

NP-Vollständigkeit

Hannes Eder

18.12.2001

aktualisiert 14.10.2007

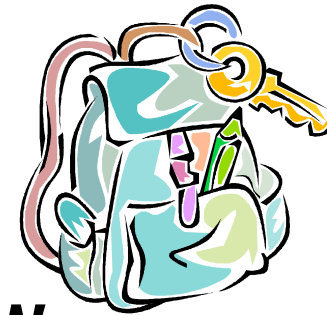
Reiseroute

- * Einige konkrete Beispiele (RUCK, 3-COL, TSP, ...)
- * Begriffsdefinitionen
 - Entscheidungsprobleme / Berechnungsprobleme / Optimierungsproblem
 - Problem / Probleminstanz
 - Deterministisch vs. Nichtdeterministisch
- * Was heißt „effizient“
- * Die Klasse NP
- * NP-Vollständigkeit
- * Lösungsansätze
 - Randomisierte lokale Suche
 - Approximation
 - Identifikation leicht lösbarer Subklassen

Das Rucksackproblem

Allgemeine Problemformulierung:

Angabe: Gegeben eine Liste der Form
<Gegenstand, Gewicht, Preis>, sowie
Maximales Gewicht **G**, gewünschter Wert **W**



Gegenstand	Gewicht	Wert
Luster	5500	14300,-
Schatulle	3200	8000,-
Schwert	1500	8500,-
Bild	3400	6800,-
...



3 mögliche Problemstellungen beim Rucksackproblem

Entscheidungsproblem: Gibt es eine Menge $S \subseteq$ Gegenstände, so dass Gesamtgewicht $\leq G$ und Gesamtwert $\geq W$?

Berechnungsproblem: Berechne eine Lösung S , so dass Gesamtgewicht $\leq G$ und Gesamtwert $\geq W$.

Optimierungsproblem: Berechne eine Lösung S , so dass Gesamtgewicht $\leq G$ und Gesamtwert maximal.

Ähnliche Problemstellungen: Frachtraumplanung, Lagerplanung, Verschnittoptimierung, ...

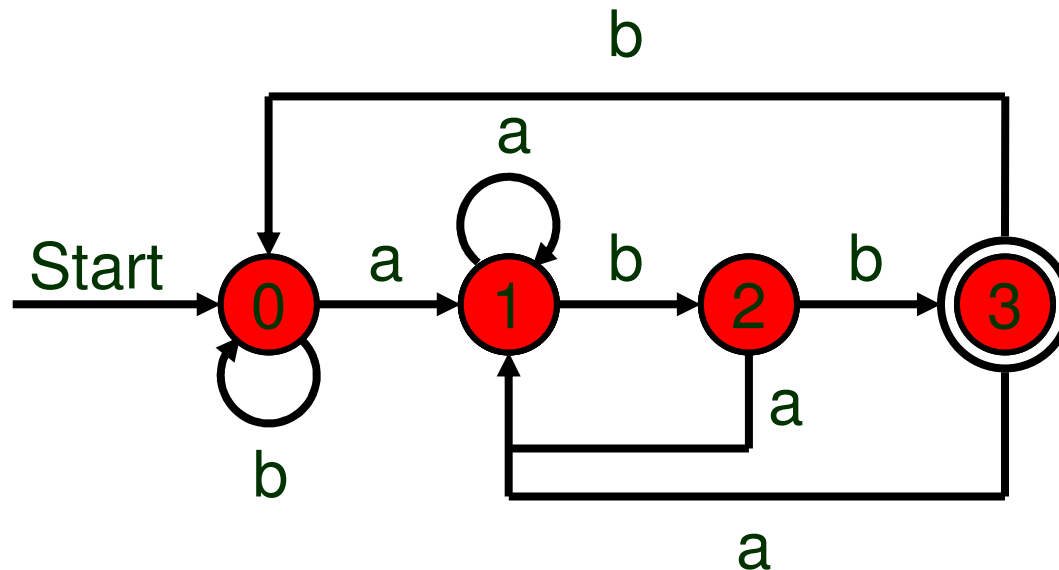
Problem / Probleminstanz

Problem: Abstrakte Beschreibung des Problems ohne konkrete Daten

Instanz: Konkrete Angaben, zu einem Problem existieren i.A. unendlich viele Instanzen.

