Die Eine-Million-Dollar Frage
Ist P = NP?
Welche Probleme können Rechner effizient lösen?

Ideen der Informatik

Kurt Mehlhorn

max planck institut informatik

SIC Saarland Informatics Campus

Ein Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Die heutigen Themen

- Ziele von Theorie
- Gibt es Probleme, die man prinzipiell NICHT mit einem Rechner lösen kann?
- Welche Probleme kann man effizient lösen?
  - Finden: P = Menge der Probleme, bei denen eine Lösung in Polynomzeit gefunden werden kann.
  - Überprüfen: NP = Menge aller Ja-/Nein-Probleme, bei denen ein Lösungsvorschlag für Ja-Instanzen in Polynomzeit überprüft werden kann.
  - Die Eine-Million-Dollar Frage: Ist P = NP?
  - Ist Finden schwerer als Überprüfen?
- Das Erfüllbarkeitsproblem der Aussagenlogik (SATisfiability Problem): das prototypische Problem für NP.
- Satz von Cook-Levin: Falls SAT $\in$ P, dann NP = P.
- Was wäre, wenn P $\neq$ NP? Was wäre, wenn P $=$ NP?

Die heutigen Themen

- Ziele von Theorie
- Gibt es Probleme, die man prinzipiell NICHT mit einem Rechner lösen kann?
- Welche Probleme kann man effizient lösen?
  - Finden: P = Menge der Probleme, bei denen eine Lösung in Polynomzeit gefunden werden kann.
  - Überprüfen: NP = Menge aller Ja-/Nein-Probleme, bei denen ein Lösungsvorschlag für Ja-Instanzen in Polynomzeit überprüft werden kann.
  - Die Eine-Million-Dollar Frage: Ist P = NP?
  - Ist Finden schwerer als Überprüfen?
- Das Erfüllbarkeitsproblem der Aussagenlogik (SATisfiability Problem): das prototypische Problem für NP.
- Satz von Cook-Levin: Falls SAT $\in$ P, dann NP = P.
- Was wäre, wenn P $\neq$ NP? Was wäre, wenn P $=$ NP?

Die heutigen Themen

- Ziele von Theorie
- Gibt es Probleme, die man prinzipiell NICHT mit einem Rechner lösen kann?
- Welche Probleme kann man effizient lösen?
  - Finden: P = Menge der Probleme, bei denen eine Lösung in Polynomzeit gefunden werden kann.
  - Überprüfen: NP = Menge aller Ja-/Nein-Probleme, bei denen ein Lösungsvorschlag für Ja-Instanzen in Polynomzeit überprüft werden kann.
  - Die Eine-Million-Dollar Frage: Ist P = NP?
  - Ist Finden schwerer als Überprüfen?
- Das Erfüllbarkeitsproblem der Aussagenlogik (SATisfiability Problem): das prototypische Problem für NP.
- Satz von Cook-Levin: Falls SAT $\in$ P, dann NP = P.
- Was wäre, wenn P $\neq$ NP? Was wäre, wenn P $=$ NP?

Die heutigen Themen

- Ziele von Theorie
- Gibt es Probleme, die man prinzipiell NICHT mit einem Rechner lösen kann?
- Welche Probleme kann man effizient lösen?
  - Finden: P = Menge der Probleme, bei denen eine Lösung in Polynomzeit gefunden werden kann.
  - Überprüfen: NP = Menge aller Ja-/Nein-Probleme, bei denen ein Lösungsvorschlag für Ja-Instanzen in Polynomzeit überprüft werden kann.
  - Die Eine-Million-Dollar Frage: Ist P = NP?
  - Ist Finden schwerer als Überprüfen?
- Das Erfüllbarkeitsproblem der Aussagenlogik (SATisfiability Problem): das prototypische Problem für NP.
- Satz von Cook-Levin: Falls SAT $\in$ P, dann NP = P.
- Was wäre, wenn P $\neq$ NP? Was wäre, wenn P $=$ NP?

Die heutigen Themen

- Ziele von Theorie
- Gibt es Probleme, die man prinzipiell NICHT mit einem Rechner lösen kann?
- Welche Probleme kann man effizient lösen?
  - Finden: P = Menge der Probleme, bei denen eine Lösung in Polynomzeit gefunden werden kann.
  - Überprüfen: NP = Menge aller Ja-/Nein-Probleme, bei denen ein Lösungsvorschlag für Ja-Instanzen in Polynomzeit überprüft werden kann.
  - Die Eine-Million-Dollar Frage: Ist P = NP?
  - Ist Finden schwerer als Überprüfen?
- Das Erfüllbarkeitsproblem der Aussagenlogik (SATisfiability Problem): das prototypische Problem für NP.
- Satz von Cook-Levin: Falls SAT $\in$ P, dann NP = P.
- Was wäre, wenn P $\neq$ NP? Was wäre, wenn P $=$ NP?

