max planck institut
informatik

Komplexitätstheorie
P versus NP

Kurt Mehlhorn und Adrian Neumann

Max Planck Institute for Informatics and Saarland University

6. Januar 2014

Gliederung

- Komplexitätstheorie und die Komplexitätslandschaft
- Informelle Formulierung der P = NP Frage
- Das Erfüllbarkeitsproblem der Aussagenlogik (SATisfiability Problem): das prototypische Problem in NP
- Ausdruckskraft von SAT
- P und NP, formale Definition
- NP-Vollständigkeit
- Was wäre, wenn P $\neq$ NP?
- Was wäre, wenn P $=$ NP?
- Wie geht man mit NP-Vollständigkeit um?

Ziele der Komplexitätstheorie

- schaffe Ordnung
- Klassifiziere Probleme nach Schwierigkeit: Problem A ist nicht schwerer als Problem B
- Verstehe Resourcen: ist Platz wertvoller als Zeit?
- Verstehe die relative Kraft von Berechnungsmodellen: deterministisch, randomisiert (nutzt es, wenn man Münzen werfen kann), Quantenrechner
- Untere Schranken: Um n Objekte zu sortieren, braucht man $n \log n$ Vergleiche.
- Verstehe das Konzept Beweis

P = Die Klasse der gutmütigen Probleme

P = Menge, der in Polynomzeit lösbaren Probleme

Ein Problem (eine Aufgabe) ist in Polynomzeit lösbar, wenn es eine Turingmaschine (einen Algorithmus) gibt, der
das Problem löst, d.h., für jede konkrete Problemstellung die richtige Anwort liefert, und
dessen Laufzeit an einer Problemstellung x beschränkt ist durch $k + |x|^k$;
dabei ist k eine feste natürliche Zahl unabhängig von x und $|x|$ die Länge von x (Anzahl der Zeichen)

P = Die Klasse der gutmütigen Probleme

P = Menge, der in Polynomzeit lösbaren Probleme

Ein Problem (eine Aufgabe) ist in Polynomzeit lösbar, wenn es eine Turingmaschine (einen Algorithmus) gibt, der
das Problem löst, d.h., für jede konkrete Problemstellung die richtige Anwort liefert, und
dessen Laufzeit an einer Problemstellung x beschränkt ist durch $k + |x|^k$;
dabei ist k eine feste natürliche Zahl unabhängig von x und $|x|$ die Länge von x (Anzahl der Zeichen)

P = Die Klasse der gutmütigen Probleme

P = Menge, der in Polynomzeit lösbaren Probleme

Ein Problem (eine Aufgabe) ist in Polynomzeit lösbar, wenn es eine Turingmaschine (einen Algorithmus) gibt, der
das Problem löst, d.h., für jede konkrete Problemstellung die richtige Anwort liefert, und
dessen Laufzeit an einer Problemstellung x beschränkt ist durch $k + |x|^k$;
dabei ist k eine feste natürliche Zahl unabhängig von x und $|x|$ die Länge von x (Anzahl der Zeichen)

P = Die Klasse der gutmütigen Probleme

P = Menge, der in Polynomzeit lösbaren Probleme

Ein Problem (eine Aufgabe) ist in Polynomzeit lösbar, wenn es eine Turingmaschine (einen Algorithmus) gibt, der
das Problem löst, d.h., für jede konkrete Problemstellung die richtige Anwort liefert, und
dessen Laufzeit an einer Problemstellung x beschränkt ist durch $k + |x|^k$;
dabei ist k eine feste natürliche Zahl unabhängig von x und $|x|$ die Länge von x (Anzahl der Zeichen)

$k = 1$, lineare Laufzeit, $k = 2$, quadratische Laufzeit,...

