{L \mid \exists \text{NTM } N \text{ mit } t_N(n) = O(f(n))\}$$

die **nichtdeterminstische Zeitkomplexitätsklasse** von $f(n)$.

Satz 37

$$\mathsf{NP} = \bigcup_{i=1}^{\infty} \mathsf{NTIME}(n^k)$$

Satz 37

$$\text{NP} = \bigcup_{i=1}^{\infty} \text{NTIME}(n^k)$$

Beweis

$\Rightarrow$-**Richtung:** $L \in \text{NP} \Rightarrow L \in \text{NTIME}(n^k)$ für ein k

Satz 37

$$\text{NP} = \bigcup_{i=1}^{\infty} \text{NTIME}(n^k)$$

Beweis

$\Rightarrow$-Richtung: $L \in \text{NP} \Rightarrow L \in \text{NTIME}(n^k)$ für ein k

- wenn L aus NP, dann existiert ein polynomieller Verifizierer V für L (d.h. V benutzt Zeugen der Länge $\leq |w|^k$, für ein $k \in \mathbf{N}$)

Satz 37

$$\text{NP} = \bigcup_{i=1}^{\infty} \text{NTIME}(n^k)$$

Beweis

$\Rightarrow$-**Richtung:** $L \in \text{NP} \Rightarrow L \in \text{NTIME}(n^k)$ für ein k

- wenn L aus NP, dann existiert ein polynomieller Verifizierer V für L (d.h. V benutzt Zeugen der Länge $\leq |w|^k$, für ein $k \in \mathbf{N}$)
- betrachte die NTM $N(w)$

$N(w)$

1. Erzeuge ein beliebiges Wort $z \in \Sigma^*$ mit Länge $\leq n^k$
2. Simuliere $V(\langle w, z \rangle)$ und gebe das Ergebnis aus

Satz 37

$$\text{NP} = \bigcup_{i=1}^{\infty} \text{NTIME}(n^k)$$

Beweis

$\Rightarrow$-**Richtung:** $L \in \text{NP} \Rightarrow L \in \text{NTIME}(n^k)$ für ein k

- wenn L aus NP, dann existiert ein polynomieller Verifizierer V für L (d.h. V benutzt Zeugen der Länge $\leq |w|^k$, für ein $k \in \mathbf{N}$)
- betrachte die NTM $N(w)$

$N(w)$

1. Erzeuge ein beliebiges Wort $z \in \Sigma^*$ mit Länge $\leq n^k$
2. Simuliere $V(\langle w, z \rangle)$ und gebe das Ergebnis aus

- Laufzeit von N ist polynomiell

Satz 37

$$\text{NP} = \bigcup_{i=1}^{\infty} \text{NTIME}(n^k)$$

Beweis

$\Rightarrow$-Richtung: $L \in \text{NP} \Rightarrow L \in \text{NTIME}(n^k)$ für ein k

- wenn L aus NP, dann existiert ein polynomieller Verifizierer V für L (d.h. V benutzt Zeugen der Länge $\leq |w|^k$, für ein $k \in \mathbf{N}$)
- betrachte die NTM $N(w)$

> $N(w)$
>
> 1. Erzeuge ein beliebiges Wort $z \in \Sigma^*$ mit Länge $\leq n^k$
> 2. Simuliere $V(\langle w, z \rangle)$ und gebe das Ergebnis aus

- Laufzeit von N ist polynomiell
- wir kennen nicht unbedingt das k, aber ein k existiert, und somit existiert die entsprechende NTM N

Satz 37

$$\text{NP} = \bigcup_{i=1}^{\infty} \text{NTIME}(n^k)$$

Beweis

$\Rightarrow$-Richtung: $L \in \text{NP} \Rightarrow L \in \text{NTIME}(n^k)$ für ein k

- wenn L aus NP, dann existiert ein polynomieller Verifizierer V für L (d.h. V benutzt Zeugen der Länge $\leq |w|^k$, für ein $k \in \mathbf{N}$)
- betrachte die NTM $N(w)$

> $N(w)$
>
> 1. Erzeuge ein beliebiges Wort $z \in \Sigma^*$ mit Länge $\leq n^k$
> 2. Simuliere $V(\langle w, z \rangle)$ und gebe das Ergebnis aus

- Laufzeit von N ist polynomiell
- wir kennen nicht unbedingt das k, aber ein k existiert, und somit existiert die entsprechende NTM N

  $w \in L \iff \exists z\colon (|z| \leq |w|^k)\colon V(\langle w, z \rangle)$ akz. $\iff N(w)$ akz.

$\Leftarrow$-Richtung: $L \in \text{NTIME}(n^k)$ für ein $k \Rightarrow L \in \text{NP}$

$\Leftarrow$-Richtung: $L \in \text{NTIME}(n^k)$ für ein $k \Rightarrow L \in \text{NP}$

- Sei N die NTM, die L in nichtdet. Polynomialzeit entscheidet

$\Leftarrow$-Richtung: $L \in \text{NTIME}(n^k)$ für ein $k \Rightarrow L \in \text{NP}$

- Sei N die NTM, die L in nichtdet. Polynomialzeit entscheidet
- $w \in L$, gdw. es gibt akz. Konfiguration im Berechnungsbaum zu $N(w)$

$\Leftarrow$-Richtung: $L \in \text{NTIME}(n^k)$ für ein $k \Rightarrow L \in \text{NP}$

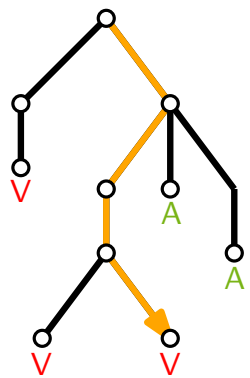
- Sei N die NTM, die L in nichtdet. Polynomialzeit entscheidet
- $w \in L$, gdw. es gibt akz. Konfiguration im Berechnungsbaum zu $N(w)$
- Zeuge kodiert die Adresse eines Knotens im Berechnungsbaum

