Script generated by TTT

Title: seidl: Theoretische_Informatik (02.07.2012)

Date: Mon Jul 02 10:15:45 CEST 2012

Duration: 89:07 min

Pages: 93

Polynomielle und exponentielle Komplexität
Taktfrequenz: 1GHz

Zeit	Problemgröße n 10	20	30	40	50	60
n	.01 ms	.02 ms	.03 ms	.04 ms	.05 ms	.06 ms
n^2	.1 ms	.4 ms	.9 ms	1.6 ms	2.5 ms	3.6 ms
n^5	100ms	0.003s	0.02s	0.1 s	0.3 s	0.7 s
2^n	1 ms	0.001s	1 s	18 m	13 T	36 J
3^n	59 ms	3 s	2 T	385J	$2 \cdot 10^7$ J	10^{12} J

ms=Mikrosek., s=Sek., m=Minute, T=Tag, J=Jahr

Polynomielle und exponentielle Komplexität
Taktfrequenz: 1GHz

Zeit	Problemgröße n 10	20	30	40	50	60
n	.01 ms	.02 ms	.03 ms	.04 ms	.05 ms	.06 ms
n^2	.1 ms	.4 ms	.9 ms	1.6 ms	2.5 ms	3.6 ms
n^5	100ms	0.003s	0.02s	0.1 s	0.3 s	0.7 s
2^n	1 ms	0.001s	1 s	18 m	13 T	36 J
3^n	59 ms	3 s	2 T	385J	$2 \cdot 10^7$ J	10^{12} J

ms=Mikrosek., s=Sek., m=Minute, T=Tag, J=Jahr

Problemgröße lösbar in fester Zeit: Speedup

Komplexität	n	n^2	n^5	2^n	3^n
1 MHz Prozessor	N_1	N_2	N_3	N_4	N_5
1 GHz Prozessor	1000 N_1	32 N_2	4 N_3	$N_4 + 10$	$N_5 + 6$

Zwei zentrale Komplexitätsklassen:

P = die von einer DTM in polynomieller Zeit lösbaren Probleme

Polynomielle und exponentielle Komplexität

Taktfrequenz: 1GHz

	Problemgröße n					
Zeit	10	20	30	40	50	60
n	.01 ms	.02 ms	.03 ms	.04 ms	.05 ms	.06 ms
n^2	.1 ms	.4 ms	.9 ms	1.6 ms	2.5 ms	3.6 ms
n^5	100ms	0.003s	0.02s	0.1 s	0.3 s	0.7 s
2^n	1 ms	0.001s	1 s	18 m	13 T	36 J
3^n	59 ms	3 s	2 T	385J	$2 \cdot 10^7$ J	10^{12} J

ms=Mikrosek., s=Sek., m=Minute, T=Tag, J=Jahr

Problemgröße lösbar in fester Zeit: Speedup

Komplexität	n	n^2	n^5	2^n	3^n
1 MHz Prozessor	N_1	N_2	N_3	N_4	N_5
1 GHz Prozessor	1000 N_1	32 N_2	4 N_3	$N_4 + 10$	$N_5 + 6$

Zwei zentrale Komplexitätsklassen:

P = die von einer DTM in polynomieller Zeit lösbaren Probleme
= die „leichten“ Probleme (*feasible* problems)

NP = die von einer NTM in polynomieller Zeit lösbaren Probleme

Zwei zentrale Komplexitätsklassen:

P = die von einer DTM in polynomieller Zeit lösbaren Probleme
= die „leichten“ Probleme (*feasible* problems)

NP = die von einer NTM in polynomieller Zeit lösbaren Probleme
In exponentieller Zeit lösbar (s. Simulation NTM durch DTM)

Zwei zentrale Komplexitätsklassen:

P = die von einer DTM in polynomieller Zeit lösbaren Probleme
= die „leichten“ Probleme (*feasible* problems)

NP = die von einer NTM in polynomieller Zeit lösbaren Probleme
In exponentieller Zeit lösbar (s. Simulation NTM durch DTM)

Zentrale Frage:

¿ P = NP ?

Zwei zentrale Komplexitätsklassen:

P = die von einer DTM in polynomieller Zeit lösbaren Probleme
= die „leichten“ Probleme (*feasible* problems)

NP = die von einer NTM in polynomieller Zeit lösbaren Probleme
In exponentieller Zeit lösbar (s. Simulation NTM durch DTM)

Zentrale Frage:

¿ P = NP ?

Ist wichtig

- nicht weil Intel den NPentium angekündigt hat
- sondern weil viele *praktisch relevante* Such- und Optimierungsprobleme in NP liegen

Zwei zentrale Komplexitätsklassen:

P = die von einer DTM in polynomieller Zeit lösbaren Probleme
= die „leichten“ Probleme (*feasible* problems)

NP = die von einer NTM in polynomieller Zeit lösbaren Probleme
In exponentieller Zeit lösbar (s. Simulation NTM durch DTM)

Zentrale Frage:

¿ P = NP ?

Ist wichtig

- nicht weil Intel den NPentium angekündigt hat
- sondern weil viele *praktisch relevante* Such- und Optimierungsprobleme in NP liegen
- und alle schnell lösbar wären wenn eines schnell lösbar wäre.