Beispiele: kürzeste Wege, maximale Flüsse, Sortieren, Suchen

P = Die Klasse der gutmütigen Probleme

P = Menge, der in Polynomzeit lösbaren Probleme

Ein Problem (eine Aufgabe) ist in Polynomzeit lösbar, wenn es eine Turingmaschine (einen Algorithmus) gibt, der
das Problem löst, d.h., für jede konkrete Problemstellung die richtige Anwort liefert, und
dessen Laufzeit an einer Problemstellung x beschränkt ist durch $k + |x|^k$;
dabei ist k eine feste natürliche Zahl unabhängig von x und $|x|$ die Länge von x (Anzahl der Zeichen)

Glaubenssatz der Informatik: nur Algorithmen mit polynomieller Laufzeit sind brauchbare Algorithmen: Verdoppelung der Problemgröße vergrößert Laufzeit von $k + |x|^k$ auf $k + (2|x|)^k \leq 2^k(k + |x|^k)$, also nur um den konstanten Faktor 2^k

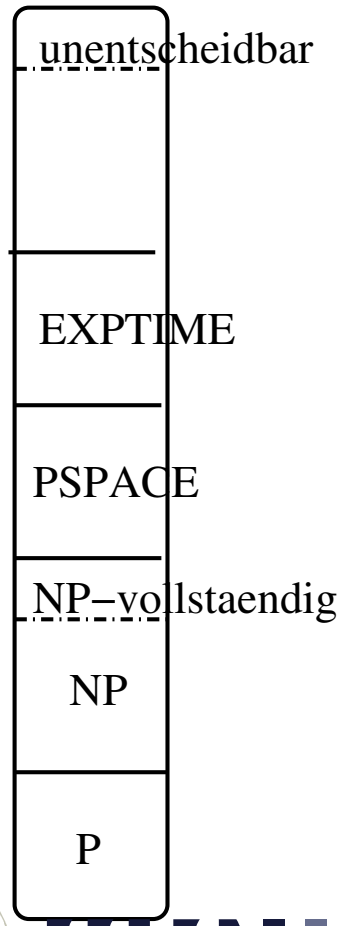
P = Die Klasse der gutmütigen Probleme

P = Menge, der in Polynomzeit lösbaren Probleme

Ein Problem (eine Aufgabe) ist in Polynomzeit lösbar, wenn es eine Turingmaschine (einen Algorithmus) gibt, der
das Problem löst, d.h., für jede konkrete Problemstellung die richtige Anwort liefert, und
dessen Laufzeit an einer Problemstellung x beschränkt ist durch $k + |x|^k$;
dabei ist k eine feste natürliche Zahl unabhängig von x und $|x|$ die Länge von x (Anzahl der Zeichen)

exponentielle Laufzeit $c^{|x|}$: Verdoppelung der Problemgröße vergrößert Laufzeit von $c^{|x|}$ auf $c^{2|x|} = (c^{|x|})^2$, d.h. quadriert die Laufzeit.
Wenn ich bei einer bestimmten Problemgröße 10^9 Schritte (1 sec) brauche, dann bei der doppelten Größe 10^{18} Schritte (10^9 sec $\approx$ 31 Jahre)

Die Komplexitätslandschaft



$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq$ alle Probleme

P = gutmütige Probleme, z.B., kürzeste Wege, in Polynomzeit lösbar

NP = mit Raten in Polynomzeit lösbar, z.B. Erfüllbarkeitsproblem der Aussagenlogik

PSPACE = mit polynomiellem Speicherplatz lösbar

EXPTIME = in exponentieller Zeit lösbar

NP-vollständig = schwerste Probleme in NP

unentscheidbar = nicht maschinell lösbar, z.B. Halteproblem

Die P = NP Frage (Geschichte)

- P = Menge aller Probleme, die in Polynomzeit lösbar sind
- NP = Menge aller Probleme, die man mit Raten in Polynomzeit lösen kann (deren Lösung in Problemzeit überprüft werden kann)
- $P \subseteq NP$ is offensichtlich
- die große Frage ist: $P = NP$?
- wurde von Cook, Levin und Karp in 71, 72 formuliert.
- Clay Foundation 1 Mio $ Preisgeld für Lösung (eines der 6 großen offenen Probleme der Mathematik)
- die Frage ist von grundlegender philosophischer/mathematischer Bedeutung

  Gleichheit hätte enorme algorithmische Konsequenzen

Was ist die P = NP Frage?

Die P = NP Frage (informelle Formulierung)

Gibt es Probleme (Aufgaben), für die es schwieriger ist, eine Lösung zu finden als eine Lösung zu überprüfen?