Größe einer Instanz (n): Anzahl der verwendeten Zeichen, Knoten, Wörter die zu sortieren sind usw.

Deterministischer Automat für $(a|b)^*abb$



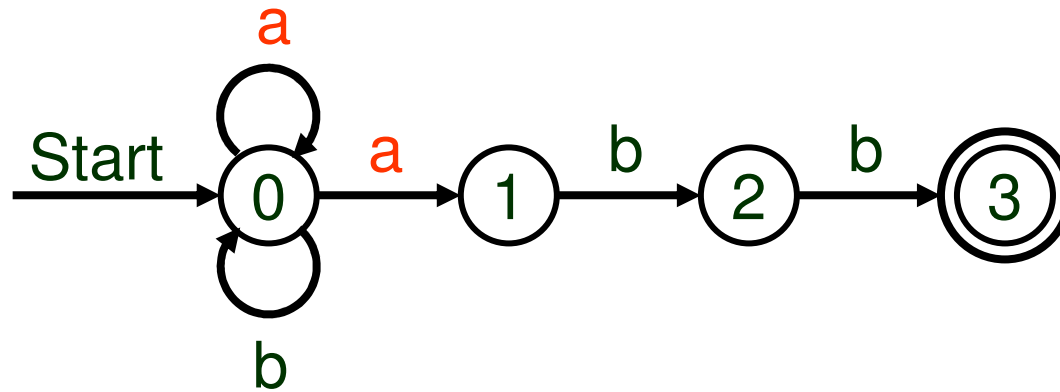
Input: aababb

Der bekannteste deterministische Automat



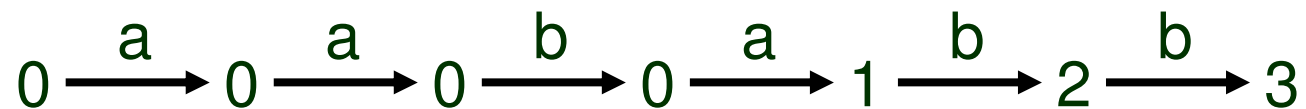
- * Viele Zustände
- * Eingabe-Alphabet ist das Instructionset der CPU
- * „kein Zufall möglich“, nur Pseudozufallszahlen

Nichtdeterministischer Automat für $(a|b)^*abb$



Input: **aababb**

Optimal geratene Zustansfolge:



Anmerkung: Bei Patternmatching wird ein NEA mit einer Zustandsmenge simuliert

Berechnungsmodelle

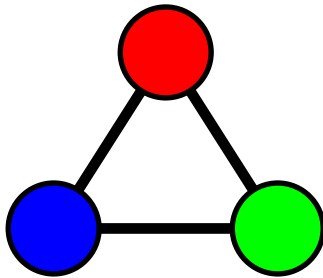
Simulierte Maschine	Simulierende Maschine		
	1TM	kTM	RAM
1-Band Turing Machine (1TM)	-	$O(T(n))$	$O(T(n)\log(T(n)))$
k-Band Turing Machine (kTM)	$O(T^2(n))$	-	$O(T(n)\log(T(n)))$
Random Access Machine (RAM)	$O(T^3(n))$	$O(T^2(n))$	-

- * Laufzeit hängt vom Berechnungsmodell ab
- * TM kennt keine Arrays
- * Häufiges Berechnungsmodell: Pascal Variante
- * These von Church:
 - Berechnungsmodelle unterscheiden sich nur um einen polynomiellen Faktor
 - Jede im intuitiven Sinne berechenbare Funktion ist auch Turing-berechenbar