DEXPTIME

Überblick:

1. Die Komplexitätsklassen P und NP

Überblick:

1. Die Komplexitätsklassen P und NP
2. NP-Vollständigkeit

Überblick:

1. Die Komplexitätsklassen P und NP
2. NP-Vollständigkeit
3. SAT: Ein NP-vollständiges Problem
4. Weitere NP-vollständige Probleme

5.1 Die Komplexitätsklasse P

Berechnungsmodelle:

DTM = deterministische Mehrband-TM

NTM = nichtdeterministische Mehrband-TM

5.1 Die Komplexitätsklasse P

Berechnungsmodelle:

DTM = deterministische Mehrband-TM

NTM = nichtdeterministische Mehrband-TM

Definition 5.1

$time_M(w)$:= Anzahl der Schritte bis die DTM $M[w]$ hält

5.1 Die Komplexitätsklasse P

Berechnungsmodelle:

DTM = deterministische Mehrband-TM

NTM = nichtdeterministische Mehrband-TM

Definition 5.1

$time_M(w)$:= Anzahl der Schritte bis die DTM $M[w]$ hält
$\in \mathbb{N} \cup \{\infty\}$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine totale Funktion.

Die Klasse der in Zeit $f(n)$ entscheidbaren Sprachen:

5.1 Die Komplexitätsklasse P

Berechnungsmodelle:

DTM = deterministische Mehrband-TM

NTM = nichtdeterministische Mehrband-TM

Definition 5.1

$time_M(w)$:= Anzahl der Schritte bis die DTM $M[w]$ hält $\in \mathbb{N} \cup \{\infty\}$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine totale Funktion.

Die Klasse der in Zeit $f(n)$ entscheidbaren Sprachen:

$$TIME(f(n)) := \{A \subseteq \Sigma^* \mid \exists \text{DTM } M.\ A = L(M)$$

5.1 Die Komplexitätsklasse P

Berechnungsmodelle:

DTM = deterministische Mehrband-TM

NTM = nichtdeterministische Mehrband-TM

Definition 5.1

$time_M(w)$:= Anzahl der Schritte bis die DTM $M[w]$ hält $\in \mathbb{N} \cup \{\infty\}$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine totale Funktion.

Die Klasse der in Zeit $f(n)$ entscheidbaren Sprachen:

$$TIME(f(n)) := \{A \subseteq \Sigma^* \mid \exists \text{DTM } M.\ A = L(M) \wedge \forall w \in \Sigma^*.\ time_M(w) \leq f(|w|)\}$$

5.1 Die Komplexitätsklasse P

Berechnungsmodelle:

DTM = deterministische Mehrband-TM

NTM = nichtdeterministische Mehrband-TM

Definition 5.1

$time_M(w)$:= Anzahl der Schritte bis die DTM $M[w]$ hält $\in \mathbb{N} \cup \{\infty\}$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine totale Funktion.

Die Klasse der in Zeit $f(n)$ entscheidbaren Sprachen:

$TIME(f(n)) := \{A \subseteq \Sigma^* \mid \exists \text{DTM } M.\ A = L(M) \wedge \forall w \in \Sigma^*.\ time_M(w) \leq f(|w|)\}$

Merke:

- n ist implizit die Länge der Eingabe

5.1 Die Komplexitätsklasse P

Berechnungsmodelle:

DTM = deterministische Mehrband-TM

NTM = nichtdeterministische Mehrband-TM

Definition 5.1

$time_M(w)$:= Anzahl der Schritte bis die DTM $M[w]$ hält $\in \mathbb{N} \cup \{\infty\}$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine totale Funktion.

Die Klasse der in Zeit $f(n)$ entscheidbaren Sprachen:

$TIME(f(n)) := \{A \subseteq \Sigma^* \mid \exists \text{DTM } M.\ A = L(M) \wedge \forall w \in \Sigma^*.\ time_M(w) \leq f(|w|)\}$

Merke:

- n ist implizit die Länge der Eingabe

- Die DTM entscheidet die Sprache A in $\leq f(n)$ Schritten

Wir betrachten nur Polynome mit Koeffizienten $a_k, \ldots a_0 \in \mathbb{N}$:

$$p(n) = a_k n^k + \cdots + a_1 n + a_0$$

Wir betrachten nur Polynome mit Koeffizienten $a_k, \ldots a_0 \in \mathbb{N}$:

$$p(n) = a_k n^k + \cdots + a_1 n + a_0$$

Definition 5.2

$$\mathsf{P} := \bigcup_{p \ Polynom} \mathit{TIME}(p(n))$$

Wir betrachten nur Polynome mit Koeffizienten $a_k, \ldots a_0 \in \mathbb{N}$:

$$p(n) = a_k n^k + \cdots + a_1 n + a_0$$

Definition 5.2

$$\mathsf{P} := \bigcup_{p \ Polynom} \mathit{TIME}(p(n))$$

Damit gilt auch

$$\mathsf{P} = \bigcup_{k \geq 0} \mathit{TIME}(O(n^k))$$

Wir betrachten nur Polynome mit Koeffizienten $a_k, \ldots a_0 \in \mathbb{N}$:

$$p(n) = a_k n^k + \cdots + a_1 n + a_0$$

Definition 5.2

$$\mathsf{P} := \bigcup_{p \ Polynom} \mathit{TIME}(p(n))$$

Damit gilt auch

$$\mathsf{P} = \bigcup_{k \geq 0} \mathit{TIME}(O(n^k))$$

wobei

$$\mathit{TIME}(O(f)) := \bigcup_{g \in O(f)} \mathit{TIME}(g)$$

Beispiel 5.3

- $\{ww^R \mid w \in \Sigma^*\} \in \mathit{TIME}(O(n^2)) \subseteq \mathsf{P}$

Beispiel 5.3

- $\{ww^R \mid w \in \Sigma^*\} \in \mathit{TIME}(O(n^2)) \subseteq \mathsf{P}$
- $\{(G, w) \mid G$ ist CFG $\wedge\, w \in L(G)\} \in \mathsf{P}$