$\Leftarrow$-Richtung: $L \in \text{NTIME}(n^k)$ für ein $k \Rightarrow L \in \text{NP}$

- Sei N die NTM, die L in nichtdet. Polynomialzeit entscheidet
- $w \in L$, gdw. es gibt akz. Konfiguration im Berechnungsbaum zu $N(w)$
- Zeuge kodiert die Adresse eines Knotens im Berechnungsbaum
- $z = a_1 \# a_2 \# \cdots \# a_k$, und a_i ist die Nummer der Alternative der $i$ten Folgekonfiguration

$\Leftarrow$-Richtung: $L \in \text{NTIME}(n^k)$ für ein $k \Rightarrow L \in \text{NP}$

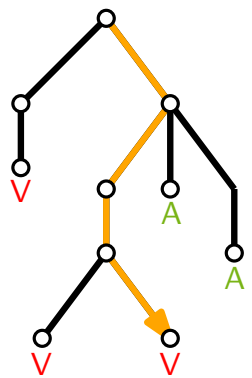
- Sei N die NTM, die L in nichtdet. Polynomialzeit entscheidet
- $w \in L$, gdw. es gibt akz. Konfiguration im Berechnungsbaum zu $N(w)$
- Zeuge kodiert die Adresse eines Knotens im Berechnungsbaum
- $z = a_1 \# a_2 \# \cdots \# a_k$, und a_i ist die Nummer der Alternative der $i$ten Folgekonfiguration



Adr.: 2#1#1#2

$\Leftarrow$-Richtung: $L \in \text{NTIME}(n^k)$ für ein $k \Rightarrow L \in \text{NP}$

- Sei N die NTM, die L in nichtdet. Polynomialzeit entscheidet
- $w \in L$, gdw. es gibt akz. Konfiguration im Berechnungsbaum zu $N(w)$
- Zeuge kodiert die Adresse eines Knotens im Berechnungsbaum
- $z = a_1 \# a_2 \# \cdots \# a_k$, und a_i ist die Nummer der Alternative der $i$ten Folgekonfiguration



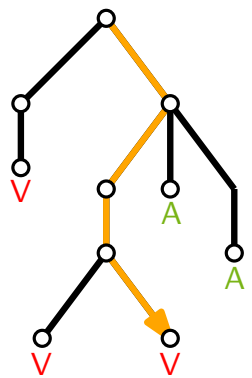
Adr.: 2#1#1#2

$V(\langle w, z \rangle)$

1. Suche im Berechnungsbaum Blatt mit Adresse z, falls es existiert
2. Akzeptiere, gdw. gefundes Blatt akz. Konfiguration

$\Leftarrow$-Richtung: $L \in \text{NTIME}(n^k)$ für ein $k \Rightarrow L \in \text{NP}$

- Sei N die NTM, die L in nichtdet. Polynomialzeit entscheidet
- $w \in L$, gdw. es gibt akz. Konfiguration im Berechnungsbaum zu $N(w)$
- Zeuge kodiert die Adresse eines Knotens im Berechnungsbaum
- $z = a_1 \# a_2 \# \cdots \# a_k$, und a_i ist die Nummer der Alternative der $i$ten Folgekonfiguration



Adr.: 2#1#1#2

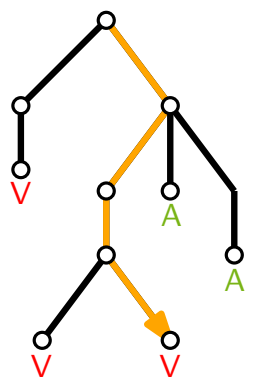
$V(\langle w, z \rangle)$

1. Suche im Berechnungsbaum Blatt mit Adresse z, falls es existiert
2. Akzeptiere, gdw. gefundes Blatt akz. Konfiguration

- $V(\langle w, z \rangle)$ arbeitet in Polynomialzeit

$\Leftarrow$-Richtung: $L \in \text{NTIME}(n^k)$ für ein $k \Rightarrow L \in \text{NP}$

- Sei N die NTM, die L in nichtdet. Polynomialzeit entscheidet
- $w \in L$, gdw. es gibt akz. Konfiguration im Berechnungsbaum zu $N(w)$
- Zeuge kodiert die Adresse eines Knotens im Berechnungsbaum
- $z = a_1 \# a_2 \# \cdots \# a_k$, und a_i ist die Nummer der Alternative der $i$ten Folgekonfiguration



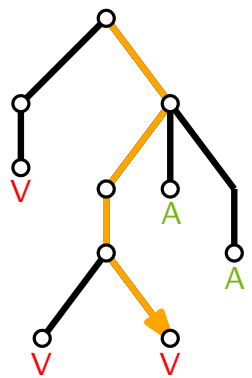
Adr.: 2#1#1#2

$V(\langle w, z \rangle)$
1. Suche im Berechnungsbaum Blatt mit Adresse z, falls es existiert
2. Akzeptiere, gdw. gefundes Blatt akz. Konfiguration

- $V(\langle w, z \rangle)$ arbeitet in Polynomialzeit
- $w \in L \iff \exists$ akz. Berechnungspfad der Länge $\leq |w|^k$
  $\iff \exists z(|z| \leq |w|^k)\colon V(\langle w, z \rangle)$ akz.