Ziele von Theorie: hier Komplexitätstheorie

Ziel einer jeden Theorie: Schaffe Einsicht und Ordnung, erlaube Vorhersagen.

Heute:

- Grenzen der Berechenbarkeit.
- Erfüllbarkeitsproblem der Aussagenlogik, Rucksackproblem, Problem des Handlungsreisenden, Graphenfärbung und tausend andere Probleme sind alle gleich schwer: Falls es für eines dieser Probleme einen effizienten Algorithmus gibt, dann für alle.
- Es gibt einen effizienten Algorithmus für eines dieser (alle diese) Probleme, wenn zwischen Beweisen und Überprüfen eines Beweises kein wesentlicher Unterschied besteht. Das ist die P = NP Frage.

Grenzen der Berechenbarkeit

Gibt es ein wohldefiniertes Problem, das man **NICHT** mit einem Rechner (Synonym: mechanisch) lösen kann?

Rechner werden nie intelligent sein.

- Was bedeutet es genau, intelligent zu sein?
- Menschenaffen werden nie Sprache benutzen, Werkzeuge benutzen, Nutzung von Werkzeugen weitergeben, Emotionen zeigen, ...

Halteproblem

Eingabe: Ein Programm M und eine Eingabe x.

Frage: Hält M an der Eingabe x? Ja/Nein.

Satz: Es gibt kein Programm für das Halteproblem.

Grenzen der Berechenbarkeit

Gibt es ein wohldefiniertes Problem, das man **NICHT** mit einem Rechner (Synonym: mechanisch) lösen kann?

Rechner werden nie intelligent sein.

- Was bedeutet es genau, intelligent zu sein?
- Menschenaffen werden nie Sprache benutzen, Werkzeuge benutzen, Nutzung von Werkzeugen weitergeben, Emotionen zeigen, ...

Halteproblem

Eingabe: Ein Programm M und eine Eingabe x.

Frage: Hält M an der Eingabe x? Ja/Nein.

Satz: Es gibt kein Programm für das Halteproblem.

Grenzen der Berechenbarkeit

Gibt es ein wohldefiniertes Problem, das man **NICHT** mit einem Rechner (Synonym: mechanisch) lösen kann?

Rechner werden nie intelligent sein.

- Was bedeutet es genau, intelligent zu sein?
- Menschenaffen werden nie Sprache benutzen, Werkzeuge benutzen, Nutzung von Werkzeugen weitergeben, Emotionen zeigen, ...

Halteproblem

Eingabe: Ein Programm M und eine Eingabe x.

Frage: Hält M an der Eingabe x? Ja/Nein.

Satz: Es gibt kein Programm für das Halteproblem.

Grenzen der Berechenbarkeit

Gibt es ein wohldefiniertes Problem, das man **NICHT** mit einem Rechner (Synonym: mechanisch) lösen kann?

Rechner werden nie intelligent sein.

- Was bedeutet es genau, intelligent zu sein?
- Menschenaffen werden nie Sprache benutzen, Werkzeuge benutzen, Nutzung von Werkzeugen weitergeben, Emotionen zeigen, ...

Halteproblem

Eingabe: Ein Programm M und eine Eingabe x.

Frage: Hält M an der Eingabe x? Ja/Nein.

Satz: Es gibt kein Programm für das Halteproblem.

Ich lüge immer.

Es gibt kein Programm für das Halteproblem.

Nehmen wir an, es gäbe so ein Programm. Nennen wir es H. Dann

$$H(M,x) = \begin{cases} 1 & \text{falls } M \text{ an der Eingabe } x \text{ anhält,} \\ 0 & \text{falls } M \text{ an der Eingabe } x \text{ nicht anhält.} \end{cases}$$

Dann gibt es auch ein Programm Q mit:

$$Q(M) = \begin{cases} 1 & \text{falls } H(M,M) = 0. \\ \text{hält nicht an} & \text{falls } H(M,M) = 1. \end{cases}$$

$$= \begin{cases} 1 & \text{falls } M \text{ an Eingabe } M \text{ nicht anhält,} \\ \text{hält nicht an} & \text{falls } M \text{ an der Eingabe } M \text{ anhält.} \end{cases}$$

Was macht Q an der Eingabe Q?

$$Q(Q) = \begin{cases} 1 & \text{falls } Q \text{ an Eingabe } Q \text{ nicht anhält,} \\ \text{hält nicht an} & \text{falls } Q \text{ an der Eingabe } Q \text{ anhält.} \end{cases}$$

Das geht nicht. Also gibt es H nicht.

Es gibt kein Programm für das Halteproblem.