Beispiel 5.3

- $\{ww^R \mid w \in \Sigma^*\} \in \mathit{TIME}(O(n^2)) \subseteq \mathsf{P}$
- $\{(G, w) \mid G$ ist CFG $\wedge\, w \in L(G)\} \in \mathsf{P}$
- $\{(G, w) \mid G$ ist Graph $\wedge\, w$ ist Pfad in $G\} \in \mathsf{P}$

Beispiel 5.3

- $\{ww^R \mid w \in \Sigma^*\} \in \mathit{TIME}(O(n^2)) \subseteq \mathsf{P}$
- $\{(G, w) \mid G$ ist CFG $\wedge\, w \in L(G)\} \in \mathsf{P}$
- $\{(G, w) \mid G$ ist Graph $\wedge\, w$ ist Pfad in $G\} \in \mathsf{P}$
- $\{bin(p) \mid p$ Primzahl$\} \in \mathsf{P}$

Beispiel 5.3

- $\{ww^R \mid w \in \Sigma^*\} \in \mathit{TIME}(O(n^2)) \subseteq \mathsf{P}$
- $\{(G, w) \mid G$ ist CFG $\wedge\, w \in L(G)\} \in \mathsf{P}$
- $\{(G, w) \mid G$ ist Graph $\wedge\, w$ ist Pfad in $G\} \in \mathsf{P}$
- $\{bin(p) \mid p$ Primzahl$\} \in \mathsf{P}$

Beweis durch Angabe eines Algorithmus mit Komplexität $O(n^k)$.

Beispiel 5.3

- $\{ww^R \mid w \in \Sigma^*\} \in \mathit{TIME}(O(n^2)) \subseteq \mathsf{P}$
- $\{(G, w) \mid G$ ist CFG $\wedge\, w \in L(G)\} \in \mathsf{P}$
- $\{(G, w) \mid G$ ist Graph $\wedge\, w$ ist Pfad in $G\} \in \mathsf{P}$
- $\{bin(p) \mid p$ Primzahl$\} \in \mathsf{P}$

Beweis durch Angabe eines Algorithmus mit Komplexität $O(n^k)$.

Bemerkungen

- $O(n \log n) \subset O(n^2)$

Beispiel 5.3

- $\{ww^R \mid w \in \Sigma^*\} \in \mathit{TIME}(O(n^2)) \subseteq \mathsf{P}$
- $\{(G, w) \mid G$ ist CFG $\wedge\, w \in L(G)\} \in \mathsf{P}$
- $\{(G, w) \mid G$ ist Graph $\wedge\, w$ ist Pfad in $G\} \in \mathsf{P}$
- $\{bin(p) \mid p$ Primzahl$\} \in \mathsf{P}$

Beweis durch Angabe eines Algorithmus mit Komplexität $O(n^k)$.

Bemerkungen

- $O(n \log n) \subset O(n^2)$

- $n^{\log n}, 2^n \notin O(n^k)$ für alle k

Beispiel 5.3

- $\{ww^R \mid w \in \Sigma^*\} \in \mathit{TIME}(O(n^2)) \subseteq \mathsf{P}$
- $\{(G, w) \mid G$ ist CFG $\wedge\, w \in L(G)\} \in \mathsf{P}$
- $\{(G, w) \mid G$ ist Graph $\wedge\, w$ ist Pfad in $G\} \in \mathsf{P}$
- $\{bin(p) \mid p$ Primzahl$\} \in \mathsf{P}$

Beweis durch Angabe eines Algorithmus mit Komplexität $O(n^k)$.

Bemerkungen

- $O(n \log n) \subset O(n^2)$
- $n^{\log n}, 2^n \notin O(n^k)$ für alle k

- Beweis $A \notin \mathsf{P}$ meist schwierig

- Warum P und nicht (zB) $O(n^{17})$?

- Warum P und nicht (zB) $O(n^{17})$?
  Um robust bzgl Maschinenmodell zu sein:

- Warum P und nicht (zB) $O(n^{17})$?
  Um robust bzgl Maschinenmodell zu sein:
  1-Band DTM braucht $O(t^2)$ Schritte
  um t Schritte einer k-Band DTM zu simulieren.
  Fast alle bekannten „vernünftigen“ Maschinenmodelle lassen sich von einer DTM in polynomieller Zeit simulieren.