$\Leftarrow$-Richtung: $L \in \text{NTIME}(n^k)$ für ein $k \Rightarrow L \in \text{NP}$

- Sei N die NTM, die L in nichtdet. Polynomialzeit entscheidet
- $w \in L$, gdw. es gibt akz. Konfiguration im Berechnungsbaum zu $N(w)$
- Zeuge kodiert die Adresse eines Knotens im Berechnungsbaum
- $z = a_1 \# a_2 \# \cdots \# a_k$, und a_i ist die Nummer der Alternative der $i$ten Folgekonfiguration



Adr.: 2#1#1#2

$V(\langle w, z \rangle)$
1. Suche im Berechnungsbaum Blatt mit Adresse z, falls es existiert
2. Akzeptiere, gdw. gefundes Blatt akz. Konfiguration

- $V(\langle w, z \rangle)$ arbeitet in Polynomialzeit
- $w \in L \iff \exists$ akz. Berechnungspfad der Länge $\leq |w|^k$
  $\iff \exists z(|z| \leq |w|^k)\colon V(\langle w, z \rangle)$ akz.
- V ist polynomieller Verifizierer für L

□

Beispiele NP

SAT (Erfüllbarkeit)

Eingabe: Boolsche Formel ϕ in den Variablen x_i

Frage: Gibt es eine ϕ erfüllende Belegung der Variablen x_i?

SAT (Erfüllbarkeit)

Eingabe: Boolsche Formel ϕ in den Variablen x_i

Frage: Gibt es eine ϕ erfüllende Belegung der Variablen x_i?

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

SAT (Erfüllbarkeit)

Eingabe: Boolsche Formel ϕ in den Variablen x_i

Frage: Gibt es eine ϕ erfüllende Belegung der Variablen x_i?

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

$\rightarrow$ erfüllbar mit z.B. $x_1 = 1, x_2 = 1$ und $x_3 = 1$

SAT (Erfüllbarkeit)

Eingabe: Boolsche Formel ϕ in den Variablen x_i

Frage: Gibt es eine ϕ erfüllende Belegung der Variablen x_i?

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

$\rightarrow$ erfüllbar mit z.B. $x_1 = 1, x_2 = 1$ und $x_3 = 1$

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

SAT (Erfüllbarkeit)

Eingabe: Boolsche Formel ϕ in den Variablen x_i

Frage: Gibt es eine ϕ erfüllende Belegung der Variablen x_i?

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

$\rightarrow$ erfüllbar mit z.B. $x_1 = 1, x_2 = 1$ und $x_3 = 1$

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

$\rightarrow$ nicht erfüllbar (Wahrheitstabelle ausfüllen)

SAT (Erfüllbarkeit)

Eingabe: Boolsche Formel ϕ in den Variablen x_i

Frage: Gibt es eine ϕ erfüllende Belegung der Variablen x_i?

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

$\rightarrow$ erfüllbar mit z.B. $x_1 = 1, x_2 = 1$ und $x_3 = 1$

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

$\rightarrow$ nicht erfüllbar (Wahrheitstabelle ausfüllen)

- Zeuge ist der Vektor der Variablenbelegung

SAT (Erfüllbarkeit)

Eingabe: Boolsche Formel ϕ in den Variablen x_i

Frage: Gibt es eine ϕ erfüllende Belegung der Variablen x_i?

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

$\rightarrow$ erfüllbar mit z.B. $x_1 = 1, x_2 = 1$ und $x_3 = 1$

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

$\rightarrow$ nicht erfüllbar (Wahrheitstabelle ausfüllen)

- Zeuge ist der Vektor der Variablenbelegung

Verifizierer zu SAT

1. Ersetze alle Variablen durch ihre Belegung
2. Suche eine atomare Verbindung mit $\wedge, \vee, \neg$ und ersetze sie (zum Beispiel ersetze $1 \vee 0$ durch 1)
3. Wiederhole bis Ausdruck ausgewertet und akzeptiere, wenn Ergebnis 1

CLIQUE

Eingabe: Graph G und natürliche Zahl k

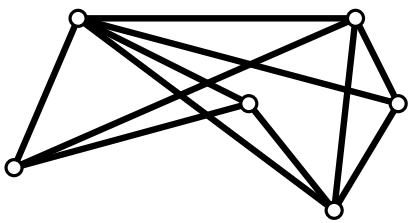
Frage: Hat G eine k-Clique (d.h., K_k als Teilgraph)?

CLIQUE

Eingabe: Graph G und natürliche Zahl k

Frage: Hat G eine k-Clique (d.h., K_k als Teilgraph)?

Bsp.



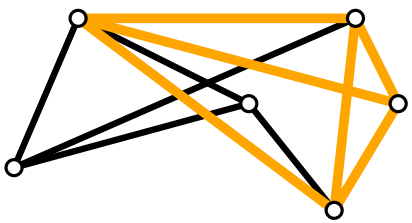
$k = 4$

CLIQUE

Eingabe: Graph G und natürliche Zahl k

Frage: Hat G eine k-Clique (d.h., K_k als Teilgraph)?

Bsp.



$k = 4$

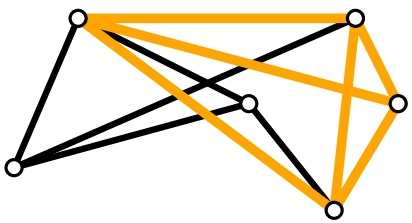
Graph besitzt 4-Clique

CLIQUE

Eingabe: Graph G und natürliche Zahl k

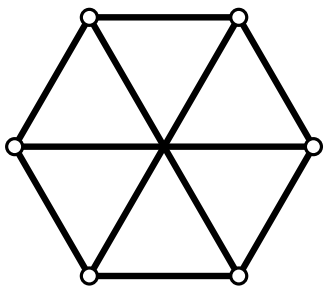
Frage: Hat G eine k-Clique (d.h., K_k als Teilgraph)?

Bsp.



$k = 4$

Graph besitzt 4-Clique



$k = 3$

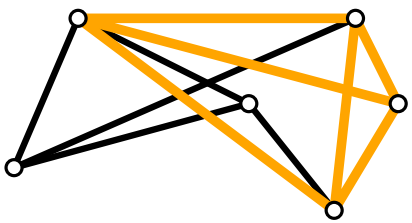
Graph besitzt keine 3-Clique

CLIQUE

Eingabe: Graph G und natürliche Zahl k

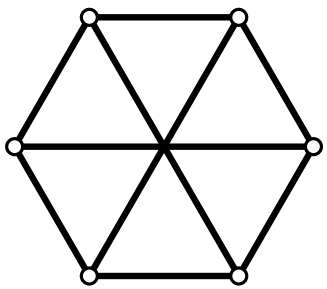
Frage: Hat G eine k-Clique (d.h., K_k als Teilgraph)?