Antwort: natürlich,

- meine Analysisübungsblätter, Sudoku und Kreuzworträtsel

Was ist die P = NP Frage?

Die P = NP Frage (informelle Formulierung)

Gibt es Probleme (Aufgaben), für die es schwieriger ist, eine Lösung zu finden als eine Lösung zu überprüfen?

Antwort: natürlich,

- meine Analysisübungsblätter, Sudoku und Kreuzworträtsel

Was ist die P = NP Frage?

Die P = NP Frage (informelle Formulierung)

Gibt es Probleme (Aufgaben), für die es schwieriger ist, eine Lösung zu finden als eine Lösung zu überprüfen?

Antwort: natürlich,

- meine Analysisübungsblätter, Sudoku und Kreuzworträtsel
- und nun Probleme, die es in beliebiger Größe gibt:
- Rucksackproblem: gegeben n Paare (g_i, w_i) von natürlichen Zahlen (Gewicht, Wert) und zwei Zielzahlen G und W. Gibt es eine Teilmenge $I \subseteq [1..n]$, so dass

$$\sum_{i \in I} g_i \leq G \quad \text{and} \quad \sum_{i \in I} w_i \geq W?$$

Was ist die P = NP Frage?

Die P = NP Frage (informelle Formulierung)

Gibt es Probleme (Aufgaben), für die es schwieriger ist, eine Lösung zu finden als eine Lösung zu überprüfen?

Antwort: natürlich,

- meine Analysisübungsblätter, Sudoku und Kreuzworträtsel
- und nun Probleme, die es in beliebiger Größe gibt:
- Problem des Handlungsreisenden: gibt es eine Tour durch die alle Orte Deutschlands mit mehr als 5 Tausend Einwohnern, die höchstens 4000 Kilometer lang ist? Allgemein:

  Gegeben eine Funktion $d : [1..n] \times [1..n] \to \mathbb{N}$ und eine Zielzahl $L \in \mathbb{N}$, gibt es eine Anordnung $i_1, i_2, \ldots, i_n$ von $[1..n]$, so dass

$$\sum_{1 \le j < n} d_{i_j, i_{j+1}} + d_{i_n, i_1} \le L?$$

SAT: das prototypische Problem in NP

Das Erfüllbarkeitsproblem (SATisfiability-Problem)

Eingabe: Eine Formel der Aussagenlogik

Frage: Ist die Formel erfüllbar, d.h., gibt es eine Belegung der Variablen mit Wahrheitswerten Wahr (W) und Falsch (F), die die Formel erfüllt?

Formeln der Aussagenlogik = Variablen verknüpft mit und ($\wedge$), oder ($\vee$) und Negation ($\neg$). Definition auf nächster Folie

Beispiel

Formel: $(x \vee y) \wedge \neg x$

Belegung 1: $x \to W$, $y \to F$, dann $(W \vee F) \wedge \neg W = W \wedge F = F$

Belegung 2: $x \to F$, $y \to W$, dann $(F \vee W) \wedge \neg F = W \wedge W = W$

SAT: das prototypische Problem in NP

Das Erfüllbarkeitsproblem (SATisfiability-Problem)

Eingabe: Eine Formel der Aussagenlogik

Frage: Ist die Formel erfüllbar, d.h., gibt es eine Belegung der Variablen mit Wahrheitswerten Wahr (W) und Falsch (F), die die Formel erfüllt?

Formeln der Aussagenlogik = Variablen verknüpft mit und ($\wedge$), oder ($\vee$) und Negation ($\neg$). Definition auf nächster Folie

Die P = NP Frage (erste Formalisierung)

Gibt es einen Polynomzeitalgorithmus für das Erfüllbarkeitsproblem?

Formeln der Aussagenlogik

(1) W (true, wahr), F (falsch, false) und Variablen sind Formeln.
(2) Wenn F und G Formeln sind, dann auch $(F \wedge G)$, $(F \vee G)$, und $\neg F$.
(3) Das ist alles.

Belegung, Wert einer Formel, erfüllbar

Eine Belegung weist jeder Variablen einen Wahrheitswert zu.

