Die Komplexitätsklassen P und NP

Prof. Dr. Berthold Vöcking
Lehrstuhl Informatik 1
Algorithmen und Komplexität
RWTH Aachen

November 2011

Definition von Polynomialzeitalgorithmus

Definition (worst case Laufzeit eines Algorithmus)

Die worst case Laufzeit $t_A(n)$, $n \in \mathbb{N}$, *eines Algorithmus* A *entspricht den maximalen Laufzeitkosten auf Eingaben der Länge* n *bezüglich des logarithmischen Kostenmaßes der RAM.*

Definition von Polynomialzeitalgorithmus

Definition (worst case Laufzeit eines Algorithmus)

Die worst case Laufzeit $t_A(n)$, $n \in \mathbb{N}$, *eines Algorithmus* A *entspricht den maximalen Laufzeitkosten auf Eingaben der Länge* n *bezüglich des logarithmischen Kostenmaßes der RAM.*

Definition (Polynomialzeitalgorithmus)

Wir sagen, die worst case Laufzeit $t_A(n)$ *eines Algorithmus* A *ist* polynomiell beschränkt, *falls gilt*

$$\exists \alpha \in \mathbb{N} : t_A(n) = O(n^\alpha) \ .$$

Einen Algorithmus mit polynomiell beschränkter worst case Laufzeit bezeichnen wir als Polynomialzeitalgorithmus.

Definition der Klasse P

Definition (Komplexitätsklasse P)

P *ist die Klasse der Probleme, für die es einen Polynomialzeitalgorithmus gibt.*

Definition der Klasse P

Definition (Komplexitätsklasse P)

P *ist die Klasse der Probleme, für die es einen Polynomialzeitalgorithmus gibt.*

Anmerkungen:

- Alternativ kann man sich auch auf die Laufzeit einer TM beziehen, da sich RAM und TM gegenseitig mit polynomiellen Zeitverlust simulieren können.

Definition der Klasse P

Definition (Komplexitätsklasse P)

P *ist die Klasse der Probleme, für die es einen Polynomialzeit-algorithmus gibt.*

Anmerkungen:

- Alternativ kann man sich auch auf die Laufzeit einer TM beziehen, da sich RAM und TM gegenseitig mit polynomiellen Zeitverlust simulieren können.
- Polynomialzeitalgorithmen werden häufig auch als „effiziente Algorithmen“ bezeichnet.

Definition der Klasse P

Definition (Komplexitätsklasse P)

P *ist die Klasse der Probleme, für die es einen Polynomialzeit-algorithmus gibt.*

Anmerkungen:

- Alternativ kann man sich auch auf die Laufzeit einer TM beziehen, da sich RAM und TM gegenseitig mit polynomiellen Zeitverlust simulieren können.
- Polynomialzeitalgorithmen werden häufig auch als „effiziente Algorithmen“ bezeichnet.
- P ist in diesem Sinne die Klasse derjenigen Probleme, die effizient gelöst werden können.

Problem (Sortieren)

Eingabe: *N Zahlen* $a_1, \ldots, a_N \in \mathbb{N}$
Ausgabe: *aufsteigend sortierte Folge der Eingabezahlen*

Anmerkung: Soweit wir nichts anderes sagen, nehmen wir an, dass Zahlen binär kodiert sind.

Satz

Sortieren $\in$ P.

Beweis:

- Wir lösen das Problem beispielsweise mit Mergesort.

Satz

Sortieren $\in$ P.

Beweis:

- Wir lösen das Problem beispielsweise mit Mergesort.
- Laufzeit im uniformen Kostenmaß:

Sortieren in P

Satz

Sortieren $\in$ P.

Beweis:

- Wir lösen das Problem beispielsweise mit Mergesort.
- Laufzeit im uniformen Kostenmaß: $O(N \log N)$.

Sortieren in P

Satz

Sortieren $\in$ P.

Beweis:

- Wir lösen das Problem beispielsweise mit Mergesort.
- Laufzeit im uniformen Kostenmaß: $O(N \log N)$.
- Laufzeit im logarithmischen Kostenmaß:

Sortieren in P

Satz

Sortieren $\in$ P.