Bsp.



$k = 4$

Graph besitzt 4-Clique



$k = 3$

Graph besitzt keine 3-Clique

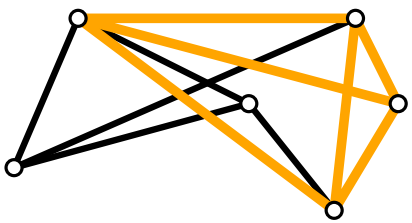
- Zeuge ist Auswahl der Knoten der Clique

CLIQUE

Eingabe: Graph G und natürliche Zahl k

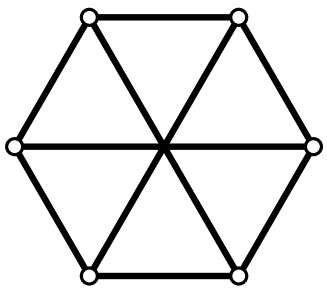
Frage: Hat G eine k-Clique (d.h., K_k als Teilgraph)?

Bsp.



$k = 4$

Graph besitzt 4-Clique



$k = 3$

Graph besitzt keine 3-Clique

- Zeuge ist Auswahl der Knoten der Clique

Verifizierer zu CLIQUE

1. Prüfe für jedes Paar von ausgewählten Knoten (des Zeugen), ob zwischen diesen Knoten eine Kante existiert
2. Ist jedes solche Paar durch eine Kante verbunden, akzeptiere

SUBSET-SUM

Eingabe: Multimenge $S = \{X_1, X_2, \ldots, X_n\}, X_i \in \mathbf{N}$ und $B \in \mathbf{N}$

Frage: Existiert $S' \subseteq S$ mit $\sum_{X \in S'} X = B$

SUBSET-SUM

Eingabe: Multimenge $S = \{X_1, X_2, \ldots, X_n\}, X_i \in \mathbf{N}$ und $B \in \mathbf{N}$

Frage: Existiert $S' \subseteq S$ mit $\sum_{X \in S'} X = B$

Bsp.

$S = \{2, 3, 4, 8, 19\}$ $B = 25$

SUBSET-SUM

Eingabe: Multimenge $S = \{X_1, X_2, \ldots, X_n\}, X_i \in \mathbf{N}$ und $B \in \mathbf{N}$

Frage: Existiert $S' \subseteq S$ mit $\sum_{X \in S'} X = B$

Bsp.

$S = \{2, 3, 4, 8, 19\}$ $B = 25$ $2 + 4 + 19 = 25$, also ja

SUBSET-SUM

Eingabe: Multimenge $S = \{X_1, X_2, \ldots, X_n\}, X_i \in \mathbf{N}$ und $B \in \mathbf{N}$

Frage: Existiert $S' \subseteq S$ mit $\sum_{X \in S'} X = B$

Bsp.

$S = \{2, 3, 4, 8, 19\}$ $B = 25$ $2 + 4 + 19 = 25$, also ja

$S = \{1, 1, 4, 8, 19\}$ $B = 30$

SUBSET-SUM

Eingabe: Multimenge $S = \{X_1, X_2, \ldots, X_n\}, X_i \in \mathbf{N}$ und $B \in \mathbf{N}$

Frage: Existiert $S' \subseteq S$ mit $\sum_{X \in S'} X = B$

Bsp.

$S = \{2, 3, 4, 8, 19\}$ $B = 25$ $2 + 4 + 19 = 25$, also ja

$S = \{1, 1, 4, 8, 19\}$ $B = 30$ nicht möglich

SUBSET-SUM

Eingabe: Multimenge $S = \{X_1, X_2, \ldots, X_n\}, X_i \in \mathbf{N}$ und $B \in \mathbf{N}$

Frage: Existiert $S' \subseteq S$ mit $\sum_{X \in S'} X = B$

Bsp.

$S = \{2, 3, 4, 8, 19\}$	$B = 25$	$2 + 4 + 19 = 25$, also ja
$S = \{1, 1, 4, 8, 19\}$	$B = 30$	nicht möglich

- Zeuge ist Auswahl S'

SUBSET-SUM

Eingabe: Multimenge $S = \{X_1, X_2, \ldots, X_n\}, X_i \in \mathbf{N}$ und $B \in \mathbf{N}$
Frage: Existiert $S' \subseteq S$ mit $\sum_{X \in S'} X = B$

Bsp.

$S = \{2, 3, 4, 8, 19\}$ $B = 25$ $2 + 4 + 19 = 25$, also ja

$S = \{1, 1, 4, 8, 19\}$ $B = 30$ nicht möglich

- Zeuge ist Auswahl S'

Verifizierer zu SUBSET-SUM

1. Prüfe ob $S' \subseteq S$
2. Prüfe, ob $\sum_{X \in S'} X = B$
3. Wenn beides erfolgreich, akzeptiere

Satz 38

$$\text{NP} \subseteq \text{EXPTIME} = \bigcup_{k=1}^{\infty} \text{TIME}\left(2^{(n^k)}\right)$$

Satz 38

$$\text{NP} \subseteq \text{EXPTIME} = \bigcup_{k=1}^{\infty} \text{TIME}\left(2^{(n^k)}\right)$$

Beweis

- wir müssen zeigen, dass jede Sprache aus NP in TIME$(2^{(n^k)})$, für ein $k \in \mathbf{N}$

Satz 38

$$\text{NP} \subseteq \text{EXPTIME} = \bigcup_{k=1}^{\infty} \text{TIME}\left(2^{(n^k)}\right)$$