Der Wert der Formel ergibt sich nach den Berechnungsregeln der Aussagenlogik: $F \vee F = F$, $F \vee W = T \vee F = W \vee W = W$, $F \wedge F = F \wedge W = W \wedge F = F$, $W \wedge W = W$, $\neg F = W$ und $\neg W = F$.

Eine Formel ist erfüllbar, wenn es eine Belegung gibt, unter der sie den Wert wahr erhält.

Die P = NP Frage

Gibt es Polynomzeitalgorithmus für das Erfüllbarkeitsproblem?

- **Verbindung mit informeller Formulierung**
  - Lösung überprüfen: ist eine gegebene Belegung erfüllend?
  - Lösung finden: finde eine erfüllende Belegung.
  - Erfüllbarkeitsproblem ist Kandidat für ein Problem, bei dem Prüfen einfacher ist als Finden.
- **Warum ist das Erfüllbarkeitsproblem so wichtig?**
  - wenn es ein Problem gibt, bei dem Prüfen einfacher ist als Finden, dann ist das Erfüllbarkeitsproblem ein solches Problem.
- **Wofür steht NP? Was bedeutet Raten?**

Bedeutung des Erfüllbarkeitproblems

Satz

Wenn man einen Polynomzeitalgorithmus für das Erfüllbarkeitsproblem kennt, dann kennt man auch Polynomzeitalgorithmen für

1. *Graphenfärbung*
2. *Hamiltonscher Kreis*
3. *Problem des Handlungsreisenden*
4. *Rucksackproblem*
5. *Partition*
6. *Sudoku*
7. *....*
8. *jedes Problem in NP*

Ich beweise 1 und 2; 8 erhält man durch Verallgemeinerung

Graphenfärbung (mit drei Farben)

Eingabe: Ein ungerichteter Graph $G = (V, E)$

Frage: Gibt es eine Dreifärbung der Knoten von G, d.h. eine Abbildung $f : V \to \{R, B, G\}$, so dass für jede Kante $\{u, v\} \in E$ gilt: $f(u) \neq f(v)$.

Reduktion: Färbung $\leq$ SAT

Variable $x_{u,c}$ für Knoten $u \in V$ und Farbe $c \in \{R, B, G\}$.

Intendierte Bedeutung: $x_{u,c} = W$ bedeutet u hat die Farbe c.

Formel:

$$\bigwedge_{u \in V} GE(x_{u,R}, x_{u,B}, x_{u,G}) \wedge \bigwedge_{\{u,v\} \in E} \bigwedge_{c \in \{R,B,G\}} \neg(x_{u,c} \wedge x_{v,c})$$

Dabei ist $GE(x_1, \ldots, x_n) = (\bigvee_i x_i) \wedge \neg(\bigvee_{i \neq j}(x_i \wedge x_j))$. GenauEine

Reduktion: Färbung $\leq$ SAT

Variable $x_{u,c}$ für Knoten $u \in V$ und Farbe $c \in \{R, B, G\}$.

Intendierte Bedeutung: $x_{u,c} = W$ bedeutet u hat die Farbe c.

$$\bigwedge_{u \in V} GE(x_{u,R}, x_{u,B}, x_{u,G}) \wedge \bigwedge_{\{u,v\} \in E} \bigwedge_{c \in \{R,B,G\}} \neg(x_{u,c} \wedge x_{v,c})$$

Korrektheit der Reduktion

Sei $f : V \to \{R, B, G\}$ eine legale Färbung. Setze $x_{u,c} = W$ genau wenn $f(u) = c$. Diese Belegung erfüllt die Formel.

Reduktion: Färbung $\leq$ SAT

Variable $x_{u,c}$ für Knoten $u \in V$ und Farbe $c \in \{R, B, G\}$.

Intendierte Bedeutung: $x_{u,c} = W$ bedeutet u hat die Farbe c.

$$\bigwedge_{u \in V} GE(x_{u,R}, x_{u,B}, x_{u,G}) \wedge \bigwedge_{\{u,v\} \in E} \bigwedge_{c \in \{R,B,G\}} \neg(x_{u,c} \wedge x_{v,c})$$

Korrektheit der Reduktion

Sei b eine erfüllende Belegung. Definiere $f(u) = c$ genau wenn $b(x_{u,c}) = W$.