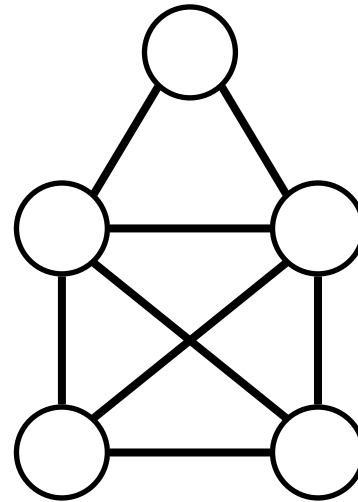
3-COL Problem

- ★ **Eingabe:** Graph $G = (\{v_1, \dots, v_n\}, E)$
- ★ **Frage:** Kann man die Knoten dieses Graphen mit 3 Farben so färben, dass 2 benachbarte Knoten nicht dieselbe Farbe haben.
- ★ **Anwendung:** Mobilfunkbetreiber mit 3 Frequenzen
- ★ **Mögliche Fragestellungen:**
 - Ist G 3-färbbar?
 - Berechne eine korrekte 3-Färbung?

3-COL Entscheidungsproblem

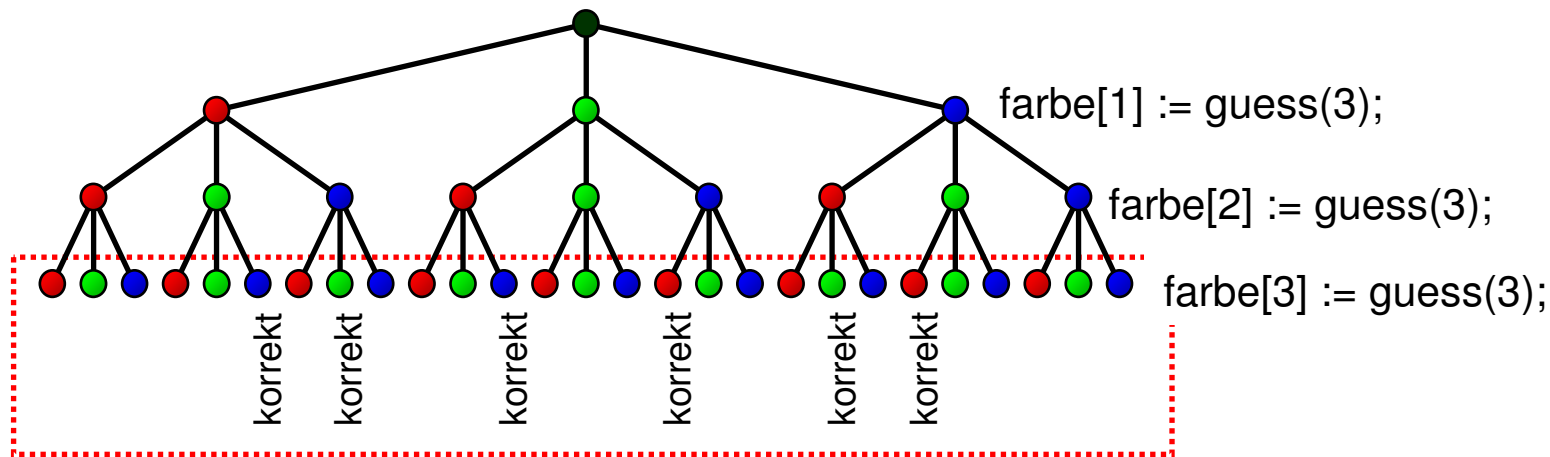
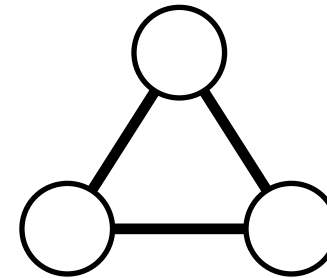


3-Färbung möglich



keine 3-Färbung
möglich

Berechnung einer 3-Färbung



Überprüfungsphase

Berechnung einer 3-Färbung

- * 3^n mögliche Kombinationen
- * Kann in $O(n^2)$ überprüft werden:

```
begin
  for i=1 to n-1 do
    for j=i+1 to n do
      if (( $v_i, v_n$ ) in E) and
          Farbe[i]=Farbe[n])
        then return "nein";
    return "ja";
end.
```