- Warum P und nicht (zB) $O(n^{17})$?
  Um robust bzgl Maschinenmodell zu sein:
  1-Band DTM braucht $O(t^2)$ Schritte
  um t Schritte einer k-Band DTM zu simulieren.
  Fast alle bekannten „vernünftigen“ Maschinenmodelle lassen sich von einer DTM in polynomieller Zeit simulieren.
  Offen: Quantencomputer

- Warum P und nicht (zB) $O(n^{17})$?
  Um robust bzgl Maschinenmodell zu sein:
  1-Band DTM braucht $O(t^2)$ Schritte
  um t Schritte einer k-Band DTM zu simulieren.
  Fast alle bekannten „vernünftigen“ Maschinenmodelle lassen sich von einer DTM in polynomieller Zeit simulieren.
  Offen: Quantencomputer

- Warum TM? Historisch.
  Ebenfalls möglich: (zB) WHILE.
  Aber zwei mögliche Kostenmodelle:

- Warum P und nicht (zB) $O(n^{17})$?
  Um robust bzgl Maschinenmodell zu sein:
  1-Band DTM braucht $O(t^2)$ Schritte
  um t Schritte einer k-Band DTM zu simulieren.
  Fast alle bekannten „vernünftigen“ Maschinenmodelle lassen sich von einer DTM in polynomieller Zeit simulieren.
  Offen: Quantencomputer

- Warum TM? Historisch.
  Ebenfalls möglich: (zB) WHILE.
  Aber zwei mögliche Kostenmodelle:

Uniform x_i := x_j + n kostet 1 Zeiteinheit.

- Warum P und nicht (zB) $O(n^{17})$?
  Um robust bzgl Maschinenmodell zu sein:
  1-Band DTM braucht $O(t^2)$ Schritte
  um t Schritte einer k-Band DTM zu simulieren.
  Fast alle bekannten „vernünftigen“ Maschinenmodelle lassen sich von einer DTM in polynomieller Zeit simulieren.
  Offen: Quantencomputer

- Warum TM? Historisch.
  Ebenfalls möglich: (zB) WHILE.
  Aber zwei mögliche Kostenmodelle:

Uniform x_i := x_j + n kostet 1 Zeiteinheit.

Logarithmisch x_i := x_j + n kostet $\log x_j$ Zeiteinheiten.

- Warum P und nicht (zB) $O(n^{17})$?
  Um robust bzgl Maschinenmodell zu sein:
  1-Band DTM braucht $O(t^2)$ Schritte
  um t Schritte einer k-Band DTM zu simulieren.
  Fast alle bekannten „vernünftigen“ Maschinenmodelle lassen sich von einer DTM in polynomieller Zeit simulieren.
  Offen: Quantencomputer

- Warum TM? Historisch.
  Ebenfalls möglich: (zB) WHILE.
  Aber zwei mögliche Kostenmodelle:

Uniform x_i := x_j + n kostet 1 Zeiteinheit.

Logarithmisch x_i := x_j + n kostet $\log x_j$ Zeiteinheiten.

5.2 Die Komplexitätsklasse NP

Intuitiv:

- NP die Klasse der Sprachen, die von einer NTM in polynomieller Zeit akzeptiert werden.

5.2 Die Komplexitätsklasse NP

Intuitiv:

- NP die Klasse der Sprachen, die von einer NTM in polynomieller Zeit akzeptiert werden.
- Dh: Eine Sprache A liegt in NP gdw
  es ein Polynom $p(n)$ und eine NTM M gibt, so dass

5.2 Die Komplexitätsklasse NP

Intuitiv:

- NP die Klasse der Sprachen, die von einer NTM in polynomieller Zeit akzeptiert werden.
- Dh: Eine Sprache A liegt in NP gdw
  es ein Polynom $p(n)$ und eine NTM M gibt, so dass
  1. $L(M) = A$ und
  2. für alle $w \in A$ *kann* $M[w]$ in $\leq p(|w|)$ Schritten akzeptieren, dh einen Endzustand erreichen.

Definition 5.4

$$\mathit{ntime}_M(w) := \begin{cases} \text{minimale Anzahl der Schritte bis NTM } M[w] \text{ akzeptiert} & \text{falls } w \in L(M) \\ 0 & \text{falls } w \notin L(M) \end{cases}$$

Definition 5.4

$$\mathit{ntime}_M(w) := \begin{cases} \text{minimale Anzahl der Schritte} \\ \text{bis NTM } M[w] \text{ akzeptiert} & \text{falls } w \in L(M) \\ 0 & \text{falls } w \notin L(M) \end{cases}$$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine totale Funktion.

$$\mathit{NTIME}(f(n)) := \{A \subseteq \Sigma^* \mid \exists \text{NTM } M.\ A = L(M)$$

Definition 5.4

$$\mathit{ntime}_M(w) := \begin{cases} \text{minimale Anzahl der Schritte} \\ \text{bis NTM } M[w] \text{ akzeptiert} & \text{falls } w \in L(M) \\ 0 & \text{falls } w \notin L(M) \end{cases}$$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine totale Funktion.

$$\mathit{NTIME}(f(n)) := \{A \subseteq \Sigma^* \mid \exists \text{NTM } M.\ A = L(M) \wedge \forall w \in \Sigma^*.\ \mathit{ntime}_M(w) \leq f(|w|)\}$$

Definition 5.4

$$\mathit{ntime}_M(w) := \begin{cases} \text{minimale Anzahl der Schritte} \\ \text{bis NTM } M[w] \text{ akzeptiert} & \text{falls } w \in L(M) \\ 0 & \text{falls } w \notin L(M) \end{cases}$$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine totale Funktion.