Da die Belegung $GE(x_{u,R}, x_{u,B}, x_{u,G})$ erfüllt, ist f wohldefiniert.

Betrachte eine beliebige Kante $\{u, v\}$: Da die Belegung $\bigwedge_{c \in \{R,B,G\}} \neg(x_{u,c} \wedge x_{v,c})$ erfüllt, haben u und v nicht die gleiche Farbe.

Reduktion: Färbung $\leq$ SAT

Variable $x_{u,c}$ für Knoten $u \in V$ und Farbe $c \in \{R, B, G\}$.

Intendierte Bedeutung: $x_{u,c} = W$ bedeutet u hat die Farbe c.

$$\bigwedge_{u \in V} GE(x_{u,R}, x_{u,B}, x_{u,G}) \wedge \bigwedge_{\{u,v\} \in E} \bigwedge_{c \in \{R,B,G\}} \neg(x_{u,c} \wedge x_{v,c})$$

Effizienz der Reduktion

$n = |V|$, $m = |E|$. Die Formel hat Länge $O((n+m) \log n)$, da

es für jeden Knoten und jede Kante eine Teilformel konstanter Größe gibt.

das $\log n$ trägt der Tatsache Rechnung, dass die Variablennamen logarithmische Länge haben.

Färbung $\leq$ SAT: Zusammenfassung

- aus $G = (V, E)$ konstruieren wir eine Formel $\varphi(G)$ mit
  - G ist dreifärbbar genau wenn $\varphi(G)$ erfüllbar
  - die Länge von $\varphi(G)$ is höchstens $(n+m)\log n$.
  - $\varphi(G)$ kann aus G in Polynomzeit konstruiert werden Übung
- nimm nun an, dass es einen Polynomzeitalgorithmus A für SAT gibt: Laufzeit auf Eingabe φ ist $k + |\varphi|^k$ für ein $k \in \mathbb{N}$.
- wende A auf $\varphi(G)$ an
- A entscheidet Dreifärbbarkeit von G
- Laufzeit auf Eingabe G ist Zeit für die Konstruktion von $\varphi(G)$ plus

$$k + ((n+m)\log n)^k \leq 2k + (n+m)^{2k},$$

also polynomiell in der Größe von G.

Weitere Probleme $\leq$ SAT

Hamiltonscher Kreis

Eingabe: Ein gerichteter Graph $G = (V, E)$

Frage: Gibt es einen Kreis, der alle Knoten genau einmal durchläuft? d.h. eine Bijektion $\pi : [1..n] \to V$, so dass $(\pi(k), \pi(k+1)) \in E$ für $1 \leq k \leq n$ (dabei steht $\pi(n+1)$ für $\pi(1)$).

Variable $x_{k,i}$ und $y_{i,j}$, wobei $i, j, k \in [1..n]$ und $V = \{1, \ldots, n\}$

$\varphi(G)$ ist die Konjunktion der folgenden Formeln:

- $\bigwedge_{ij \in E} y_{ij} \wedge \bigwedge_{ij \notin E} \neg y_{ij}$ Kodierung des Graphen
- x kodiert eine Bijektion π durch $\pi(k) = i$ genau wenn $x_{ki} = T$.
  $\bigwedge_{k \in [1..n]} GE(x_{k1}, \ldots, x_{kn}) \wedge \bigwedge_{i \in V} GE(x_{1i}, \ldots, x_{ni})$
- $(\pi(i), \pi(i+1)) \in E$ für $1 \leq i \leq n$ (dabei steht $\pi(n+1)$ für $\pi(1)$)
  $\bigwedge_{k,i,j}((x_{k,i} \wedge x_{k+1,j}) \to y_{i,j})$ $n+1$ steht für 1.

Weitere Probleme $\leq$ SAT

Hamiltonscher Kreis

Eingabe: Ein gerichteter Graph $G = (V, E)$

Frage: Gibt es einen Kreis, der alle Knoten genau einmal durchläuft? d.h. eine Bijektion $\pi : [1..n] \to V$, so dass $(\pi(k), \pi(k+1)) \in E$ für $1 \leq k \leq n$ (dabei steht $\pi(n+1)$ für $\pi(1)$).