3-COL, Vergleich DM / NDM

DM

- * Solange kein besserer Algorithmus gefunden wurde, müssen alle 3^n Möglichkeiten überprüft werden
- * Da 3-COL aber NP-Vollständig ist, ist es unwahrscheinlich, dass einer gefunden wird

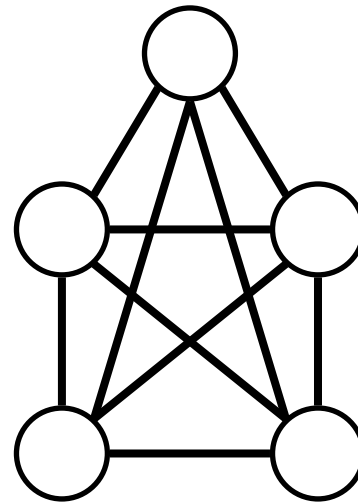
NDM

```
/* Ratephase */  
for i=1 to n do  
    Farbe[i] = guess(3);  
  
/* Überprüfungsphase */
```

Travelling Salesperson Problem (TSP)

Angabe: Ein
Straßennetz G mit
Distanzen, Zahl M
(gewichteter Graph)

Frage: Gibt es eine
„Tour“ durch alle
Städte mit der
Gesamtlänge $\leq M$



$4! = 24$ mögliche Touren

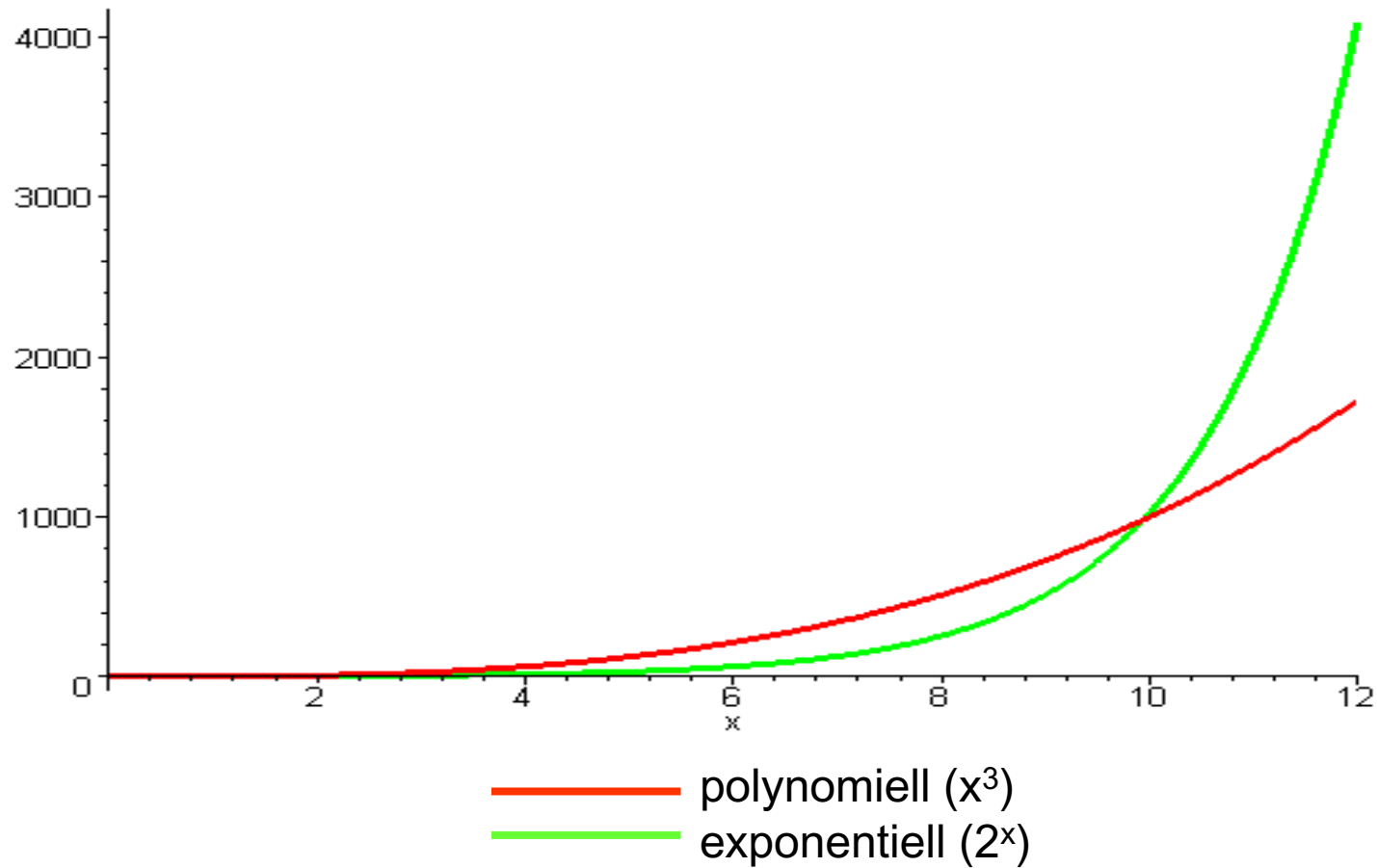
Was heißt überhaupt „effizient“?

