Algorithmische Graphentheorie

WS 2008/2009

Vorlesung: Dr. Felix Brandt, Dr. Jan Johannsen

Übung: Markus Brill, Felix Fischer

Institut für Informatik
LMU München

Organisatorisches

- Vorlesung
  - Donnerstag, 14.15-15.45 Uhr, Raum 1.39
- Übung (ab 27.10.)
  - Montag, 16.15-17.45, Raum 0.43
    - Übungsblätter werden Donnerstags in der Vorlesung ausgegeben
    - *Tutoraufgaben* werden in der nächsten Übung vorgerechnet
    - *Hausaufgaben* müssen bis zum Montag der darauf folgenden Woche abgegeben werden
- Kriterium zur Erlangung eines Scheins
  - mind. 50% der Hausaufgaben richtig
  - ggf. mündliche Prüfung

Literatur

- Inhalt der Vorlesung
  - Graphen sind kombinatorische Strukturen, die bei der Modellierung zahlreicher Probleme (z.B. in der Routenplanung, im Mobilfunk oder in der künstlichen Intelligenz) hilfreich sind.
  - Algorithmische Aspekte grundlegender graphentheoretischer Konzepte wie Wege, Kreise, Färbungen, Überdeckungen oder Matchings
  - Verbindungen zu Effiziente Algorithmen und Komplexitätstheorie
- Literatur
  - Cormen, Leiserson, Rivest & Stein: Introduction to Algorithms (MIT Press)
  - Diestel: Graphentheorie (Springer Verlag), elektronische Ausgabe
  - Krumke & Noltemeier: Graphentheoretische Konzepte und Algorithmen (Teubner Verlag)

Vorläufige Themenübersicht

- Grundbegriffe
  - Graphen, Speicherung von Graphen, Algorithmen, NP-Vollständigkeit, Approximationsalgorithmen
- Kürzeste Wege
- Eulerkreise, Hamiltonkreise
- Cliquen, unabhängige Mengen
- Färbungen, Überdeckungen
- Planare Graphen
- Matchings
- Steinerbäume
- Baumweite
- Graphisomorphie

Grundbegriffe

- Def.: Ein ungerichteter Graph $G = (V, E)$ ist ein Tupel bestehend aus einer Knotenmenge V und einer Kantenmenge $E \subseteq \binom{V}{2}$.
  - Beispiele: vollständige Graphen K_n, Pfade P_n, Kreise C_n
  - Nachbarschaft: $N(v) = \{u \in V \mid \{u, v\} \in E\}$
  - Knotengrad: $\deg(v) = |N(v)|$ $\quad (\Delta(G) = \max_{v \in V}(\deg(v)))$
  - Ein Graph G=(V,E) heißt k-regulär wenn deg(v)=k für alle v∈V.
  - Weitere wichtige Teilklassen: bipartite Graphen, planare Graphen

- Erweiterungen
  - Graphen mit Schleifen dürfen auch Kanten der Form (v,v) enthalten.
  - Bei Graphen mit Mehrfachkanten ist E eine Multimenge.
  - Gewichtete Graphen besitzt zusätzlich eine Gewichtsfunktion g: E→R.

Grundbegriffe (2)

- Adjazenz und Inzidenz
  - Zwei Knoten $u,v \in V$ heissen adjazent, wenn $\{u,v\} \in E$.
  - Ein Knoten $v \in V$ und eine Kante $e \in E$ heissen inzident, wenn $v \in e$.
  - Zwei Kanten $e,f \in E$ heissen inzident, wenn $e \cap f \neq \emptyset$.
- Isomorphie
  - Zwei Graphen $G_1=(V_1,E_1)$, $G_2=(V_2,E_2)$ heissen isomorph, wenn es eine bijektive Abbildung $f:V_1 \rightarrow V_2$ gibt, so dass für alle $u,v \in V_1$ gilt: $\{u,v\} \in E_1 \Leftrightarrow \{f(u),f(v)\} \in E_2$.
- Satz: In jedem Graphen ist die Anzahl der Knoten mit ungeradem Grad gerade.
  - "Auf einem Empfang geben immer gerade viele Gäste ungerade vielen Gästen die Hand."
  - Beweis: Tafel

Grundbegriffe (3)

- Teilgraphen
  - G'=(V',E') ist ein Teilgraph von G=(V,E), wenn V'⊆V und $E' \subseteq E \cap \binom{V'}{2}$.
  - G'=(V',E') ist ein induzierter Teilgraph von G=(V,E), kurz G'=G[V'], wenn V'⊆V und $E' = E \cap \binom{V'}{2}$.
- Komplementgraph
  - Der Komplementgraph des Graphen G=(V,E) ist $\overline{G}$=(V,E') mit $E' = \binom{V}{2} \setminus E$.
- Kantengraph (*line graph*)
  - Der Kantengraph des Graphen G=(V,E) ist L(G)=(V',E') mit V'=E und E'={ {u,v} | u,v∈V' und u∩v≠∅}.