Beweis

- wir müssen zeigen, dass jede Sprache aus NP in TIME$(2^{(n^k)})$, für ein $k \in \mathbf{N}$
- sei $L \in$ NP mit polynomiellen Verifizierer V

Satz 38

$$\text{NP} \subseteq \text{EXPTIME} = \bigcup_{k=1}^{\infty} \text{TIME}\left(2^{(n^k)}\right)$$

Beweis

- wir müssen zeigen, dass jede Sprache aus NP in TIME$(2^{(n^k)})$, für ein $k \in \mathbf{N}$
- sei $L \in$ NP mit polynomiellen Verifizierer V
- wir konstruieren Entscheider M für L wie folgt

Satz 38

$$\text{NP} \subseteq \text{EXPTIME} = \bigcup_{k=1}^{\infty} \text{TIME}\left(2^{(n^k)}\right)$$

Beweis

- wir müssen zeigen, dass jede Sprache aus NP in TIME$(2^{(n^k)})$, für ein $k \in \mathbf{N}$
- sei $L \in$ NP mit polynomiellen Verifizierer V
- wir konstruieren Entscheider M für L wie folgt

$M(w)$

1. Zähle alle möglichen Zeugen auf (d.h. alle Worte aus Σ^*)
   $\rightarrow$ für jeden Zeugen z simuliere $V(\langle w, z \rangle)$
2. akzeptiere, sobald V akzeptiert, ansonsten prüfe nächstes z

Satz 38

$$\text{NP} \subseteq \text{EXPTIME} = \bigcup_{k=1}^{\infty} \text{TIME}\left(2^{(n^k)}\right)$$

Beweis

- wir müssen zeigen, dass jede Sprache aus NP in TIME$(2^{(n^k)})$, für ein $k \in \mathbf{N}$
- sei $L \in$ NP mit polynomiellen Verifizierer V
- wir konstruieren Entscheider M für L wie folgt

$M(w)$
1. Zähle alle möglichen Zeugen auf (d.h. alle Worte aus Σ^*)
   $\rightarrow$ für jeden Zeugen z simuliere $V(\langle w, z \rangle)$
2. akzeptiere, sobald V akzeptiert, ansonsten prüfe nächstes z

- es gibt $2^{O(p(n))}$ Zeugen, wobei p ein Polynom

Satz 38

$$\text{NP} \subseteq \text{EXPTIME} = \bigcup_{k=1}^{\infty} \text{TIME}\left(2^{(n^k)}\right)$$

Beweis

- wir müssen zeigen, dass jede Sprache aus NP in TIME$(2^{(n^k)})$, für ein $k \in \mathbf{N}$
- sei $L \in$ NP mit polynomiellen Verifizierer V
- wir konstruieren Entscheider M für L wie folgt

$M(w)$
1. Zähle alle möglichen Zeugen auf (d.h. alle Worte aus Σ^*)
   $\rightarrow$ für jeden Zeugen z simuliere $V(\langle w, z \rangle)$
2. akzeptiere, sobald V akzeptiert, ansonsten prüfe nächstes z

- es gibt $2^{O(p(n))}$ Zeugen, wobei p ein Polynom
- pro Zeugen z benötigt Simulation $p'(|\langle w, z \rangle|)$ Zeit, wobei p' Polynom

Satz 38

$$\text{NP} \subseteq \text{EXPTIME} = \bigcup_{k=1}^{\infty} \text{TIME}\left(2^{(n^k)}\right)$$

Beweis

- wir müssen zeigen, dass jede Sprache aus NP in TIME$(2^{(n^k)})$, für ein $k \in \mathbf{N}$
- sei $L \in$ NP mit polynomiellen Verifizierer V
- wir konstruieren Entscheider M für L wie folgt

$M(w)$
1. Zähle alle möglichen Zeugen auf (d.h. alle Worte aus Σ^*)
   $\rightarrow$ für jeden Zeugen z simuliere $V(\langle w, z\rangle)$
2. akzeptiere, sobald V akzeptiert, ansonsten prüfe nächstes z

- es gibt $2^{O(p(n))}$ Zeugen, wobei p ein Polynom
- pro Zeugen z benötigt Simulation $p'(|\langle w, z\rangle|)$ Zeit, wobei p' Polynom
- L ist aus TIME$(2^{O(p(n))}p'(n + p(n))) \subset$ EXPTIME

□

Definition

Seien $L_1 \subseteq \Sigma^*$ und $L_2 \subseteq \Sigma'^*$ Sprachen. Die Funktion f heißt **polyzeit Reduktion von L_1 auf L_2**, gdw.

1. f ist berechenbar und total,
2. $\forall w \in \Sigma^* : w \in L_1 \iff f(w) \in L_2$,
3. f wird durch eine TM berechnet deren Laufzeit ein Polynom in der Eingabelänge ist

Definition

Seien $L_1 \subseteq \Sigma^*$ und $L_2 \subseteq \Sigma'^*$ Sprachen. Die Funktion f heißt **polyzeit Reduktion von L_1 auf L_2**, gdw.

1. f ist berechenbar und total,
2. $\forall w \in \Sigma^* : w \in L_1 \iff f(w) \in L_2$,
3. f wird durch eine TM berechnet deren Laufzeit ein Polynom in der Eingabelänge ist

Definition

Eine Sprache L_1 ist **polyzeit reduzierbar** auf L_2, gdw. es eine polyzeit Reduktion von L_1 auf L_2 gibt.
Schreibweise: $L_1 \leq_p L_2$

Definition

Seien $L_1 \subseteq \Sigma^*$ und $L_2 \subseteq \Sigma'^*$ Sprachen. Die Funktion f heißt **polyzeit Reduktion von L_1 auf L_2**, gdw.