worst case auf einem 1-MIPS Rechner

Zeitkomplexitätsfunktion	Größe (n)					
	10	20	30	40	50	60
n	100s	200s	300s	400s	500s	600s
n^2	1000	4000	9000s	1.6ms	2.5ms	3.6ms
n^3	1ms	8ms	27ms	64ms	125ms	216ms
n^5	0.1s	3.2s	24.3s	1.7min	5.2min	13min
2^n	1ms	1s	17.9min	12.7d	35.7a	366jhd
3^n	59ms	58min	6.5a	3855jhd	2×10^{10} a	1.3×10^{15} a

- * effizient = polynomiell (**Klasse P**)
- * in der Praxis nicht höher als n^8

Exponentiell und polynomiell im Vergleich



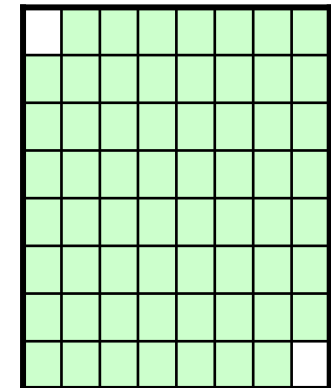
Problemlomplexität

* unentscheidbar

- Gödel'sches Unvollständigkeitstheorem
- Halteproblem

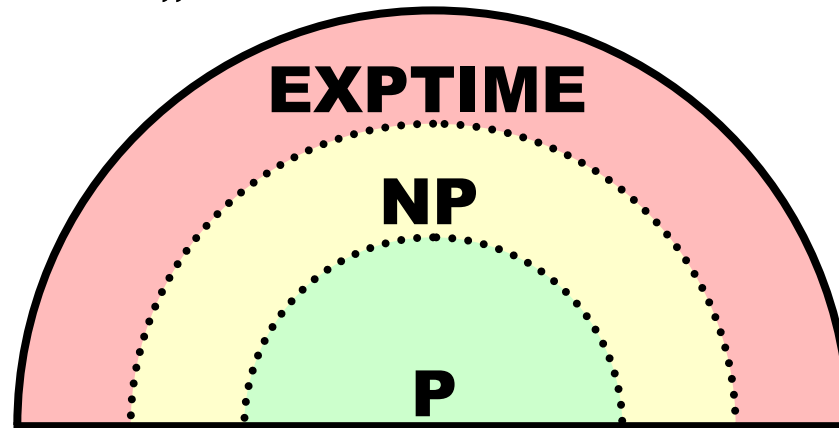
* entscheidbar

- Beweisbar exponentiell (Domino Problem, ...)
- NP-Vollständig (Rucksackproblem (RUCK), 3-Färbung (3-COL), Travelling Salesperson (TSP), Erfüllbarkeit (SAT))
- Beweisbar polynomiell (Sortieren, usw.)