$$\mathit{NTIME}(f(n)) := \{A \subseteq \Sigma^* \mid \exists \text{NTM } M.\ A = L(M) \wedge \forall w \in \Sigma^*.\ \mathit{ntime}_M(w) \leq f(|w|)\}$$

$$\text{NP} := \bigcup_{p\ \mathit{Polynom}} \mathit{NTIME}(p(n))$$

Definition 5.4

$$\mathit{ntime}_M(w) := \begin{cases} \text{minimale Anzahl der Schritte} \\ \text{bis NTM } M[w] \text{ akzeptiert} & \text{falls } w \in L(M) \\ 0 & \text{falls } w \notin L(M) \end{cases}$$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine totale Funktion.

$$\mathit{NTIME}(f(n)) := \{A \subseteq \Sigma^* \mid \exists \text{NTM } M.\ A = L(M) \wedge \forall w \in \Sigma^*.\ \mathit{ntime}_M(w) \leq f(|w|)\}$$

$$\text{NP} := \bigcup_{p\ \mathit{Polynom}} \mathit{NTIME}(p(n))$$

Bemerkungen:

- P $\subseteq$ NP

Definition 5.4

$$\mathit{ntime}_M(w) := \begin{cases} \text{minimale Anzahl der Schritte} \\ \text{bis NTM } M[w] \text{ akzeptiert} & \text{falls } w \in L(M) \\ 0 & \text{falls } w \notin L(M) \end{cases}$$

Sei $f : \mathbb{N} \to \mathbb{N}$ eine totale Funktion.

$$\begin{aligned} \mathit{NTIME}(f(n)) &:= \{A \subseteq \Sigma^* \mid \exists \text{NTM } M.\ A = L(M) \wedge \\ &\qquad \forall w \in \Sigma^*.\ \mathit{ntime}_M(w) \leq f(|w|)\} \\ \text{NP} &:= \bigcup_{p\ \mathit{Polynom}} \mathit{NTIME}(p(n)) \end{aligned}$$

Bemerkungen zur Definition von NP:
Akzeptierende NTM M

- braucht für $w \notin L(M)$ nicht zu halten.
- kann für $w \in L(M)$ auch beliebig lange Berechnunsgfolgen haben.

Bemerkungen zur Definition von NP:
Akzeptierende NTM M

- braucht für $w \notin L(M)$ nicht zu halten.
- kann für $w \in L(M)$ auch beliebig lange Berechnunsgfolgen haben.

Äquivalente Definition NP' von NP:
Die NTM $M[w]$ *muss* nach maximal $p(|w|)$ Schritten halten.

Bemerkungen zur Definition von NP:
Akzeptierende NTM M

- braucht für $w \notin L(M)$ nicht zu halten.
- kann für $w \in L(M)$ auch beliebig lange Berechnunsgfolgen haben.

Äquivalente Definition NP' von NP:
Die NTM $M[w]$ *muss* nach maximal $p(|w|)$ Schritten halten.

NP' $\subseteq$ NP: Klar.

Bemerkungen zur Definition von NP:
Akzeptierende NTM M

- braucht für $w \notin L(M)$ nicht zu halten.
- kann für $w \in L(M)$ auch beliebig lange Berechnunsgfolgen haben.

Äquivalente Definition NP' von NP:
Die NTM $M[w]$ *muss* nach maximal $p(|w|)$ Schritten halten.

NP' $\subseteq$ NP: Klar.

NP $\subseteq$ NP':

Bemerkungen zur Definition von NP:
Akzeptierende NTM M

- braucht für $w \notin L(M)$ nicht zu halten.
- kann für $w \in L(M)$ auch beliebig lange Berechnunsgfolgen haben.

Äquivalente Definition NP' von NP:
Die NTM $M[w]$ *muss* nach maximal $p(|w|)$ Schritten halten.

NP' $\subseteq$ NP: Klar.

NP $\subseteq$ NP': Falls $A \in$ NP mit Polynom p und NTM M,
so kann man NTM M' konstruieren mit $L(M') = A$,
so dass $M'[w]$ immer innerhalb von polynomieller Zeit hält.

1. Eingabe w
2. Schreibe $p(|w|)$ auf ein getrenntes Band („timer“)

Bemerkungen zur Definition von NP:
Akzeptierende NTM M

- braucht für $w \notin L(M)$ nicht zu halten.
- kann für $w \in L(M)$ auch beliebig lange Berechnunsgfolgen haben.

Äquivalente Definition NP' von NP:
Die NTM $M[w]$ *muss* nach maximal $p(|w|)$ Schritten halten.

NP' $\subseteq$ NP: Klar.

NP $\subseteq$ NP': Falls $A \in$ NP mit Polynom p und NTM M,
so kann man NTM M' konstruieren mit $L(M') = A$,
so dass $M'[w]$ immer innerhalb von polynomieller Zeit hält.

1. Eingabe w
2. Schreibe $p(|w|)$ auf ein getrenntes Band („timer“)

3. Simuliere $M[w]$, aber dekrementiere timer nach jedem Schritt.
4. Wird timer=0, ohne dass M gehalten hat, halte in einem nicht-Endzustand („timeout“)

Beispiel 5.5

- Ein Euler-Kreis ist ein geschlossener Pfad in einem (ungerichteten) Graphen, der jede Kante genau einmal enthält.

Beispiel 5.5

- Ein Euler-Kreis ist ein geschlossener Pfad in einem (ungerichteten) Graphen, der jede Kante genau einmal enthält.
- Ein Graph hat einen Euler-Kreis gdw er zusammenhängend ist, und jeder Knoten geraden Grad hat.