Beweis:

- Wir lösen das Problem beispielsweise mit Mergesort.
- Laufzeit im uniformen Kostenmaß: $O(N \log N)$.
- Laufzeit im logarithmischen Kostenmaß: $O(\ell N \log N)$, wobei $\ell = \max_{1 \leq i \leq N} \log(a_i)$.

Sortieren in P

Satz

Sortieren $\in$ P.

Beweis:

- Wir lösen das Problem beispielsweise mit Mergesort.
- Laufzeit im uniformen Kostenmaß: $O(N \log N)$.
- Laufzeit im logarithmischen Kostenmaß: $O(\ell N \log N)$, wobei $\ell = \max_{1 \le i \le N} \log(a_i)$.
- Sei n die Eingabelänge. Es gilt $\ell \le n$ und $\log N \le N \le n$.

Sortieren in P

Satz

Sortieren $\in$ P.

Beweis:

- Wir lösen das Problem beispielsweise mit Mergesort.
- Laufzeit im uniformen Kostenmaß: $O(N \log N)$.
- Laufzeit im logarithmischen Kostenmaß: $O(\ell N \log N)$, wobei $\ell = \max_{1 \leq i \leq N} \log(a_i)$.
- Sei n die Eingabelänge. Es gilt $\ell \leq n$ und $\log N \leq N \leq n$.
- Somit ist die Laufzeit beschränkt durch $\ell N \log N \leq n^3$.

$\square$

Problem (Graphzusammenhang)

Eingabe: *Graph* $G = (V, E)$
Frage: *Ist* G *zusammenhängend?*

Anmerkung: Bei Graphproblemen gehen wir grundsätzlich davon aus, dass der Graph in Form einer Adjazenzmatrix eingegeben wird.

Graphzusammenhang in P

Satz

Graphzusammenhang $\in$ P.

Beweis

- Wir lösen das Problem mit einer Tiefensuche.
- Laufzeit im uniformen Kostenmaß:

Graphzusammenhang in P

Satz

Graphzusammenhang $\in$ P.

Beweis

- Wir lösen das Problem mit einer Tiefensuche.
- Laufzeit im uniformen Kostenmaß: $O(|V| + |E|)$

Graphzusammenhang in P

Satz

Graphzusammenhang $\in$ P.

Beweis

- Wir lösen das Problem mit einer Tiefensuche.
- Laufzeit im uniformen Kostenmaß: $O(|V| + |E|)$
- Laufzeit im logarithmischen Kostenmaß:

Graphzusammenhang in P

Satz

Graphzusammenhang $\in$ P.

Beweis

- Wir lösen das Problem mit einer Tiefensuche.
- Laufzeit im uniformen Kostenmaß: $O(|V| + |E|)$
- Laufzeit im logarithmischen Kostenmaß: $O((|V| + |E|) \cdot \log |V|)$

Graphzusammenhang in P

Satz

Graphzusammenhang $\in$ P.

Beweis

- Wir lösen das Problem mit einer Tiefensuche.
- Laufzeit im uniformen Kostenmaß: $O(|V| + |E|)$
- Laufzeit im logarithmischen Kostenmaß: $O((|V| + |E|) \cdot \log |V|)$
- Die Eingabelänge ist $n = |V|^2 \geq |E|$.

Graphzusammenhang in P

Satz

Graphzusammenhang $\in$ P.

Beweis

- Wir lösen das Problem mit einer Tiefensuche.
- Laufzeit im uniformen Kostenmaß: $O(|V| + |E|)$
- Laufzeit im logarithmischen Kostenmaß: $O((|V| + |E|) \cdot \log |V|)$
- Die Eingabelänge ist $n = |V|^2 \geq |E|$.
- Die Gesamtlaufzeit ist somit

$$O((|V| + |E|) \log |V|) \;=\; O(n \log n) \;=\; O(n^2) \enspace .$$

$\square$

Weitere Beispiele für Probleme in P

- Kürzester Weg

Weitere Beispiele für Probleme in P

- Kürzester Weg
- Minimaler Spannbaum

Weitere Beispiele für Probleme in P

- Kürzester Weg
- Minimaler Spannbaum
- Maximaler Fluss

- Kürzester Weg
- Minimaler Spannbaum
- Maximaler Fluss
- Maximum Matching

Weitere Beispiele für Probleme in P

- Kürzester Weg
- Minimaler Spannbaum
- Maximaler Fluss
- Maximum Matching
- Lineare Programmierung