Gerichtete Graphen

- Def.: Ein gerichteter Graph G=(V,A) ist ein Tupel bestehend aus einer Knotenmenge V und einer Kantenmenge A⊆VxV.
  - Im Gegensatz zu ungerichteten Graphen sind Kanten geordnete Paare.
  - Wichtige Teilklasse: Turniergraphen (vollständige gerichtete Graphen ohne Doppelkanten und Schleifen)
- Relationen
  - Für jeden gerichteten Graphen G=(V,A) ist A eine binäre Relation auf V.
  - Für jeden ungerichteten Graphen G=(V,E) beschreibt E eine irreflexive, symmetrische binäre Relation auf V.

Anwendungen

- Routenplanung
  - gerichteter Graph
  - Deutschland: ca. 5 Mio. Kreuzungen und 6 Mio. Strassen, ca. 10^100 Wege von München nach Berlin
  - kürzeste Wege
- Mobilfunk
  - beschränkte Menge von Kanälen/Frequenzen
  - ungerichteter Graph, Knoten: Antennen, Kanten: gleiche Frequenz sorgt für Interferenz
  - Färbbarkeit

Anwendungen (2)

- Künstliche Intelligenz
  - 3x3 Schiebepuzzle
  - ungerichteter Graph, Knoten: Zustände, Kanten: mögliche Zustandsübergange
  - kürzeste Wege mit Abschätzung des Restaufwands
- Wahlverfahren
  - gerichteter Graph, Knoten: Kandidaten, Kanten: paarweise Mehrheitsrelation
  - verschiedene Wahlverfahren, die nur paarweise Vergleiche in Betracht ziehen

Asymptotische Komplexität

- Berechnungsmodell der Vorlesung
  - Unit-Cost RAM
- $O(g) = \{f \mid \exists c, n_0 > 0 \text{ so dass } \forall n \geq n_0 \text{ gilt } f(n) \leq c \cdot g(n)\}$
  - f wächst höchstens so schnell wie g
- $\Omega(g) = \{f \mid \exists c, n_0 > 0 \text{ so dass } \forall n \geq n_0 \text{ gilt } f(n) \geq c \cdot g(n)\}$
  - f wächst mindestens so schnell wie g
- $\Theta(g) = O(g) \cap \Omega(g)$
  - f wächst genau so schnell wie g

Speicherung von Graphen

- $V=\{v_1, v_2, ..., v_n\}$, $E=\{e_1, e_2, ..., e_m\}$
- Adjazenzmatrix $A=(a_{ij})_{i,j\in[n]}$
  - $a_{ij} = |\{v_i, v_j\} \cap E|$
- Inzidenzmatrix $B=(b_{ij})_{i\in[n],j\in[m]}$
  - $b_{ij} = |\{v_i\} \cap e_j|$
- Adjazenzlisten

Speicherung	Speicherplatz	$(v,w)\in E$?	deg(v)	$\exists v: \deg(v)=0$?
Adjazenzmatrix	$\Theta(n^2)$	$O(1)$	$O(n)$	$O(n^2)$
Inzidenzmatrix	$\Theta(nm)$	$O(m)$	$O(m)$	$O(nm)$
Adjazenzliste	$\Theta(n+m)$	$O(\min(\deg(v),\deg(w)))$	$O(\deg(v))$	$O(n)$

Effiziente Algorithmen (1)

- Eine der wichtigsten Ressourcen eines Algorithmus ist Zeit.
- Algorithmen, deren Laufzeiten durch ein Polynom beschränkt sind, nennt man effizient.
  - Laufzeit beschränkt durch n^k für konstantes k
- Essentielle Frage: Existiert für ein gegebenes Problem ein effizienter Algorithmus?
  - Wenn ja: Wie lautet die asymptotische Komplexität des Algorithmus?
- Warum ausgerechnet *polynomielle Laufzeit*?