Beispiel 5.5

- Ein Euler-Kreis ist ein geschlossener Pfad in einem (ungerichteten) Graphen, der jede Kante genau einmal enthält.
- Ein Graph hat einen Euler-Kreis gdw er zusammenhängend ist, und jeder Knoten geraden Grad hat.
- Beide Eigenschaften sind in polynomieller Zeit von einer DTM überprüfbar.

Beispiel 5.6

- Ein Hamilton-Kreis ist ein geschlossener Pfad in einem (ungerichteten) Graphen, der jeden Knoten genau einmal enthält.

Beispiel 5.6

- Ein Hamilton-Kreis ist ein geschlossener Pfad in einem (ungerichteten) Graphen, der jeden Knoten genau einmal enthält.
- HAMILTON $:= \{G \mid G \text{ hat Hamilton-Kreis}\} \in \text{NP}$

Beispiel 5.6

- Ein Hamilton-Kreis ist ein geschlossener Pfad in einem (ungerichteten) Graphen, der jeden Knoten genau einmal enthält.
- HAMILTON $:= \{G \mid G \text{ hat Hamilton-Kreis}\} \in$ NP mit folgendem Algorithmus der Art „Rate und prüfe“:

Beispiel 5.6

- Ein Hamilton-Kreis ist ein geschlossener Pfad in einem (ungerichteten) Graphen, der jeden Knoten genau einmal enthält.
- HAMILTON $:= \{G \mid G \text{ hat Hamilton-Kreis}\} \in$ NP mit folgendem Algorithmus der Art „Rate und prüfe“:
  1. Rate eine Permutation der Knoten des Graphen.
  2. Prüfe, ob diese Permutation ein Hamilton-Kreis ist.

Beispiel 5.6

- Ein Hamilton-Kreis ist ein geschlossener Pfad in einem (ungerichteten) Graphen, der jeden Knoten genau einmal enthält.
- HAMILTON $:= \{G \mid G \text{ hat Hamilton-Kreis}\} \in$ NP mit folgendem Algorithmus der Art „Rate und prüfe“:
  1. Rate eine Permutation der Knoten des Graphen.
  2. Prüfe, ob diese Permutation ein Hamilton-Kreis ist.

  Das Raten ist in polynomieller Zeit von einer NTM machbar.

Beispiel 5.6

- Ein Hamilton-Kreis ist ein geschlossener Pfad in einem (ungerichteten) Graphen, der jeden Knoten genau einmal enthält.
- HAMILTON $:= \{G \mid G \text{ hat Hamilton-Kreis}\} \in$ NP mit folgendem Algorithmus der Art „Rate und prüfe“:
  1. Rate eine Permutation der Knoten des Graphen.
  2. Prüfe, ob diese Permutation ein Hamilton-Kreis ist.

  Das Raten ist in polynomieller Zeit von einer NTM machbar.
  Das Prüfen ist in polynomieller Zeit von einer DTM machbar.

Beispiel 5.6

- Ein Hamilton-Kreis ist ein geschlossener Pfad in einem (ungerichteten) Graphen, der jeden Knoten genau einmal enthält.
- HAMILTON $:= \{G \mid G \text{ hat Hamilton-Kreis}\} \in$ NP
  mit folgendem Algorithmus der Art „Rate und prüfe“:
  1. Rate eine Permutation der Knoten des Graphen.
  2. Prüfe, ob diese Permutation ein Hamilton-Kreis ist.

  Das Raten ist in polynomieller Zeit von einer NTM machbar.
  Das Prüfen ist in polynomieller Zeit von einer DTM machbar.

Vermutung: HAMILTON $\notin$ P

Viele Probleme sind von der Art dass

- schwer ist, zu entscheiden, ob sie lösbar sind,

Viele Probleme sind von der Art dass

- schwer ist, zu entscheiden, ob sie lösbar sind,
- leicht ist, zu entscheiden, ob eine Lösungsvorschlag eine Lösung ist.

Viele Probleme sind von der Art dass

- schwer ist, zu entscheiden, ob sie lösbar sind,
- leicht ist, zu entscheiden, ob eine Lösungsvorschlag eine Lösung ist.

Definition 5.7
Sei M eine DTM mit $L(M) \subseteq \{w\#c \mid w \in \Sigma^*, c \in \Delta^*\}$.

Viele Probleme sind von der Art dass

- schwer ist, zu entscheiden, ob sie lösbar sind,
- leicht ist, zu entscheiden, ob eine Lösungsvorschlag eine Lösung ist.

Definition 5.7
Sei M eine DTM mit $L(M) \subseteq \{w\#c \mid w \in \Sigma^*, c \in \Delta^*\}$.

- Falls $w\#c \in L(M)$, so heißt c Zertifikat für w.

Viele Probleme sind von der Art dass

- schwer ist, zu entscheiden, ob sie lösbar sind,
- leicht ist, zu entscheiden, ob eine Lösungsvorschlag eine Lösung ist.

Definition 5.7
Sei M eine DTM mit $L(M) \subseteq \{w\#c \mid w \in \Sigma^*, c \in \Delta^*\}$.