Weitere Beispiele für Probleme in P

- Kürzester Weg
- Minimaler Spannbaum
- Maximaler Fluss
- Maximum Matching
- Lineare Programmierung
- Größter Gemeinsamer Teiler

Weitere Beispiele für Probleme in P

- Kürzester Weg
- Minimaler Spannbaum
- Maximaler Fluss
- Maximum Matching
- Lineare Programmierung
- Größter Gemeinsamer Teiler
- Primzahltest („PRIMES is in P" [Agrawal, Kayal, Saxena, 2002])

Definition von NTM

Definition (Nichtdeterministische Turingmaschine – NTM)

Eine nichtdeterministische Turingmaschine (NTM) ist definiert wie eine deterministische Turingmaschine (TM), nur die Zustandsüberführungsfunktion wird zu einer Relation

$$\delta \subseteq ((Q \setminus \{\bar{q}\}) \times \Gamma) \times (Q \times \Gamma \times \{L, R, N\}) \ .$$

- Eine Konfiguration K' ist direkter Nachfolger einer Konfiguration K, falls K' durch einen der in δ beschriebenen Übergänge aus K hervorgeht.

Erläuterung der Rechnung einer NTM

- Eine Konfiguration K' ist direkter Nachfolger einer Konfiguration K, falls K' durch einen der in δ beschriebenen Übergänge aus K hervorgeht.
- Rechenweg $=$ Konfigurationsfolge, die mit Startkonfiguration beginnt und mit Nachfolgekonfigurationen fortgesetzt wird bis sie eine Endkonfiguration im Zustand $\bar{q}$ erreicht.

Erläuterung der Rechnung einer NTM

- Eine Konfiguration K' ist direkter Nachfolger einer Konfiguration K, falls K' durch einen der in δ beschriebenen Übergänge aus K hervorgeht.
- Rechenweg $=$ Konfigurationsfolge, die mit Startkonfiguration beginnt und mit Nachfolgekonfigurationen fortgesetzt wird bis sie eine Endkonfiguration im Zustand $\bar{q}$ erreicht.
- Der Verlauf der Rechnung ist also *nicht deterministisch*, d.h., zu einer Konfiguration kann es mehrere direkte Nachfolgekonfigurationen geben.

Erläuterung der Rechnung einer NTM

Die möglichen Rechenwege von M für eine Eingabe $w \in \Sigma^*$ können in Form eines *Berechnungsbaumes* beschrieben werden:

- Die Knoten des Baumes entsprechen Konfigurationen.
- Die Wurzel des Baumes entspricht der Startkonfiguration.
- Die Kinder einer Konfiguration entsprechen den möglichen Nachfolgekonfigurationen.

Erläuterung der Rechnung einer NTM

Die möglichen Rechenwege von M für eine Eingabe $w \in \Sigma^*$ können in Form eines *Berechnungsbaumes* beschrieben werden:

- Die Knoten des Baumes entsprechen Konfigurationen.
- Die Wurzel des Baumes entspricht der Startkonfiguration.
- Die Kinder einer Konfiguration entsprechen den möglichen Nachfolgekonfigurationen.

Der *maximale Verzweigungsgrad* des Berechnungsbaumes ist

$$\Delta = \max\{|\delta(q,a)| \mid q \in Q \setminus \{\bar{q}\}, a \in \Gamma\}.$$

Beachte, dass Δ nicht von der Eingabe abhängt, also konstant ist.

Definition des Akzeptanzverhaltens

Definition (Akzeptanzverhalten der NTM)

Eine NTM M akzeptiert die Eingabe $x \in \Sigma^*$, falls es mindestens einen Rechenweg von M gibt, der in eine akzeptierende Endkonfiguration führt.