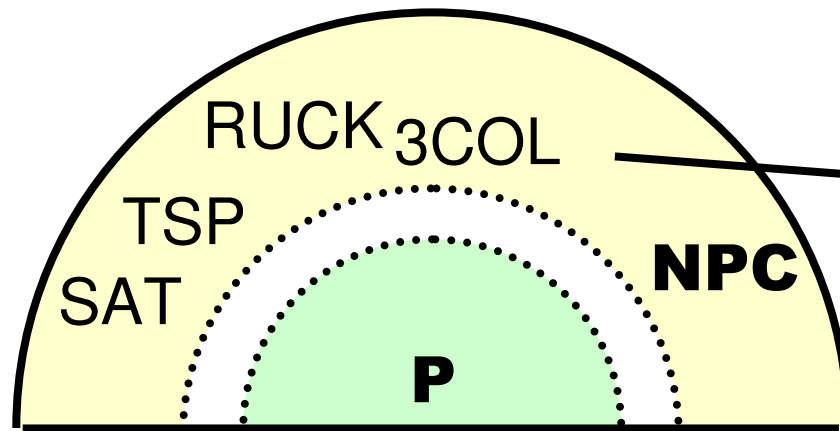
NP (Nichtdeterministisch Polynomiell)

- ★ Paradigma: „Guess and Check“



- ★ Kein Problem in NP konnte bisher als nicht polynomiell bewiesen werden
- ★ Für viele Probleme in NP kennt man aber keinen polynomiellen Algorithmus

NP-Vollständige Probleme (NP-Complete)



Die Klasse NP im Detail

- ★ NPC: Die **schwersten** Probleme in NP
 - Alle „polynomiell ineinander überführbar“
 - Eines polynomiell \Rightarrow Alle polynomiell \Leftrightarrow **NP=P**

Polynomiale Überführung = Reduktion

Eingabe: Instanz x von Problem A ;

begin

$y = T(x);$

/ y ist Instanz von Problem B */*

$z = U_B(y);$

return $z;$

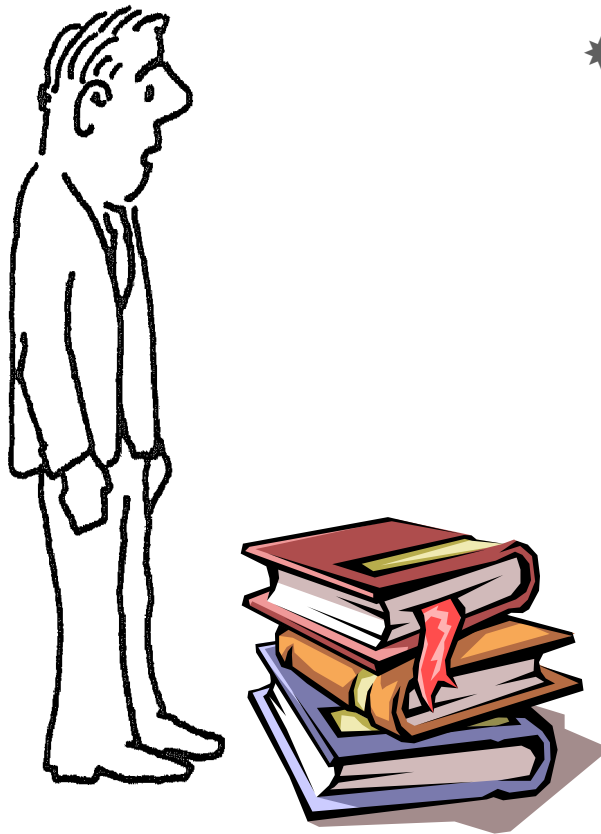
end.