Nehmen wir an, es gäbe so ein Programm. Nennen wir es H. Dann

$$H(M,x) = \begin{cases} 1 & \text{falls } M \text{ an der Eingabe } x \text{ anhält,} \\ 0 & \text{falls } M \text{ an der Eingabe } x \text{ nicht anhält.} \end{cases}$$

Dann gibt es auch ein Programm Q mit:

$$Q(M) = \begin{cases} 1 & \text{falls } H(M,M) = 0. \\ \text{hält nicht an} & \text{falls } H(M,M) = 1. \end{cases}$$

$$= \begin{cases} 1 & \text{falls } M \text{ an Eingabe } M \text{ nicht anhält,} \\ \text{hält nicht an} & \text{falls } M \text{ an der Eingabe } M \text{ anhält.} \end{cases}$$

Was macht Q an der Eingabe Q?

$$Q(Q) = \begin{cases} 1 & \text{falls } Q \text{ an Eingabe } Q \text{ nicht anhält,} \\ \text{hält nicht an} & \text{falls } Q \text{ an der Eingabe } Q \text{ anhält.} \end{cases}$$

Das geht nicht. Also gibt es H nicht.

Es gibt kein Programm für das Halteproblem.

Nehmen wir an, es gäbe so ein Programm. Nennen wir es H. Dann

$$H(M,x) = \begin{cases} 1 & \text{falls } M \text{ an der Eingabe } x \text{ anhält,} \\ 0 & \text{falls } M \text{ an der Eingabe } x \text{ nicht anhält.} \end{cases}$$

Dann gibt es auch ein Programm Q mit:

$$Q(M) = \begin{cases} 1 & \text{falls } H(M,M) = 0. \\ \text{hält nicht an} & \text{falls } H(M,M) = 1. \end{cases}$$

$$= \begin{cases} 1 & \text{falls } M \text{ an Eingabe } M \text{ nicht anhält,} \\ \text{hält nicht an} & \text{falls } M \text{ an der Eingabe } M \text{ anhält.} \end{cases}$$

Was macht Q an der Eingabe Q?

$$Q(Q) = \begin{cases} 1 & \text{falls } Q \text{ an Eingabe } Q \text{ nicht anhält,} \\ \text{hält nicht an} & \text{falls } Q \text{ an der Eingabe } Q \text{ anhält.} \end{cases}$$

Das geht nicht. Also gibt es H nicht.

Es gibt kein Programm für das Halteproblem.

Nehmen wir an, es gäbe so ein Programm. Nennen wir es H. Dann

$$H(M,x) = \begin{cases} 1 & \text{falls } M \text{ an der Eingabe } x \text{ anhält,} \\ 0 & \text{falls } M \text{ an der Eingabe } x \text{ nicht anhält.} \end{cases}$$

Dann gibt es auch ein Programm Q mit:

$$Q(M) = \begin{cases} 1 & \text{falls } H(M,M) = 0. \\ \text{hält nicht an} & \text{falls } H(M,M) = 1. \end{cases}$$

$$= \begin{cases} 1 & \text{falls } M \text{ an Eingabe } M \text{ nicht anhält,} \\ \text{hält nicht an} & \text{falls } M \text{ an der Eingabe } M \text{ anhält.} \end{cases}$$

Was macht Q an der Eingabe Q?

$$Q(Q) = \begin{cases} 1 & \text{falls } Q \text{ an Eingabe } Q \text{ nicht anhält,} \\ \text{hält nicht an} & \text{falls } Q \text{ an der Eingabe } Q \text{ anhält.} \end{cases}$$

Das geht nicht. Also gibt es H nicht.

Polynome

Eine Funktion der Form $x \mapsto 5x^3 + 4x^2 + 3x + 5$ nennt man Polynom.

Allgemein $x \mapsto a_k x^k + a_{k-1} x^{k-1} + \ldots + a_1 x + a_0$, wobei a_k, $a_{k-1}, \ldots, a_0$ Zahlen sind. Die Zahl k heißt der Grad.

$k = 1$, lineare Funktion; $k = 2$, quadratische Funktion; $k = 3$, kubische Funktion.

Für große x ist der Wert im Wesentlichen gleich dem führenden Glied $a_k x^k$.

Polynome wachsen nicht zu schnell. Verdoppelt man x, so vergrößert sich der Wert des führenden Glieds um den Faktor 2^k.

Ganz anders bei der Exponentialfunktion, etwa $x \mapsto 2^x$. Erhöhen von x um 1, erhöht den Wert um den Faktor 2. Verdoppeln von x, quadriert den Wert.

Polynome

Eine Funktion der Form $x \mapsto 5x^3 + 4x^2 + 3x + 5$ nennt man Polynom.

Allgemein $x \mapsto a_k x^k + a_{k-1} x^{k-1} + \ldots + a_1 x + a_0$, wobei a_k, $a_{k-1}, \ldots, a_0$ Zahlen sind. Die Zahl k heißt der Grad.

$k = 1$, lineare Funktion; $k = 2$, quadratische Funktion; $k = 3$, kubische Funktion.