Polynomielle vs. exponentielle Algorithmen

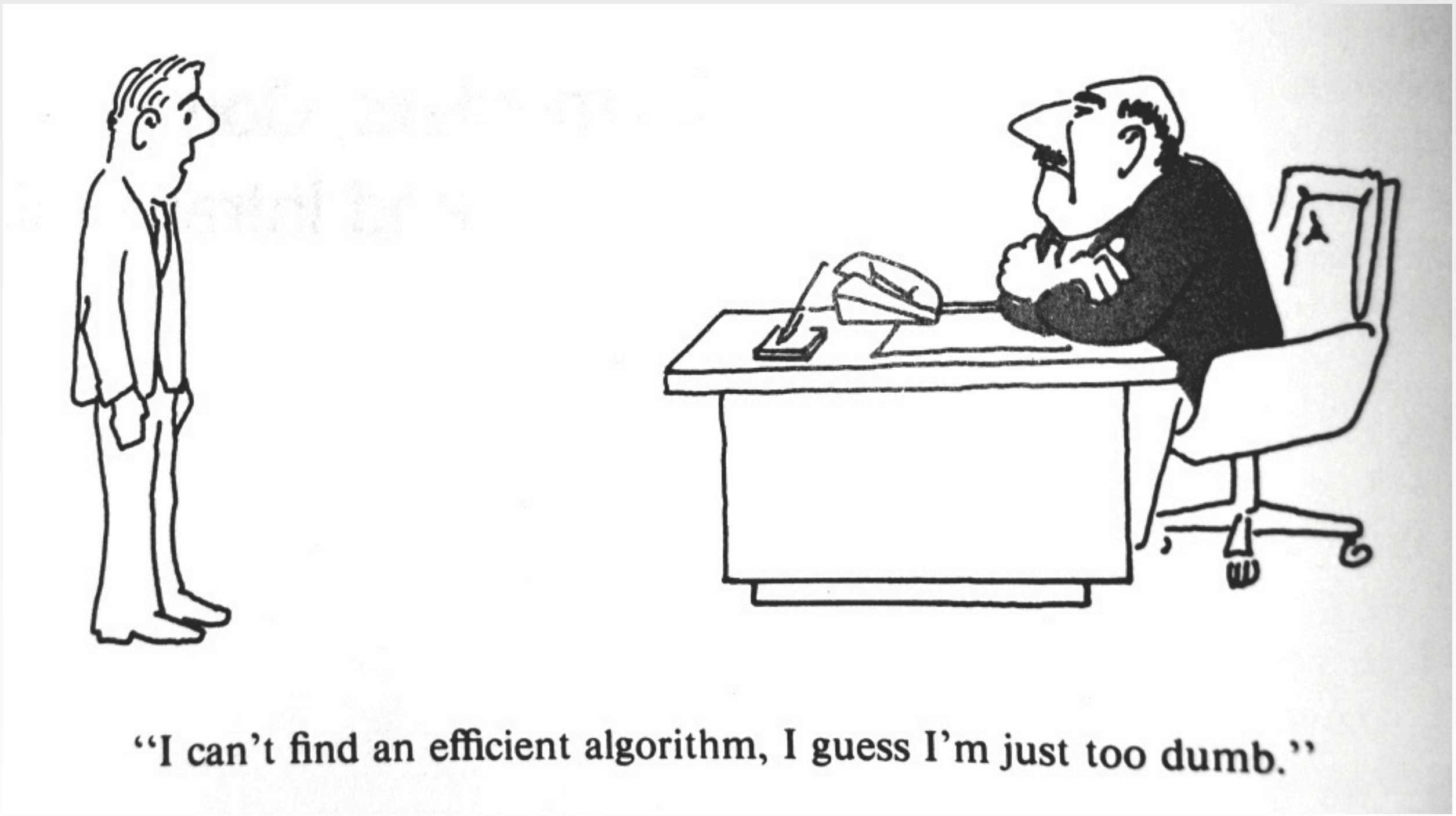
[Garey & Johnson, 1979]

	10	20	30	40	50	60
n	0.00001 Sek.	0.00002 Sek.	0.00003 Sek.	0.00004 Sek.	0.00005 Sek.	0.00006 Sek.
n^2	0.0001 Sek.	0.0004 Sek.	0.0009 Sek.	0.0016 Sek.	0.0025 Sek.	0.0036 Sek.
n^3	0.001 Sek.	0.008 Sek.	0.027 Sek.	0.064 Sek.	0.125 Sek.	0.216 Sek.
n^5	1 Sek.	3.2 Sek.	24.3 Sek.	1.7 Min.	5.2 Min.	13.0 Min.
2^n	0.001 Sek.	1.0 Sek.	17.9 Min.	12.7 Tage	35.7 Jahre	366 Jahrhunderte
3^n	0.059 Sek.	58 Min.	6.5 Jahre	3855 Jahrhunderte	$2 \cdot 10^8$ Jahrhunderte	$1.3 \cdot 10^{13}$ Jahrhunderte