1. f ist berechenbar und total,
2. $\forall w \in \Sigma^* : w \in L_1 \iff f(w) \in L_2$,
3. f wird durch eine TM berechnet deren Laufzeit ein Polynom in der Eingabelänge ist

Definition

Eine Sprache L_1 ist **polyzeit reduzierbar** auf L_2, gdw. es eine polyzeit Reduktion von L_1 auf L_2 gibt.
Schreibweise: $L_1 \leq_p L_2$

- modifizierter Reduktionsbegriff erlaubt es uns Probleme zu "übersetzen" unter Beibehaltung von polynomiellen Laufzeiten ihrer Entscheider

Satz 39

Wenn $B \in \mathsf{P}$ und $A \leq_p B$, dann auch $A \in \mathsf{P}$.

Satz 39

Wenn $B \in \mathsf{P}$ und $A \leq_p B$, dann auch $A \in \mathsf{P}$.

Beweis

- Sei M der polynomieller Entscheider für B und f eine polyzeit Reduktion von A auf B.

Satz 39

Wenn $B \in \mathsf{P}$ und $A \leq_p B$, dann auch $A \in \mathsf{P}$.

Beweis

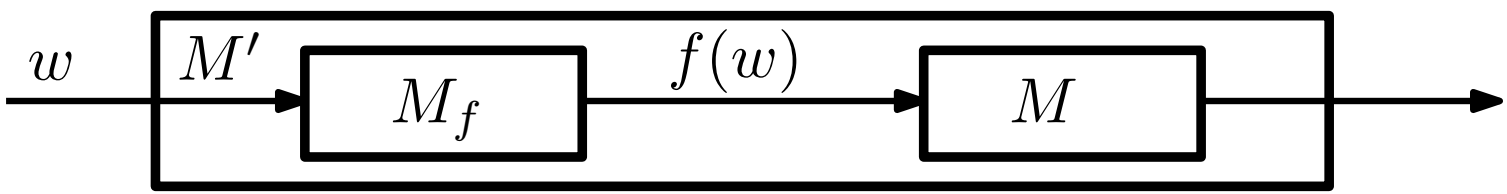
- Sei M der polynomieller Entscheider für B und f eine polyzeit Reduktion von A auf B.
- Sei M_f TM, die f in polyzeit berechnet.

Satz 39

Wenn $B \in \mathsf{P}$ und $A \leq_p B$, dann auch $A \in \mathsf{P}$.

Beweis

- Sei M der polynomieller Entscheider für B und f eine polyzeit Reduktion von A auf B.
- Sei M_f TM, die f in polyzeit berechnet.
- Konstuiere TM M', als Hintereinanderschaltung von M und M_f

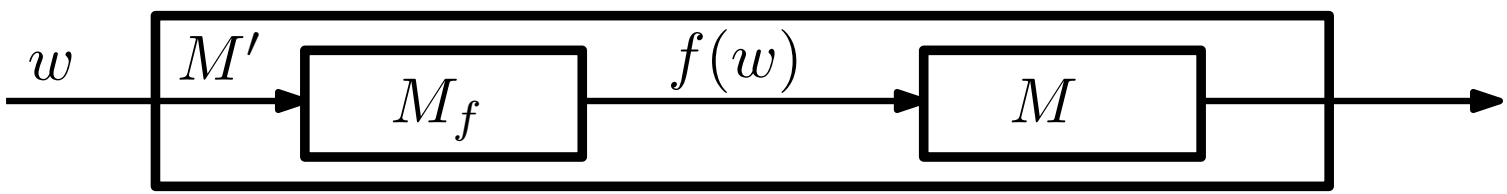


Satz 39

Wenn $B \in \mathsf{P}$ und $A \leq_p B$, dann auch $A \in \mathsf{P}$.

Beweis

- Sei M der polynomieller Entscheider für B und f eine polyzeit Reduktion von A auf B.
- Sei M_f TM, die f in polyzeit berechnet.
- Konstuiere TM M', als Hintereinanderschaltung von M und M_f



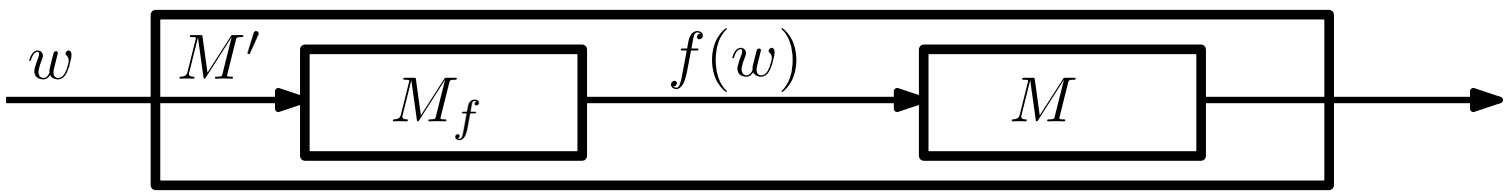
$M'(w)$ akz. $\iff$ $M(f(w))$ akz. $\iff$ $f(w) \in B$ $\iff$ $w \in A$

Satz 39

Wenn $B \in \mathrm{P}$ und $A \leq_p B$, dann auch $A \in \mathrm{P}$.

Beweis

- Sei M der polynomieller Entscheider für B und f eine polyzeit Reduktion von A auf B.
- Sei M_f TM, die f in polyzeit berechnet.
- Konstuiere TM M', als Hintereinanderschaltung von M und M_f



$M'(w)$ akz. $\iff$ $M(f(w))$ akz. $\iff$ $f(w) \in B$ $\iff$ $w \in A$

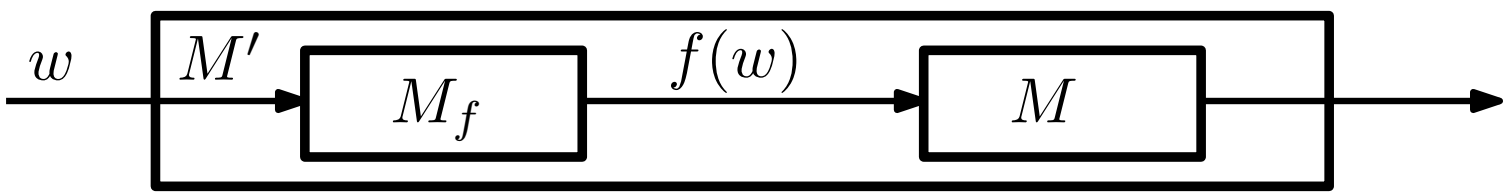
- Laufzeit von $M'(w)$ enspricht $t_f(|w|) + t_M(|f(w)|)$

Satz 39

Wenn $B \in \mathrm{P}$ und $A \leq_p B$, dann auch $A \in \mathrm{P}$.