Variable $x_{k,i}$ und $y_{i,j}$, wobei $i, j, k \in [1..n]$ und $V = \{1, \ldots, n\}$

$\varphi(G)$ ist die Konjunktion der folgenden Formeln:

- $\bigwedge_{ij \in E} y_{ij} \wedge \bigwedge_{ij \notin E} \neg y_{ij}$ Kodierung des Graphen
- x kodiert eine Bijektion π durch $\pi(k) = i$ genau wenn $x_{ki} = T$.
  $\bigwedge_{k \in [1..n]} GE(x_{k1}, \ldots, x_{kn}) \wedge \bigwedge_{i \in V} GE(x_{1i}, \ldots, x_{ni})$
- $(\pi(i), \pi(i+1)) \in E$ für $1 \leq i \leq n$ (dabei steht $\pi(n+1)$ für $\pi(1)$)
  $\bigwedge_{k,i,j}((x_{k,i} \wedge x_{k+1,j}) \to y_{i,j})$ $n+1$ steht für 1.

Problem

Sei Σ ein festes Alphabet. Eine Teilmenge von Σ^* heißt Problem.

Wir brauchen nun eine formale Definition des Begriffs Problems.

P (Polynomzeitentscheidbar)

Ein Problem L gehört zu P, wenn es eine polynomzeit-beschränkte Turingmaschine M gibt, die L einscheidet.

An einer beliebigen Eingabe x gibt M entweder JA oder NEIN aus und es gilt $x \in L$ genau wenn die Ausgabe JA ist.

Laufzeit ist durch Polynom beschränkt.

P (Polynomzeitentscheidbar)

Ein Problem L gehört zu P, wenn es eine polynomzeit-beschränkte Turingmaschine M gibt, die L einscheidet.

NP (Polynomzeitentscheidbar mit Raten)

Ein Problem L gehört zu NP, wenn es ein Problem L' in P und ein Polynom q gibt, so dass gilt:

$$x \in L \text{ genau wenn es ein } w \in (\Sigma \setminus \#)^* \text{ gibt, so dass } |w| \leq q(|x|) \text{ und } x\#w \in L'.$$

wir sagen: w bezeugt (beweist) die Mitgliedschaft von x oder auch w ist eine Lösung von x.

man erhält L aus L' wie folgt: Betrachte ein beliebiges Wort in L'. Falls es kein # enthält, trägt es nicht zu L bei. Anderfalls schreibe das Wort als $x\#w$, wobei w kein # enthält. Falls $|w| \leq q(|x|)$, dann $x \in L$.

NP (Polynomzeitentscheidbar mit Raten)

Ein Problem L gehört zu NP, wenn es ein Problem L' in P und ein Polynom q gibt, so dass gilt:

$$x \in L \text{ genau wenn es ein } w \in (\Sigma \setminus \#)^* \text{ gibt, so dass } |w| \leq q(|x|) \text{ und } x\#w \in L'.$$

wir sagen: w bezeugt (beweist) die Mitgliedschaft von x oder auch w ist eine Lösung von x.

Rechenmodell: man errät ein w mit $|w| \leq q(|x|)$ und benutzt dann den Algorithmus für L';
alternativ: ein allmächtiges Helferlein verrät w und dann ...

Beispiele für Probleme in NP

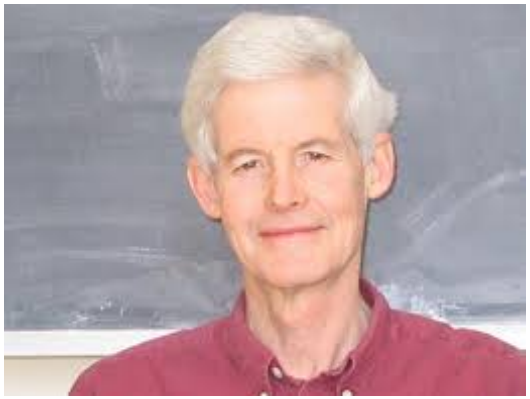
1. Erfüllbarkeitsproblem: $x =$ aussagenlogische Formel, $w =$ erfüllende Belegung
2. Knapsackproblem: $x =$ Problemstellung, $w =$ leichte Teilmenge mit hohem Wert
3. Nichtprimzahlen: $x =$ eine natürliche Zahl, $w =$ eine nicht-triviale Faktorisierung
4. Sudoku: $x =$ Spielplan, $w =$ Lösung