Die von M erkannte Sprache $L(M)$ besteht aus allen von M akzeptierten Wörtern.

Definition der Laufzeit

Definition (Laufzeit der NTM)

Sei M eine NTM. Die Laufzeit von M auf einer Eingabe $x \in L(M)$ ist definiert als

$$T_M(x) := \text{Länge des kürzesten akzeptierenden Rechenweges von } M \text{ auf } x \enspace .$$

Definition der Laufzeit

Definition (Laufzeit der NTM)

Sei M eine NTM. Die Laufzeit von M auf einer Eingabe $x \in L(M)$ ist definiert als

$$T_M(x) := \text{Länge des kürzesten akzeptierenden Rechenweges von } M \text{ auf } x \quad .$$

Für $x \notin L(M)$ definieren wir $T_M(x) = 0$.

Definition der Laufzeit

Definition (Laufzeit der NTM)

Sei M eine NTM. Die Laufzeit von M auf einer Eingabe $x \in L(M)$ ist definiert als

$$T_M(x) := \textit{Länge des kürzesten akzeptierenden Rechenweges von } M \textit{ auf } x \enspace .$$

Für $x \notin L(M)$ definieren wir $T_M(x) = 0$.

Die worst case Laufzeit $t_M(n)$ für M auf Eingaben der Länge $n \in \mathbb{N}$ ist definiert durch

$$t_M(n) := \max\{T_M(x) \mid x \in \Sigma^n\} \enspace .$$

Definition der Klasse NP

Definition (Komplexitätsklasse NP)

NP *ist die Klasse der Entscheidungsprobleme, die durch eine NTM* M *erkannt werden, deren worst case Laufzeit* $t_M(n)$ *polynomiell beschränkt ist.*

NP steht dabei für nichtdeterministisch polynomiell.

Beispiel für ein Problem aus NP

Problem (Cliquenproblem – CLIQUE)

Eingabe: *Graph* $G = (V, E)$, $k \in \{1, \ldots, |V|\}$
Frage: *Gibt es eine k-Clique?*

Beispiel für ein Problem aus NP

Problem (Cliquenproblem – CLIQUE)

Eingabe: *Graph* $G = (V, E)$, $k \in \{1, \ldots, |V|\}$
Frage: *Gibt es eine* k*-Clique?*

- Für das Cliquenproblem ist kein Polynomialzeitalgorithmus bekannt.
- Die besten bekannten Algorithmen haben eine exponentielle Laufzeit.

Beispiel für ein Problem aus NP

Satz

CLIQUE $\in$ NP.

Beweis: Wir beschreiben eine NTM M mit $L(M) = \text{CLIQUE}$:

Beispiel für ein Problem aus NP

Satz

CLIQUE $\in$ NP.

Beweis: Wir beschreiben eine NTM M mit $L(M) =$ CLIQUE:

1. Syntaktisch inkorrekte Eingaben werden verworfen.

Beispiel für ein Problem aus NP

Satz

$CLIQUE \in \mathrm{NP}$.

Beweis: Wir beschreiben eine NTM M mit $L(M) = \mathrm{CLIQUE}$:

1. Syntaktisch inkorrekte Eingaben werden verworfen.
2. M „rät" einen 0-1-String y der Länge $|V|$.

Satz

CLIQUE $\in$ NP.

Beweis: Wir beschreiben eine NTM M mit $L(M) = \text{CLIQUE}$:

1. Syntaktisch inkorrekte Eingaben werden verworfen.
2. M „rät" einen 0-1-String y der Länge $|V|$.
3. M akzeptiert, falls
   - der String y genau k viele Einsen enthält und
   - die Knotenmenge $C = \{i \in V \mid y_i = 1\}$ eine Clique ist.

Satz

CLIQUE $\in$ NP.