Beweis

- Sei M der polynomieller Entscheider für B und f eine polyzeit Reduktion von A auf B.
- Sei M_f TM, die f in polyzeit berechnet.
- Konstuiere TM M', als Hintereinanderschaltung von M und M_f



$M'(w)$ akz. $\iff$ $M(f(w))$ akz. $\iff$ $f(w) \in B$ $\iff$ $w \in A$

- Laufzeit von $M'(w)$ enspricht $t_f(|w|) + t_M(|f(w)|)$

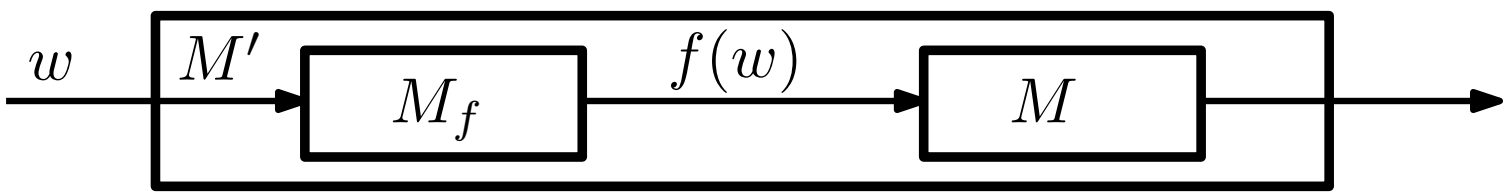
Laufzeit M_f Laufzeit M

Satz 39

Wenn $B \in \mathrm{P}$ und $A \leq_p B$, dann auch $A \in \mathrm{P}$.

Beweis

- Sei M der polynomieller Entscheider für B und f eine polyzeit Reduktion von A auf B.
- Sei M_f TM, die f in polyzeit berechnet.
- Konstuiere TM M', als Hintereinanderschaltung von M und M_f



$M'(w)$ akz. $\iff$ $M(f(w))$ akz. $\iff$ $f(w) \in B$ $\iff$ $w \in A$

- Laufzeit von $M'(w)$ enspricht $t_f(|w|) + t_M(|f(w)|)$

Laufzeit M_f Laufzeit M

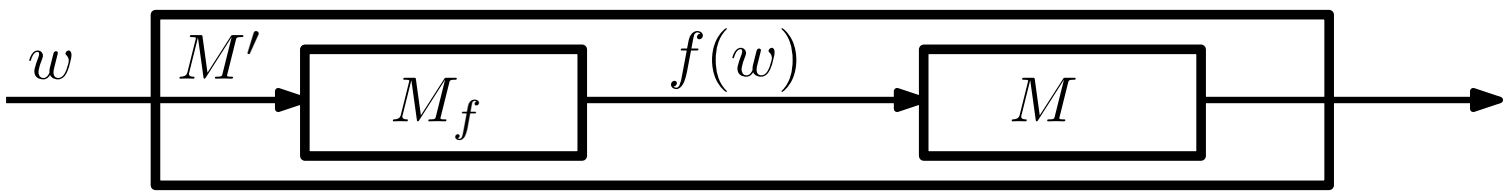
- $|f(w)|$ ist beschränkt durch ein Polynom p und t_M durch ein Polynom p' $\rightarrow t_M(|f(w)|) \leq p'(p(n))$

Satz 39

Wenn $B \in \mathrm{P}$ und $A \leq_p B$, dann auch $A \in \mathrm{P}$.

Beweis

- Sei M der polynomieller Entscheider für B und f eine polyzeit Reduktion von A auf B.
- Sei M_f TM, die f in polyzeit berechnet.
- Konstuiere TM M', als Hintereinanderschaltung von M und M_f



$M'(w)$ akz. $\iff$ $M(f(w))$ akz. $\iff$ $f(w) \in B$ $\iff$ $w \in A$

- Laufzeit von $M'(w)$ enspricht $t_f(|w|) + t_M(|f(w)|)$

  Laufzeit M_f Laufzeit M

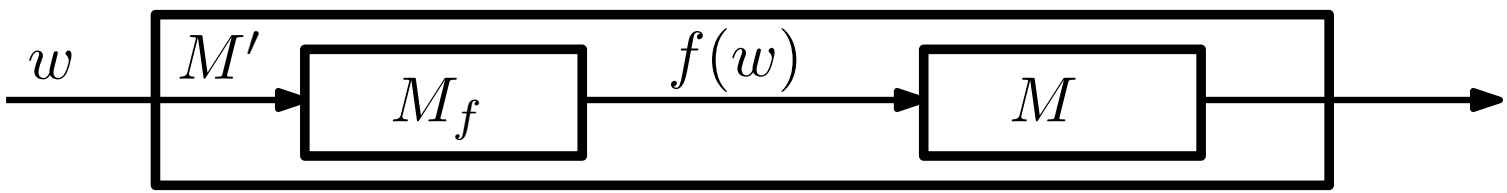
- $|f(w)|$ ist beschränkt durch ein Polynom p und t_M durch ein Polynom p' $\rightarrow t_M(|f(w)|) \leq p'(p(n))$ ← Polynom

Satz 39

Wenn $B \in \mathrm{P}$ und $A \leq_p B$, dann auch $A \in \mathrm{P}$.