- * Kann transformiert werden \Rightarrow „Nicht schwerer als“

Das 1. NP-Vollständige Problem

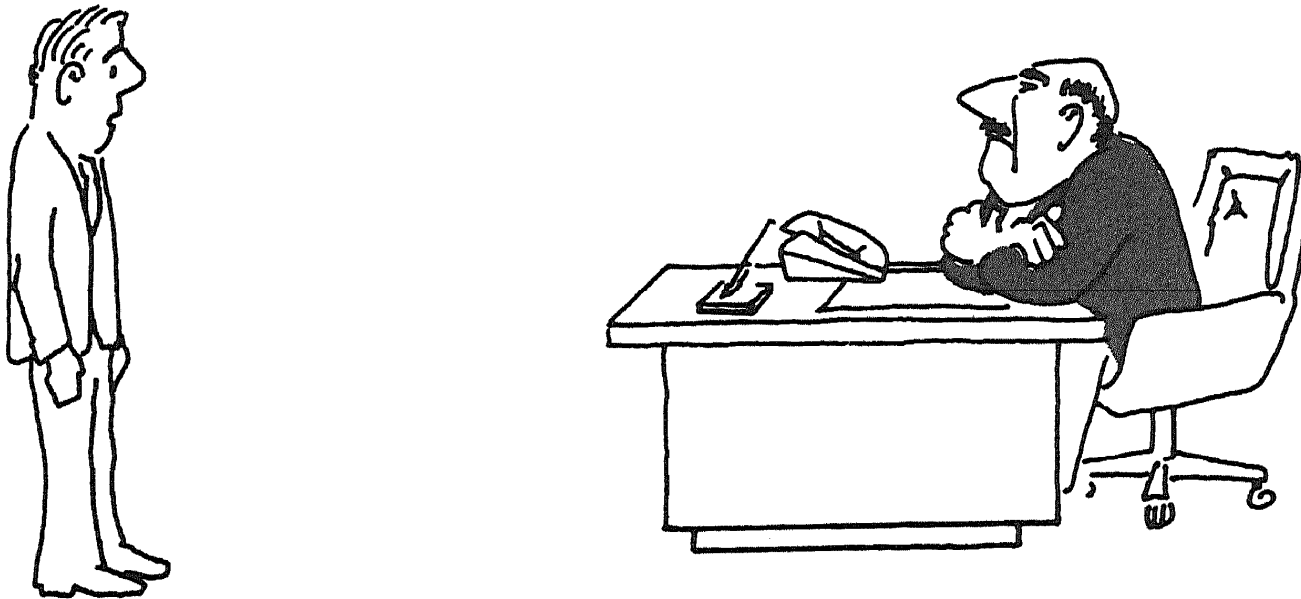
- ★ Stephen Cook und Leonid Levin präsentieren unabhängig voneinander das 1. NPV-Problem
- ★ 1970: Cook veröffentlicht das Erfüllbarkeitsproblem (SAT) und zeigt das es NPC ist.
- ★ SAT
 - $A=(X \vee \neg Y) \wedge (\neg X \vee Y)$... Ist erfüllbar($X=1, Y=1$)
 - $B=(X \vee Y) \wedge (X \vee \neg Y) \wedge (\neg X)$... Ist nicht erfüllbar

Harald und sein Chef, 1. Woche



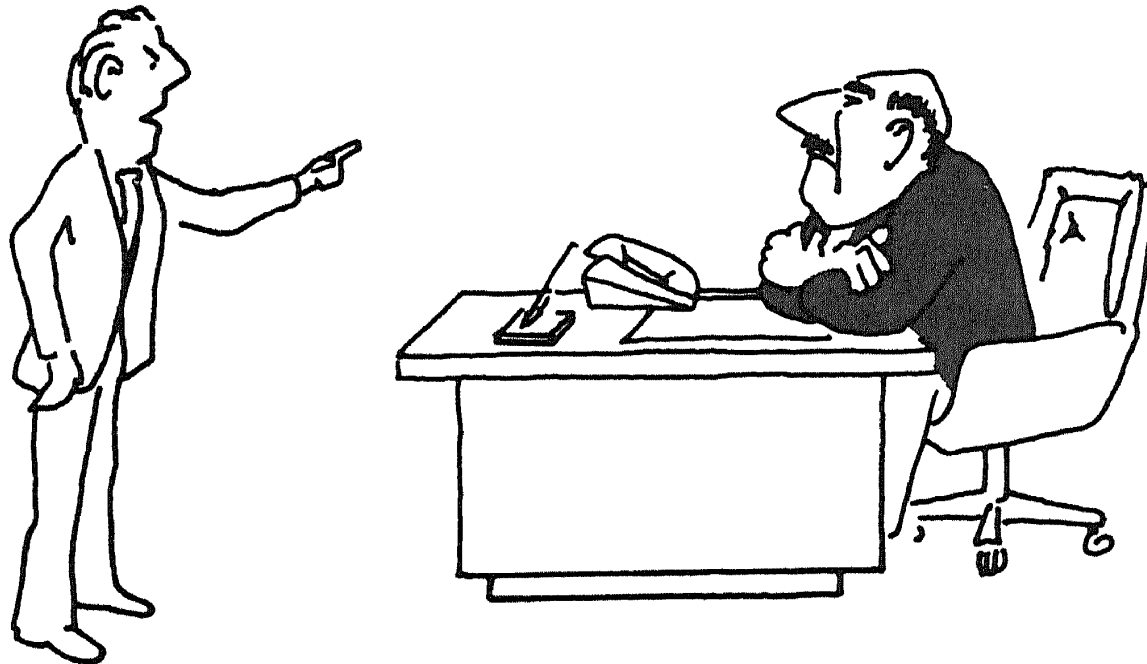
- * Harald ist Chef-Algorithmen-Designer in einem großen Softwarehaus. Er hat von seinem Chef den Auftrag bekommen, einen **effizienten** Algorithmus für ein Problem zu entwickeln.