Für große x ist der Wert im Wesentlichen gleich dem führenden Glied $a_k x^k$.

Polynome wachsen nicht zu schnell. Verdoppelt man x, so vergrößert sich der Wert des führenden Glieds um den Faktor 2^k.

Ganz anders bei der Exponentialfunktion, etwa $x \mapsto 2^x$. Erhöhen von x um 1, erhöht den Wert um den Faktor 2. Verdoppeln von x, quadriert den Wert.

Polynome

Eine Funktion der Form $x \mapsto 5x^3 + 4x^2 + 3x + 5$ nennt man Polynom.

Allgemein $x \mapsto a_k x^k + a_{k-1} x^{k-1} + \ldots + a_1 x + a_0$, wobei a_k, $a_{k-1}, \ldots, a_0$ Zahlen sind. Die Zahl k heißt der Grad.

$k = 1$, lineare Funktion; $k = 2$, quadratische Funktion; $k = 3$, kubische Funktion.

Für große x ist der Wert im Wesentlichen gleich dem führenden Glied $a_k x^k$.

Polynome wachsen nicht zu schnell. Verdoppelt man x, so vergrößert sich der Wert des führenden Glieds um den Faktor 2^k.

Ganz anders bei der Exponentialfunktion, etwa $x \mapsto 2^x$. Erhöhen von x um 1, erhöht den Wert um den Faktor 2. Verdoppeln von x, quadriert den Wert.

Was ist ein Problem?

Suchproblem: Wie weit ist es von A nach B?

Entscheidungsproblem: Ist der Abstand ≤ 500 km?

Heute: Problem = Entscheidungsproblem, JA-/NEIN-Problem

P = Menge der Probleme, die man in Polynomzeit entscheiden kann, d.h.

$$x \mapsto \{\, \text{JA}, \text{Nein} \,\}$$

ist in Polynomzeit berechenbar. Dabei ist x eine Instanz des Problems.

P = die Klasse der effizient lösbaren Probleme

Suchproblem: Wie weit ist es von A nach B?

Entscheidungsproblem: Ist der Abstand $\leq$ 500km?

Heute: Problem = Entscheidungsproblem, JA-/NEIN-Problem

P = Menge, der in Polynomzeit lösbaren Probleme.

Ein Problem ist in P, wenn es ein Programm M gibt, das

- für jede Problemstellung x die richtige Antwort liefert, und
- dessen Laufzeit polynomiell beschränkt ist, d.h., es gibt ein Polynom p, so dass

$$\text{Laufzeit von M an Eingabe } x \leq p(|x|)$$

für alle Eingaben x; dabei ist $|x|$ die Länge von x (Anzahl der Zeichen).

P = die Klasse der effizient lösbaren Probleme

Suchproblem: Wie weit ist es von A nach B?

Entscheidungsproblem: Ist der Abstand $\leq$ 500km?

Heute: Problem = Entscheidungsproblem, JA-/NEIN-Problem

P = Menge, der in Polynomzeit lösbaren Probleme.

Ein Problem ist in P, wenn es ein Programm M gibt, das

- für jede Problemstellung x die richtige Antwort liefert, und
- dessen Laufzeit polynomiell beschränkt ist, d.h., es gibt ein Polynom p, so dass

$$\text{Laufzeit von M an Eingabe } x \leq p(|x|)$$

für alle Eingaben x; dabei ist $|x|$ die Länge von x (Anzahl der Zeichen).

P = die Klasse der effizient lösbaren Probleme

Heute: Problem = Entscheidungsproblem, JA-/NEIN-Problem

P = Menge, der in Polynomzeit lösbaren Probleme.

Ein Problem ist in P, wenn es ein Programm M gibt, das

- für jede Problemstellung x die richtige Antwort liefert, und
- dessen Laufzeit polynomiell beschränkt ist, d.h., es gibt ein Polynom p, so dass

$$\text{Laufzeit von M an Eingabe } x \leq p(|x|)$$

für alle Eingaben x; dabei ist $|x|$ die Länge von x (Anzahl der Zeichen).

Kürzer: Ein Problem ist in P, wenn die zugehörige Funktion $x \mapsto \{\text{JA,NEIN}\}$ in Polynomzeit berechenbar ist.

P = die Klasse der effizient lösbaren Probleme

Heute: Problem = Entscheidungsproblem, JA-/NEIN-Problem

P = Menge, der in Polynomzeit lösbaren Probleme.

Ein Problem ist in P, wenn es ein Programm M gibt, das

- für jede Problemstellung x die richtige Antwort liefert, und
- dessen Laufzeit polynomiell beschränkt ist, d.h., es gibt ein Polynom p, so dass

$$\text{Laufzeit von M an Eingabe } x \leq p(|x|)$$

für alle Eingaben x; dabei ist $|x|$ die Länge von x (Anzahl der Zeichen).

lineares Polynom $\rightarrow$ lineare Laufzeit, quadratisches Polynom $\rightarrow$ quadratische Laufzeit, ...