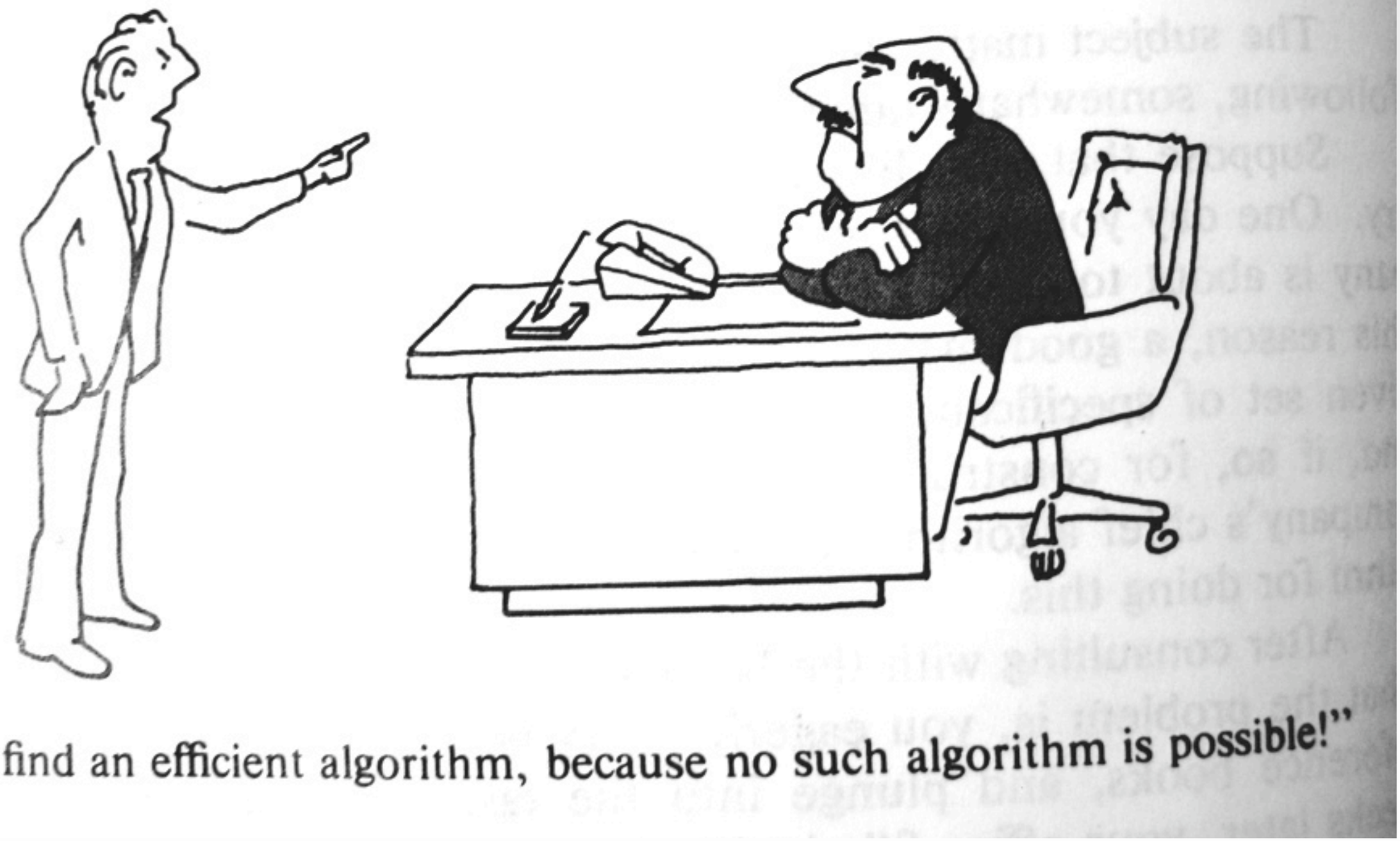
Effiziente Algorithmen (2)

- Wie findet man einen effizienten Algorithmus?
  - Häufig mit Standardtechniken wie greedy, divide and conquer, dynamische Programmierung, lineare Optimierung, Reduktion auf ein Problem für das man einen effizienten Algorithmus kennt, usw.
- Wie kann man zeigen, dass kein effizienter Algorithmus existiert?
  - In den meisten Fällen gar nicht.
  - Aber häufig kann man eine ähnlich starke Aussage treffen: NP-Schwere.