- Falls $w\#c \in L(M)$, so heißt c Zertifikat für w.
- M ist ein polynomiell beschränkter Verifikator für die Sprache $\{w \in \Sigma^* \mid \exists c \in \Delta^*.\ w\#c \in L(M)\}$

Viele Probleme sind von der Art dass

- schwer ist, zu entscheiden, ob sie lösbar sind,
- leicht ist, zu entscheiden, ob eine Lösungsvorschlag eine Lösung ist.

Definition 5.7
Sei M eine DTM mit $L(M) \subseteq \{w\#c \mid w \in \Sigma^*, c \in \Delta^*\}$.

- Falls $w\#c \in L(M)$, so heißt c Zertifikat für w.
- M ist ein polynomiell beschränkter Verifikator für die Sprache $\{w \in \Sigma^* \mid \exists c \in \Delta^*.\ w\#c \in L(M)\}$
  falls es ein Polynom p gibt, so dass $\mathit{time}_M(w\#c) \leq p(|w|)$.

Viele Probleme sind von der Art dass

- schwer ist, zu entscheiden, ob sie lösbar sind,
- leicht ist, zu entscheiden, ob eine Lösungsvorschlag eine Lösung ist.

Definition 5.7
Sei M eine DTM mit $L(M) \subseteq \{w\#c \mid w \in \Sigma^*, c \in \Delta^*\}$.

- Falls $w\#c \in L(M)$, so heißt c Zertifikat für w.
- M ist ein polynomiell beschränkter Verifikator für die Sprache $\{w \in \Sigma^* \mid \exists c \in \Delta^*.\ w\#c \in L(M)\}$
  falls es ein Polynom p gibt, so dass $\mathit{time}_M(w\#c) \leq p(|w|)$.

NB:
In Zeit $p(n)$ kann M maximal die ersten $p(n)$ Zeichen von c lesen.

Beispiel 5.9 (RUCKSACK)

Gegeben: Zahlen $a_1, \ldots a_n \in \mathbb{N}$ und $c \in \mathbb{N}$.

Beispiel 5.9 (RUCKSACK)

Gegeben: Zahlen $a_1, \ldots a_n \in \mathbb{N}$ und $c \in \mathbb{N}$.
Problem: Gibt es $R \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in R} a_i = c$?

Beispiel 5.9 (RUCKSACK)

Gegeben: Zahlen $a_1, \ldots a_n \in \mathbb{N}$ und $c \in \mathbb{N}$.
Problem: Gibt es $R \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in R} a_i = c$?

$$\text{RUCKSACK} := \{bin(a_1)\# \ldots \#bin(a_n)\#bin(c) \mid \exists R \subseteq \{1, \ldots, n\}.\ \sum_{i \in R} a_i = c\}$$

RUCKSACK $\in$ NP:
Zertifikat: Indexmenge R. In polynomieller Zeit verifizierbar.

Beispiel 5.9 (RUCKSACK)

Gegeben: Zahlen $a_1, \ldots a_n \in \mathbb{N}$ und $c \in \mathbb{N}$.
Problem: Gibt es $R \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in R} a_i = c$?

$$\text{RUCKSACK} := \{bin(a_1)\# \ldots \#bin(a_n)\#bin(c) \mid \exists R \subseteq \{1, \ldots, n\}.\ \sum_{i \in R} a_i = c\}$$

RUCKSACK $\in$ NP:
Zertifikat: Indexmenge R. In polynomieller Zeit verifizierbar.

Merke: Der Nachweis $A \in$ NP ist meistens einfach.

Satz 5.10

$A \in$ NP gdw es gibt polynomiell beschränkten Verifikator für A.

Satz 5.10

$A \in$ NP gdw es gibt polynomiell beschränkten Verifikator für A.

Beweis:

„$\Rightarrow$“:

Sei $A \in$ NP. Dh es gibt NTM N, die A in Zeit $p(n)$ akzeptiert.

Satz 5.10

$A \in$ NP gdw es gibt polynomiell beschränkten Verifikator für A.

Beweis:

„$\Rightarrow$“:

Sei $A \in$ NP. Dh es gibt NTM N, die A in Zeit $p(n)$ akzeptiert.

Satz 5.10

$A \in$ NP gdw es gibt polynomiell beschränkten Verifikator für A.

Beweis:

„$\Rightarrow$“:

Sei $A \in$ NP. Dh es gibt NTM N, die A in Zeit $p(n)$ akzeptiert.

Ein Zertifikat für $w \in A$ ist die Folge der benutzten Transitionen $\delta(q, a) \ni (q', a', d)$ einer akzeptierenden Berechnunsgfolge von $N[w]$ mit $\leq p(n)$ Schritten.

Satz 5.10

$A \in$ NP gdw es gibt polynomiell beschränkten Verifikator für A.

Beweis:

„$\Rightarrow$“:
Sei $A \in$ NP. Dh es gibt NTM N, die A in Zeit $p(n)$ akzeptiert.
Ein Zertifikat für $w \in A$ ist die Folge der benutzten Transitionen $\delta(q, a) \ni (q', a', d)$ einer akzeptierenden Berechnunsgfolge von $N[w]$ mit $\leq p(n)$ Schritten.
Ein polynomiell beschränkter Verfikator für $L(N)$:

1. Eingabe $w\#c$
2. Simuliere $N[w]$, gesteuert durch die Transitionen in c.
3. Überprüfe dabei, ob die Transition in c jeweils zu N und zur augenblicklichen Konfiguration von N passt.
4. Akzeptiere, falls c in einen Endzustand führt.