SAT ist ein schwerstes Problem in NP

Satz (Stephen Cook und Leonid Levin, 71)

Falls es einen Polynomzeitalgorithmus für das Erfüllbarkeitsproblem gibt, dann gibt es einen Polynomzeitalgorithmus für jedes Problem in NP.

Beweisidee: man gibt eine Reduktion an ähnlich zu Graphenfärbung und Hamiltonscher Kreis.



Cook: Turing Award
Levin: Knuth Prize.

P versus NP

P $\subseteq$ NP

Sei L in P beliebig. Definiere $L' = \{ x\#; x \in L \}$. Dann ist sicher $L' \in$ P. Also gilt $L \in$ NP.

Die P = NP Frage (zweite Formalisierung)

Ist P = NP?

Die P = NP Frage (informelle Formulierung)

Gibt es Probleme, für die es schwieriger ist, eine Lösung zu finden als eine Lösung zu überprüfen?

P versus NP

P $\subseteq$ NP

Sei L in P beliebig. Definiere $L' = \{ x\#; x \in L \}$. Dann ist sicher $L' \in$ P. Also gilt $L \in$ NP.

Die P = NP Frage (zweite Formalisierung)

Ist P = NP?

Die P = NP Frage (informelle Formulierung)

Gibt es Probleme, für die es schwieriger ist, eine Lösung zu finden als eine Lösung zu überprüfen?

NP-Vollständigkeit

Definition

Ein Problem L in NP ist NP-vollständig, wenn aus $L \in P$ folgt $P = NP$.

Cook-Levine bewiesen, dass das Erfüllbarkeitsproblem NP-vollständig ist.

Satz (Karp, 1972)

Das Graphenfärbungsproblem, das Hamiltonsche Kreisproblem, Knapsack, und 20 andere Probleme sind NP-vollständig.

Die Liste ist inzwischen auf mehrere Tausend angewachsen

Richard Karp erhielt den Turing-Award in 1985.

War wäre, wenn P $\neq$ NP?

- es würde sich nicht viel ändern.
- da wir keinen Polynomzeitalgorithmus für das Erfüllbarkeitsproblem kennen, leben wir faktisch in einer Welt, in der P ungleich NP ist.
- die meisten Fachleute glauben, dass P $\neq$ NP?
- aber: im Augenblick gibt es keinen Ansatz, wie man P $\neq$ NP beweisen könnte. Man weiß nur, dass einige natürliche Ansätze NICHT funktionieren können. Wenn man einen Beweis findet, muss dieser eine neue Methode einführen. Diese Methode könnte weitere Anwendungen haben.
- alle paar Jahre wird ein (falscher) Beweis angekündigt.

War wäre, wenn P = NP?

- das wäre eine Revolution
- wir hätten Polynomzeitalgorithmen für Erfüllbarkeit, . . . ,
- Mathematiker würden arbeitslos:

  Input: ein mathematischer Satz *S*, eine Anzahl *n* unbeschriebener Blätter

  Frage: gibt es einen Beweis für *S* (in einem formalen System), der auf die *n* Blätter passt?

  Dieses Problem ist in NP. Falls P = NP, dann ist dieses Problem in P.
- Philosophen müssten neu über den Begriff Kreativität nachdenken.
- alle paar Monate wird ein (falscher) Beweis angekündigt.

Wie geht man mit NP-Vollständigkeit um?

- in den 70ern war NP-Vollständigkeit ein Killerargument (mit dem Problem braucht man sich nicht zu beschäftigen), jetzt sieht man es als Herausforderung. Folgende Ansätze gibt es:
- Heuristiken
- Exakte Algorithmen für kleine n
- Speziallfälle
- Approximationsalgorithmen