Harald und sein Chef, 2. Woche



Harald: “Ich kann keinen effizienten Algorithmus finden, vielleicht bin ich einfach zu dumm ...”

Harald und sein Chef, 3. Woche



Harald: “Ich kann keinen effizienten Algorithmus finden, weil es keinen gibt ...”

Harald und sein Chef, 4. Woche



Harald: “Ich kann keinen effizienten Algorithmus finden, aber all diese berühmten Wissenschaftler auch nicht.”

Lösungsansätze

- ★ NP-Vollständige Probleme treten in der Praxis häufig auf.
- ★ Sie müssen mit akzeptablen Methoden gelöst werden.
 - Randomisierte lokale Suche
 - Approximation
 - Identifikation leicht lösbarer Subklassen

Randomisierte lokale Suche

1. Erzeuge zufälligen Lösungskandidaten
2. Führe solange wie mögliche lokale Verbesserungen durch
3. Wenn Lösung gefunden: Ausgeben und Programm beenden
4. Wenn Zeitlimit erreicht, Abbruch: „Keine Lösung gefunden“
5. Gehe zu Schritt 1

Randomisierte lokale Suche (Fortsetzung)

★ Vorteile:

- Funktioniert erstaunlich gut für viele praktische Probleme
- Führt mit hoher Wahrscheinlichkeit zum Erfolg

★ Nachteile:

- Funktioniert schlecht, falls keine Lösung existiert
- Liefert keinen „Unlösbarkeitsbeweis“
- Funktioniert schlecht bei Instanzen mit wenigen Lösungen
- Funktioniert nur mit Bewertungsfunktion (z.B. nicht geeignet zum Knacken von Codes)

Approximation von Berechnungsproblemen

- * **Ziel:** Finde annähernd optimale Lösungen
- * z.B. Routing der Leitungen von Integrierten Schaltungen
- * Rucksackproblem ist beliebig approximierbar
- * **Vorteile:**
 - Approximationen sind in der Praxis ausreichend
 - Auch auf manche Probleme, die beweisbar exponentiell sind, anwendbar
- * **Nachteile:**
 - Wenige praktische Probleme sind approximierbar
 - Nicht auf Entscheidungsprobleme anwendbar

Identifikation leicht lösbarer Subklassen

- ★ Hohe Komplexität ist oft nur „worst case“
- ★ z.B. für einen azyklischen Graphen (Baum) ist das 3COL Problem trivial lösbar.

P=NP?

- ★ Eine der wichtigsten Fragen der theoretischen Informatik
- ★ Man nimmt allgemein an, dass $P \neq NP$
- ★ Weder bewiesen noch widerlegt
- ★ Eines der 7 **Millenium Prize Problems** des Clay Mathematic Institute. Mit **1 Mio. US \$** dotiert

Das war's...



... Frohe Weihnachten!