Beispiele: kürzeste Wege, Sortieren, Suchen, Editierdistanz, Lösen von linearen Gleichungen, ...

P = die Klasse der effizient lösbaren Probleme

Heute: Problem = Entscheidungsproblem, JA-/NEIN-Problem

P = Menge, der in Polynomzeit lösbaren Probleme.

Ein Problem ist in P, wenn es ein Programm M gibt, das

- für jede Problemstellung x die richtige Antwort liefert, und
- dessen Laufzeit polynomiell beschränkt ist, d.h., es gibt ein Polynom p, so dass

$$\text{Laufzeit von M an Eingabe } x \leq p(|x|)$$

für alle Eingaben x; dabei ist $|x|$ die Länge von x (Anzahl der Zeichen).

Postulat: effizient = polynomielle Laufzeit.

- Nichtpolynomiell ist sicher nicht effizient.
- Definition ist robust und hängt nicht von den Details der Rechenanlagen ab.

Beispiele für Probleme in NP

NP = Menge aller Ja-/Nein-Probleme, bei denen ein Lösungsvorschlag für Ja-Instanzen in Polynomzeit überprüft werden kann (genaue Definition folgt).

Rucksackproblem

Gegeben sind n Objekte und zwei Zahlen G und W. Das i-te Objekt hat das Gewicht g_i und den Wert w_i.

Gibt es eine Teilmenge der Objekte, die als Gesamtgewicht höchstens G und als Wert mindestens W hat?

Eingabe: $(2,5)$, $(7,4)$, $(9,3)$, $(4,6)$, $G = 6$, $W = 10$.

Lösungsvorschlag: Nimm das 1. und das 4. Objekt.

Überprüfung: $g_1 + g_3 = 2 + 4$. Das ist ≤ 6. $w_1 + w_4 = 5 + 6$. Das ist ≥ 10.

Beispiele für Probleme in NP

Problem des Handlungsreisenden

Gibt es eine Tour durch alle Orte Deutschlands mit mehr als 5 Tausend Einwohnern, die höchstens 4000 Kilometer lang ist?

Allgemein: Gegeben ist ein Graph, für jede Kante ihre Länge, und eine Zahl L.
Gibt es eine Rundreise, die alle Knoten besucht und deren Länge höchstens L ist?

Lösungsvorschlag: Eine Rundreise durch deutsche Städte.

Überprüfung:

- Werden alle Städte mit mindestens 5000 Einwohnern besucht?
- Ist die Gesamtlänge höchstens 4000 Kilometer?

Graphenfärbung

Graphenfärbung

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

Frage: Gibt es eine Färbung der Knoten mit drei Farben, so dass die Endpunkte einer jeder Kante verschiedene Farbe haben.

Lösungsvorschlag: Eine Färbung der Knoten mit drei Farben.

Überprüfung: Sieh nach, ob die Endpunkte einer jeden Kante verschiedene Farben haben.

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: Wahrheitswerte und Variablen verknüpft mit und ($\wedge$), oder ($\vee$) und Negation ($\neg$). Details auf nächster Folie.

Beispiel

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

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

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

Formeln der Aussagenlogik

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

Belegung, Wert einer Formel, erfüllbar

Eine Belegung weist jeder Variablen einen Wahrheitswert zu.

Der Wert der Formel ergibt sich nach folgenden Regeln:

x	y	$x \vee y$	$x \wedge y$	$\neg x$
F	F	F	F	W
F	W	W	F	W
W	F	W	F	F
W	W	W	W	F

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

Wichtige Formeln

Gib eine Formel in den Variablen x, y und z an, die genau dann wahr ist, wenn mindestens eine der Variablen wahr ist.

Gib eine Formel in den Variablen x, y und z an, die genau dann wahr ist, wenn mindestens zwei der Variablen wahr sind.

Gib eine Formel in den Variablen x, y und z an, die genau dann wahr ist, wenn genau eine der Variablen wahr ist.

Zeigen Sie: SAT $\in$ NP.

Lösungsvorschlag:

Überprüfung:

Algorithmen für SAT, Rucksack, Handlungsreisender, ...

SAT: Probiere alle Belegungen durch und finde heraus, ob es eine erfüllende gibt.

Bei n Variablen gibt es 2^n mögliche Belegungen.

Rucksack: Probiere alle Teilmengen der Objekte durch und ....

Bei n Variablen gibt es 2^n mögliche Teilmengen.

Handlungsreisender: Probiere alle möglichen Touren aus und ....

Bei n Städten gibt es $n!$ mögliche Rundreisen.

Alle bekannten Algorithmen haben exponentielle Laufzeit im schlechtesten Fall.