Beweis

- Sei M der polynomieller Entscheider für B und f eine polyzeit Reduktion von A auf B.
- Sei M_f TM, die f in polyzeit berechnet.
- Konstuiere TM M', als Hintereinanderschaltung von M und M_f



$$M'(w) \text{ akz.} \iff M(f(w)) \text{ akz.} \iff f(w) \in B \iff w \in A$$

- Laufzeit von $M'(w)$ enspricht $t_f(|w|) + t_M(|f(w)|)$
  - $t_f(|w|)$: Laufzeit M_f
  - $t_M(|f(w)|)$: Laufzeit M
- $|f(w)|$ ist beschränkt durch ein Polynom p und t_M durch ein Polynom p' $\rightarrow t_M(|f(w)|) \leq p'(p(n))$ ← Polynom
- Laufzeit von M ist beschränkt durch Polynom $\rightarrow A \in \mathrm{P}$

□

Definition

Eine Sprache L heißt NP- **vollständig**, gdw.

1. $L \in$ NP,
2. $\forall L' \in$ NP gilt $L' \leq_p L$.

Definition

Eine Sprache L heißt NP- **vollständig**, gdw.

1. $L \in$ NP,
2. $\forall L' \in$ NP gilt $L' \leq_p L$.

- NP-vollständige Sprachen kann man als die schwersten Sprachen in der Klasse NP verstehen

Definition

Eine Sprache L heißt NP- **vollständig**, gdw.

1. $L \in$ NP,
2. $\forall L' \in$ NP gilt $L' \leq_p L$.

- NP-vollständige Sprachen kann man als die schwersten Sprachen in der Klasse NP verstehen

$$\text{NPC} := \{L \mid \ L \text{ ist NP-vollständig}\}$$

Definition

Eine Sprache L heißt NP- **vollständig**, gdw.

1. $L \in$ NP,
2. $\forall L' \in$ NP gilt $L' \leq_p L$.

- NP-vollständige Sprachen kann man als die schwersten Sprachen in der Klasse NP verstehen

$$\text{NPC} := \{L \mid \ L \text{ ist NP-vollständig}\}$$

- Gilt in der obigen Definition nur der Punkt 2., heißt L NP-**schwer**.

Definition

Eine Sprache L heißt NP- **vollständig**, gdw.

1. $L \in$ NP,
2. $\forall L' \in$ NP gilt $L' \leq_p L$.

- NP-vollständige Sprachen kann man als die schwersten Sprachen in der Klasse NP verstehen

$$\text{NPC} := \{L \mid \ L \text{ ist NP-vollständig}\}$$

- Gilt in der obigen Definition nur der Punkt 2., heißt L NP-**schwer**.

Satz 40

Wenn $B \in$ P und $B \in$ NPC, dann gilt P $=$ NP.

Definition

Eine Sprache L heißt NP- **vollständig**, gdw.

1. $L \in$ NP,
2. $\forall L' \in$ NP gilt $L' \leq_p L$.

- NP-vollständige Sprachen kann man als die schwersten Sprachen in der Klasse NP verstehen

$$\text{NPC} := \{L \mid \ L \text{ ist NP-vollständig}\}$$

- Gilt in der obigen Definition nur der Punkt 2., heißt L NP-**schwer**.

Satz 40

Wenn $B \in$ P und $B \in$ NPC, dann gilt P $=$ NP.

Beweis

- Für jedes $A \in$ NP gilt nach Definition $A \leq_p B$

Definition

Eine Sprache L heißt NP- **vollständig**, gdw.

1. $L \in$ NP,
2. $\forall L' \in$ NP gilt $L' \leq_p L$.

- NP-vollständige Sprachen kann man als die schwersten Sprachen in der Klasse NP verstehen

$$\text{NPC} := \{L \mid \ L \text{ ist NP-vollständig}\}$$

- Gilt in der obigen Definition nur der Punkt 2., heißt L NP-**schwer**.

Satz 40

Wenn $B \in$ P und $B \in$ NPC, dann gilt P $=$ NP.

Beweis

- Für jedes $A \in$ NP gilt nach Definition $A \leq_p B$
- Aus Satz 39 folgt dann aber, dass $A \in$ P

Definition

Eine Sprache L heißt NP- **vollständig**, gdw.

1. $L \in$ NP,
2. $\forall L' \in$ NP gilt $L' \leq_p L$.

- NP-vollständige Sprachen kann man als die schwersten Sprachen in der Klasse NP verstehen

$$\text{NPC} := \{L \mid \ L \text{ ist NP-vollständig}\}$$

- Gilt in der obigen Definition nur der Punkt 2., heißt L NP-**schwer**.

Satz 40

Wenn $B \in$ P und $B \in$ NPC, dann gilt P = NP.

Beweis

- Für jedes $A \in$ NP gilt nach Definition $A \leq_p B$
- Aus Satz 39 folgt dann aber, dass $A \in$ P
- Also gilt NP $\subseteq$ P, und P $\subseteq$ NP wurde bereits gezeigt

Definition

Eine Sprache L heißt NP- **vollständig**, gdw.

1. $L \in$ NP,
2. $\forall L' \in$ NP gilt $L' \leq_p L$.

- NP-vollständige Sprachen kann man als die schwersten Sprachen in der Klasse NP verstehen

$$\text{NPC} := \{L \mid \ L \text{ ist NP-vollständig}\}$$

- Gilt in der obigen Definition nur der Punkt 2., heißt L NP-**schwer**.

Satz 40

Wenn $B \in$ P und $B \in$ NPC, dann gilt P $=$ NP.

Beweis

- Für jedes $A \in$ NP gilt nach Definition $A \leq_p B$
- Aus Satz 39 folgt dann aber, dass $A \in$ P
- Also gilt NP $\subseteq$ P, und P $\subseteq$ NP wurde bereits gezeigt $\longrightarrow$ P $=$ NP

□