Beweis: Wir beschreiben eine NTM M mit $L(M) = \text{CLIQUE}$:

1. Syntaktisch inkorrekte Eingaben werden verworfen.
2. M „rät“ einen 0-1-String y der Länge $|V|$.
3. M akzeptiert, falls
   - der String y genau k viele Einsen enthält und
   - die Knotenmenge $C = \{i \in V \mid y_i = 1\}$ eine Clique ist.

Korrektheit: Es gibt genau dann einen akzeptierenden Rechenweg, wenn G eine k-Clique enthält.

Satz

CLIQUE $\in$ NP.

Beweis: Wir beschreiben eine NTM M mit $L(M) =$ CLIQUE:

1. Syntaktisch inkorrekte Eingaben werden verworfen.
2. M „rät" einen 0-1-String y der Länge $|V|$.
3. M akzeptiert, falls
   - der String y genau k viele Einsen enthält und
   - die Knotenmenge $C = \{i \in V \mid y_i = 1\}$ eine Clique ist.

Korrektheit: Es gibt genau dann einen akzeptierenden Rechenweg, wenn G eine k-Clique enthält.

Laufzeit: Alle Schritte haben polynomielle Laufzeit. $\square$

Definition (Komplexitätsklasse EXPTIME)

EXPTIME *ist die Klasse der Probleme, die sich auf einer DTM mit Laufzeitschranke* $2^{q(n)}$ *berechnen lassen, wobei* q *ein geeignetes Polynom ist.*

Definition (Komplexitätsklasse EXPTIME)

EXPTIME *ist die Klasse der Probleme, die sich auf einer DTM mit Laufzeitschranke* $2^{q(n)}$ *berechnen lassen, wobei* q *ein geeignetes Polynom ist.*

Wie verhält sich NP zu P und EXPTIME?

Wie verhält sich NP zu P und EXPTIME?

Im Folgenden schränken wir die Klassen P und EXPTIME auf Entscheidungsprobleme ein, um sie mit der Klasse NP in Beziehung setzen zu können.

Wie verhält sich NP zu P und EXPTIME?

Im Folgenden schränken wir die Klassen P und EXPTIME auf Entscheidungsprobleme ein, um sie mit der Klasse NP in Beziehung setzen zu können.

Satz

$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{EXPTIME}$

Wie verhält sich NP zu P und EXPTIME?

Im Folgenden schränken wir die Klassen P und EXPTIME auf Entscheidungsprobleme ein, um sie mit der Klasse NP in Beziehung setzen zu können.

Satz

P $\subseteq$ NP $\subseteq$ EXPTIME

Beweis:

Offensichtlich gilt P $\subseteq$ NP, weil eine DTM als eine spezielle NTM aufgefasst werden kann.

Wie verhält sich NP zu P und EXPTIME?

Im Folgenden schränken wir die Klassen P und EXPTIME auf Entscheidungsprobleme ein, um sie mit der Klasse NP in Beziehung setzen zu können.

Satz

$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{EXPTIME}$

Beweis:

Offensichtlich gilt $\mathrm{P} \subseteq \mathrm{NP}$, weil eine DTM als eine spezielle NTM aufgefasst werden kann.

Wir müssen noch zeigen $\mathrm{NP} \subseteq \mathrm{EXPTIME}$.

Exponentielle Laufzeitschranke für Probleme aus NP

Sei $L \in \mathrm{NP}$. Sei M eine NTM mit polynomiell beschränkter Laufzeitschranke $p(n)$, die L erkennt.

Exponentielle Laufzeitschranke für Probleme aus NP

Sei $L \in \mathrm{NP}$. Sei M eine NTM mit polynomiell beschränkter Laufzeitschranke $p(n)$, die L erkennt.

Sei $w \in \Sigma^*$. Wir konstruieren eine DTM M', die die NTM M mit Eingabe w simuliert:

Sei $L \in \mathrm{NP}$. Sei M eine NTM mit polynomiell beschränkter Laufzeitschranke $p(n)$, die L erkennt.

Sei $w \in \Sigma^*$. Wir konstruieren eine DTM M', die die NTM M mit Eingabe w simuliert:

- In einer Breitensuche generiert M' den Berechnungsbaum von M bis zu einer Tiefe von $p(|w|)$.
- Falls dabei eine akzeptierende Konfiguration gefunden wird, so akzeptiert M' die Eingabe; sonst verwirft M' die Eingabe.