Beweis (Forts.):

„$\Leftarrow$“:

Beweis (Forts.):

„$\Leftarrow$“:
Sei M ein polynomiell (durch p) beschränkter Verifikator für A.

Beweis (Forts.):

„$\Leftarrow$“:
Sei M ein polynomiell (durch p) beschränkter Verifikator für A.
Wir bauen eine polynomiell beschränkte NTM N mit $L(M) = A$:

1. Eingabe w.
2. Schreibe hinter w zuerst $\#$ und dann ein nichtdeterministisch gewähltes Wort $c \in \Delta^*$, $|c| = p(|w|)$:
   for $i := 1, \ldots, p(|w|)$ **do**
   schreibe ein Zeichen aus Δ und gehe nach rechts
3. Führe M aus (mit Eingabe $w\#c$).

Nach Annahme gilt $\mathit{time}_M(w\#c) \leq p(|w|)$.

Fazit:

P sind die Sprachen, bei denen $w \in L$ schnell entschieden werden kann.

Fazit:

P sind die Sprachen, bei denen $w \in L$ schnell entschieden werden kann.

NP sind die Sprachen, bei denen ein Zertifkat für $w \in L$ schnell verifiziert/überprüft werden kann.

Fazit:

P sind die Sprachen, bei denen $w \in L$ schnell entschieden werden kann.

NP sind die Sprachen, bei denen ein Zertifkat für $w \in L$ schnell verifiziert/überprüft werden kann.

Intuition:
Es ist leichter, eine Lösung zu verifizieren als zu finden.

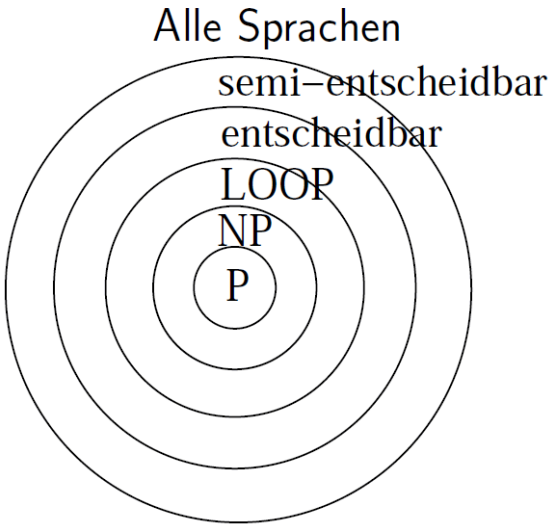
Fazit:

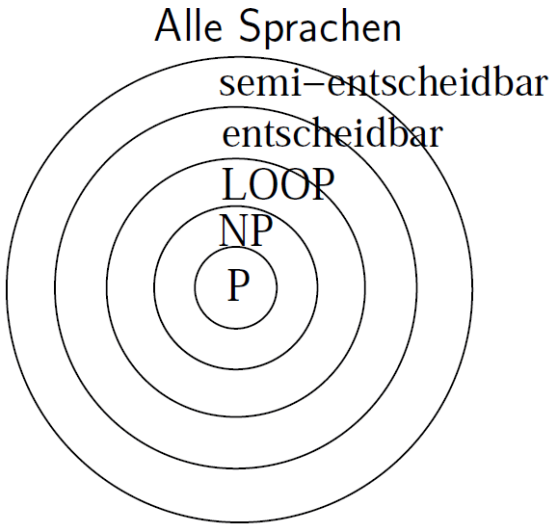
P sind die Sprachen, bei denen $w \in L$ schnell entschieden werden kann.

NP sind die Sprachen, bei denen ein Zertifkat für $w \in L$ schnell verifiziert/überprüft werden kann.

Intuition:
Es ist leichter, eine Lösung zu verifizieren als zu finden.

Aber:
Noch wurde von keiner Sprache bewiesen, dass sie in NP\P liegt.





Nur die P-NP-Grenze ist offen.

Satz 5.11
Ist $A \in TIME(f)$ und ist f LOOP-berechenbar,
dann ist A LOOP-entscheidbar, dh χ_A ist LOOP-berechenbar.

Satz 5.11
Ist $A \in TIME(f)$ und ist f LOOP-berechenbar,
dann ist A LOOP-entscheidbar, dh χ_A ist LOOP-berechenbar.

Beweis:
Sei M eine DTM mit $L(M) = A$ und $time_M(w) \leq f(|w|)$.

Satz 5.11
Ist $A \in$ TIME(f) und ist f LOOP-berechenbar,
dann ist A LOOP-entscheidbar, dh χ_A ist LOOP-berechenbar.

Beweis:
Sei M eine DTM mit $L(M) = A$ und $time_M(w) \leq f(|w|)$.
$M \mapsto$ GOTO $\mapsto$ WHILE:

Satz 5.11
Ist $A \in$ TIME(f) und ist f LOOP-berechenbar,
dann ist A LOOP-entscheidbar, dh χ_A ist LOOP-berechenbar.

Beweis:
Sei M eine DTM mit $L(M) = A$ und $time_M(w) \leq f(|w|)$.
$M \mapsto$ GOTO $\mapsto$ WHILE:

```
pc := 1;
WHILE pc ≠ 0 DO
  IF pc = 1 THEN P_1 ELSE
  ...
  IF pc = k THEN P_k ELSE pc := 0
END
```