Genaue Definition von NP

Ein JA-/NEIN-Problem ist in NP, wenn es eine in Polynomzeit berechenbare Funktion

$$R : (x, y) \mapsto \{\,\text{JA}, \text{NEIN}\,\}$$

und ein Polynom p gibt, so dass für jede Instanz x des Problems gilt:

- x ist JA-Instanz: dann gibt es ein y, so dass $R(x, y) = \text{JA}$ und $|y| \leq p(|x|)$.
- x is NEIN-Instanz: für alle y gilt $R(x, y) = \text{NEIN}$.

- R muss effizient berechenbar sein.
- Im Fall einer JA-Instanz x heißt das y mit $R(x, y) = \text{JA}$ Zeuge (Witness) für x.
- JA-Instanzen haben einen Zeugen, dessen Länge polynomiell ist relativ zur eigenen Länge.
- NEIN-Instanzen haben keinen Zeugen.

Genaue Definition von NP

Ein JA-/NEIN-Problem ist in NP, wenn es eine in Polynomzeit berechenbare Funktion

$$R : (x, y) \mapsto \{\,\text{JA}, \text{NEIN}\,\}$$

und ein Polynom p gibt, so dass für jede Instanz x des Problems gilt:

- x ist JA-Instanz: dann gibt es ein y, so dass $R(x, y) = \text{JA}$ und $|y| \leq p(|x|)$.
- x is NEIN-Instanz: für alle y gilt $R(x, y) = \text{NEIN}$.

- R muss effizient berechenbar sein.
- Im Fall einer JA-Instanz x heißt das y mit $R(x, y) = \text{JA}$ Zeuge (Witness) für x.
- JA-Instanzen haben einen Zeugen, dessen Länge polynomiell ist relativ zur eigenen Länge.
- NEIN-Instanzen haben keinen Zeugen.

Genaue Definition von NP

Ein JA-/NEIN-Problem ist in NP, wenn es eine in Polynomzeit berechenbare Funktion

$$R : (x, y) \mapsto \{\,\text{JA}, \text{NEIN}\,\}$$

und ein Polynom p gibt, so dass für jede Instanz x des Problems gilt:

- x ist JA-Instanz: dann gibt es ein y, so dass $R(x, y) = \text{JA}$ und $|y| \leq p(|x|)$.
- x is NEIN-Instanz: für alle y gilt $R(x, y) = \text{NEIN}$.

Beispiel: Erfüllbarkeitsproblem

- $x =$ eine Formel der Aussagenlogik.
- $y =$ eine Belegung der Variablen mit Wahrheitswerten.
- $R(x, y) = \begin{cases} \text{JA} & \text{falls die Belegung } y \text{ erfüllt } x. \\ \text{NEIN} & \text{falls die Belegung } y \text{ erfüllt } x \text{ nicht.} \end{cases}$.

Genaue Definition von NP

Ein JA-/NEIN-Problem ist in NP, wenn es eine in Polynomzeit berechenbare Funktion

$$R : (x, y) \mapsto \{\,\text{JA, NEIN}\,\}$$

und ein Polynom p gibt, so dass für jede Instanz x des Problems gilt:

- x ist JA-Instanz: dann gibt es ein y, so dass $R(x, y) = \text{JA}$ und $|y| \leq p(|x|)$.
- x is NEIN-Instanz: für alle y gilt $R(x, y) = \text{NEIN}$.

P ist eine Teilmenge von NP

Für ein Problem in P ist die zugehörige Funktion $f : x \mapsto \{\,\text{JA, NEIN}\,\}$ in Polynomzeit berechenbar.

Definiere $R(x, y) = f(x)$.

Der Satz von Cook/Levin

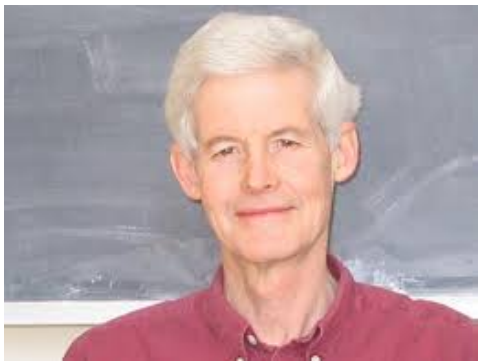
NP = Menge aller Ja-/Nein-Probleme, bei denen ein Lösungsvorschlag für Ja-Instanzen in Polynomzeit überprüft werden kann.

Satz (Stephen Cook und Leonid Levin, 1971)

SAT $\in$ P genau wenn NP = P.

Ich zeige ein Beispiel: SAT $\in$ P impliziert Graphenfärbung $\in$ P.

Theorie schafft Einsicht.



Cook: Turing Award

Levin: Knuth Prize.