Exponentielle Laufzeitschranke für Probleme aus NP

Korrektheit:

- Falls $w \in L$ ist, so gibt es einen akzeptierenden Rechenweg von M der Länge $p(|w|)$. M' generiert diesen Weg und akzeptiert w.
- Falls $w \notin L$ ist, so gibt es keinen akzeptierenden Rechenweg von M der Länge $p(|w|)$. In diesem Fall wird w von M' verworfen.

Korrektheit:

- Falls $w \in L$ ist, so gibt es einen akzeptierenden Rechenweg von M der Länge $p(|w|)$. M' generiert diesen Weg und akzeptiert w.
- Falls $w \notin L$ ist, so gibt es keinen akzeptierenden Rechenweg von M der Länge $p(|w|)$. In diesem Fall wird w von M' verworfen.

Laufzeit:

Sei $\Delta \geq 2$ der maximale Verzweigungsgrad der Rechung.

Die Laufzeit von M' auf w ist proportional zur Anzahl der Knoten im Berechnungsbaum bis zur Tiefe $p(|w|)$. Diese Anzahl ist beschränkt durch

Exponentielle Laufzeitschranke für Probleme aus NP

Korrektheit:

- Falls $w \in L$ ist, so gibt es einen akzeptierenden Rechenweg von M der Länge $p(|w|)$. M' generiert diesen Weg und akzeptiert w.
- Falls $w \notin L$ ist, so gibt es keinen akzeptierenden Rechenweg von M der Länge $p(|w|)$. In diesem Fall wird w von M' verworfen.

Laufzeit:

Sei $\Delta \geq 2$ der maximale Verzweigungsgrad der Rechung.

Die Laufzeit von M' auf w ist proportional zur Anzahl der Knoten im Berechnungsbaum bis zur Tiefe $p(|w|)$. Diese Anzahl ist beschränkt durch

$$\Delta^{p(|w|)+1}$$

Exponentielle Laufzeitschranke für Probleme aus NP

Korrektheit:

- Falls $w \in L$ ist, so gibt es einen akzeptierenden Rechenweg von M der Länge $p(|w|)$. M' generiert diesen Weg und akzeptiert w.
- Falls $w \notin L$ ist, so gibt es keinen akzeptierenden Rechenweg von M der Länge $p(|w|)$. In diesem Fall wird w von M' verworfen.

Laufzeit:

Sei $\Delta \geq 2$ der maximale Verzweigungsgrad der Rechung.

Die Laufzeit von M' auf w ist proportional zur Anzahl der Knoten im Berechnungsbaum bis zur Tiefe $p(|w|)$. Diese Anzahl ist beschränkt durch

$$\Delta^{p(|w|)+1} = 2^{(p(|w|)+1)\cdot\log_2 \Delta}$$

Korrektheit:

- Falls $w \in L$ ist, so gibt es einen akzeptierenden Rechenweg von M der Länge $p(|w|)$. M' generiert diesen Weg und akzeptiert w.
- Falls $w \notin L$ ist, so gibt es keinen akzeptierenden Rechenweg von M der Länge $p(|w|)$. In diesem Fall wird w von M' verworfen.

Laufzeit:

Sei $\Delta \geq 2$ der maximale Verzweigungsgrad der Rechung.

Die Laufzeit von M' auf w ist proportional zur Anzahl der Knoten im Berechnungsbaum bis zur Tiefe $p(|w|)$. Diese Anzahl ist beschränkt durch

$$\Delta^{p(|w|)+1} = 2^{(p(|w|)+1)\cdot\log_2 \Delta} = 2^{O(p(|w|))} .$$

Die große offene Frage der Informatik lautet

$$P = NP?$$

Diese Frage ist so bedeutend, weil sehr viele wichtige Probleme in NP enthalten sind, für die kein Polynomialzeitalgorithmus bekannt ist. Einige dieser Problem lernen wir in der nächsten Vorlesung kennen.