NP-Vollständigkeit (Satz von Karp)

Definition

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

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

Satz (Karp, 1972)

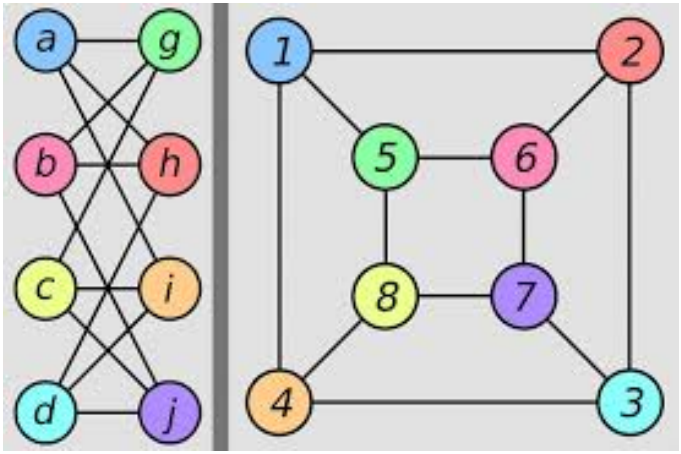
Das Graphenfärbungsproblem, das Hamiltonsche Kreisproblem, Knapsack und 20 andere Probleme sind NP-vollständig.

Die Liste ist inzwischen auf mehrere Tausend angewachsen: Theorie schafft Ordnung.

Richard Karp, Turing-Award in 1985.

Nicht NP-vollständige Probleme in NP

Graphenisomorphie: Zwei Graphen heißen isomorph, wenn man die Knoten des einen so umbenennen kann, dass man den zweiten Graphen erhält.



aus Spektrum der Wissenschaft

Primzahltest war lange ein solches Problem. Seit 2002 kennt man einen effizienten Algorithmus.

Graphenfärbung ist nicht schwerer als SAT

Graphenfärbung (mit drei Farben)

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

Frage: Gibt es eine Färbung der Knoten von G mit den Farben rot, blau und grün, so dass die Endpunkte einer jeden Kante verschieden gefärbt sind?

Graphenfärbung ist nicht schwerer als SAT

Graphenfärbung (mit drei Farben)

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

Frage: Gibt es eine Färbung der Knoten von G mit den Farben rot, blau und grün, so dass die Endpunkte einer jeden Kante verschieden gefärbt sind?

Satz: Falls SAT effizient lösbar ist, dann ist auch Graphenfärbung effizient lösbar (Färbung $\leq$ SAT)

Gegeben ein Graph G, konstruiere eine Formel F mit:

- G ist dreifärbbar genau wenn F erfüllbar ist.
- Die Konstruktion von F aus G ist in Polynomzeit durchführbar.

Reduktion von Färbung auf SAT.

Graphenfärbung ist nicht schwerer als SAT

Satz: Falls SAT effizient lösbar ist, dann ist auch Graphenfärbung effizient lösbar (Färbung $\leq$ SAT)

Gegeben ein Graph G, konstruiere eine Formel F mit:

- G ist dreifärbbar genau wenn F erfüllbar ist.
- Die Konstruktion von F aus G ist in Polynomzeit durchführbar.

Reduktion von Färbung auf SAT.

Färbungsalgorithmus: An Eingabe G tue:

- Konstruiere F.
- Entscheide Erfüllbarkeit von F mit Hilfe des SAT-Algorithmus.

Algorithmus ist korrekt und effizient.

Reduktion: Färbung $\leq$ SAT

Für jeden Knoten $u \in V$ führen wir drei Variablen ein, nämlich u_R, u_B, und u_G, d.h., eine Variable für jede Farbe.

Intendierte Bedeutung: $u_c = Wahr$ bedeutet u hat die Farbe c.

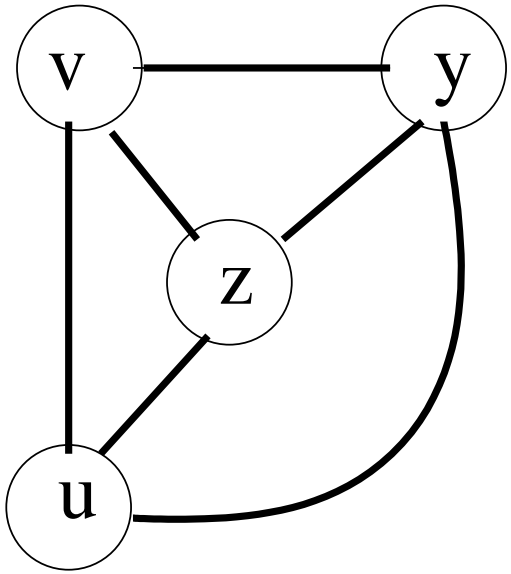
Was müssen wir ausdrücken?

- Jeder Knoten hat eine eindeutige Farbe: $\bigwedge_{u \in V} GE(u_R, u_B, u_G)$.
  Dabei ist $GE(x, y, z) = (x \vee y \vee z) \wedge \neg (xy \vee xz \vee yz)$ GenauEine
- Für alle Kanten $\{ u, v \}$: u und v haben verschiedene Farben:
  u und v haben die gleiche Farbe: $u_R v_R \vee u_B v_B \vee u_G v_G$.
  u und v haben verschiedene Farben: $VF(u, v) = \neg(u_R v_R \vee u_B v_B \vee u_G v_G)$.
  Also, $\bigwedge_{\{ u,v \} \in E} \neg(u_R v_R \vee u_B v_B \vee u_G v_G)$.

Insgesamt

$$\bigwedge_{u \in V} GE(u_R, u_B, u_G) \wedge \bigwedge_{\{ u,v \} \in E} VF(u, v)$$

Beispiel



- Ist der Graph dreifärbbar? JA oder NEIN?
- Gib die Formel für die Existenz einer Dreifärbung an.
- Streiche eine beliebige Kante und ändere die Formel entsprechend ab.
- Finde eine erfüllende Belegung und leite daraus die Dreifärbung ab.

Die P = NP Frage (Geschichte)

- P = Menge aller Probleme, die in Polynomzeit lösbar sind.
- NP = Menge aller Ja-/Nein-Probleme, bei denen ein Lösungsvorschlag für Ja-Instanzen in Polynomzeit überprüft werden kann.
- 1970: Man hatte effiziente Algorithmen für kürzeste Wege, Paarungsprobleme, Flussprobleme, aber es gab auch viele Probleme, die man nicht effizient lösen konnte.
- 1971, 1972: Cook, Levin und Karp haben Ordnung in dieses Chaos gebracht. Viele dieser Probleme sind zu SAT äquivalent. SAT $\in$ P impliziert P = NP.
- Clay Foundation gibt je 1 Mio. $ für Lösung von 6 großen mathematischen Fragen. Eine davon ist: “Ist P = NP?”.
- Frage hat grundlegende philosophische/mathematische Bedeutung (Ist Beweisen schwerer als Prüfen?).

  Gleichheit hätte enorme algorithmische Konsequenzen.

Was 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.

Was 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?

  Formales System = Korrektheit eines Beweises entscheidbar in Polynomzeit

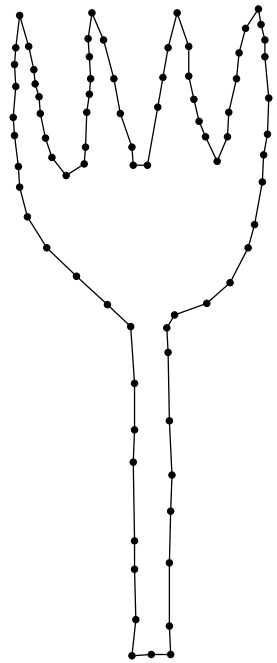
  Dieses Problem ist in NP. Falls P = NP, dann ist dieses Problem in P.
- Philosophen müssten neu über den Begriff Kreativität nachdenken.
- Alle paar Monate wird ein (falscher) Beweis angekündigt.

Wie geht man mit NP-Vollständigkeit um?

Nur weil ein Problem schwer ist, verschwindet es nicht.

NP-Vollständigkeit bedeutet: Man kennt keinen Algorithmus, der jede Problemstellung in Polynomzeit löst. Es kann durchaus Algorithmen geben, die viele (interessante) Instanzen in Polynomzeit lösen. Folgende Ansätze gibt es:

- Heuristiken, z.B. für TSP: gehe immer zur nächstgelegenen noch nicht besuchten Stadt.
- Exakte Algorithmen für kleine Instanzen (Applegate et al.: an optimal TSP tour through 85900 cities.)
- Spezialfälle, z.B., für Punkte, die von einer gutartigen Kurve kommen, ist TSP in Polynomzeit (Althaus/Mehlhorn, 2002).
- Approximationsalgorithmen, z.B., 3/2-Approximation der kürzesten Tour in Polynomzeit (Christofides – Serdyukov, 1978)



Zusammenfassung

- Es gibt Probleme, die nicht mechanisierbar sind.
- P = Menge der Probleme, bei denen eine Lösung in Polynomzeit gefunden werden kann.
- NP = Menge aller Ja-/Nein-Probleme, bei denen ein Lösungsvorschlag für Ja-Instanzen in Polynomzeit überprüft werden kann.
- P = NP genau wenn es einen polynomiellen Algorithmus für das Erfüllbarkeitsproblem der Aussagenlogik (SATisfiability Problem) gibt.
- P = NP, eines der großen offenen Probleme der Informatik/Mathematik (Clay Prize)
- Falls P $\neq$ NP, dann ...
- Falls P $=$ NP, dann ...