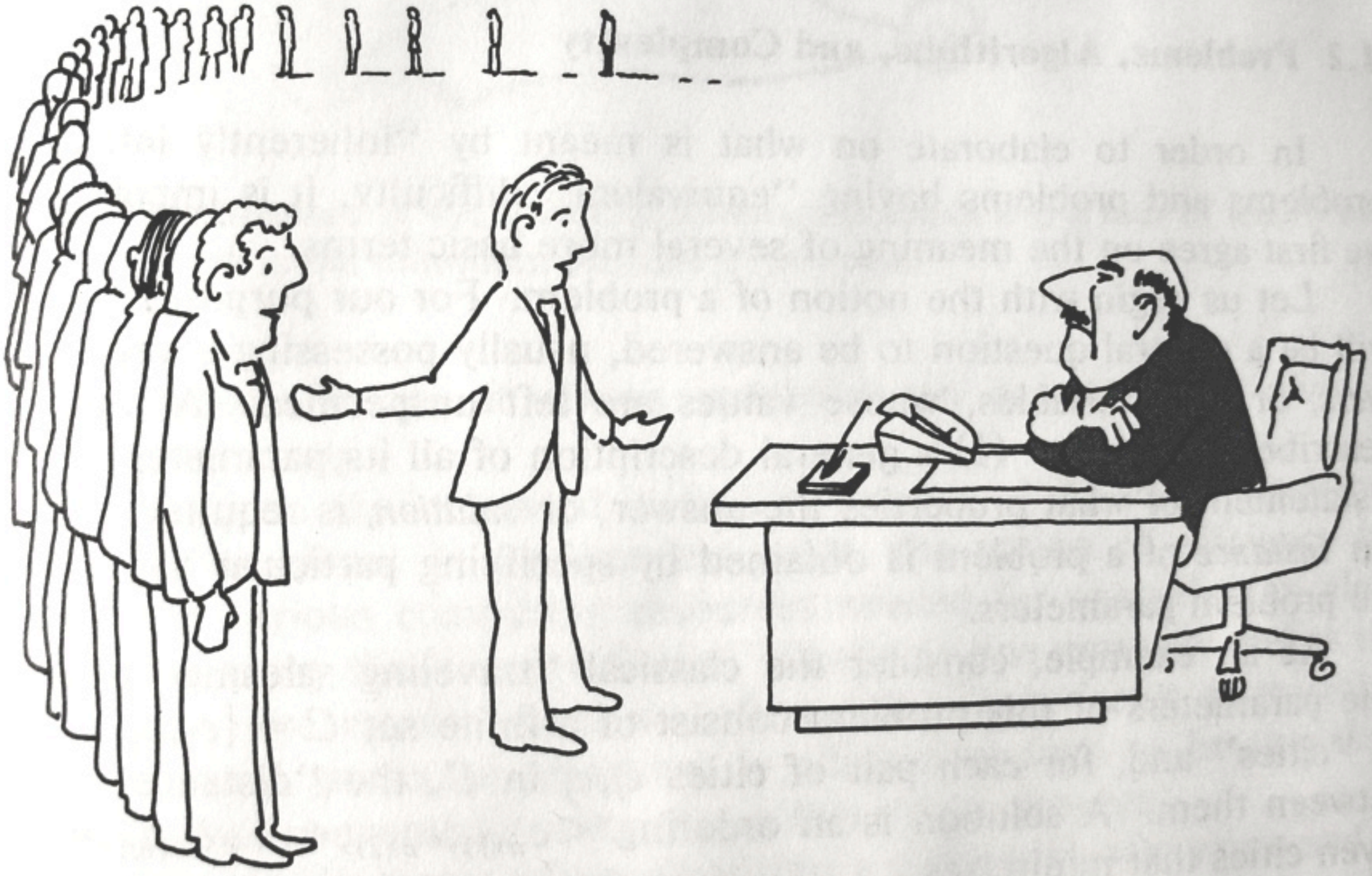
[Garey & Johnson, 1979]



"I can't find an efficient algorithm, I guess I'm just too dumb."



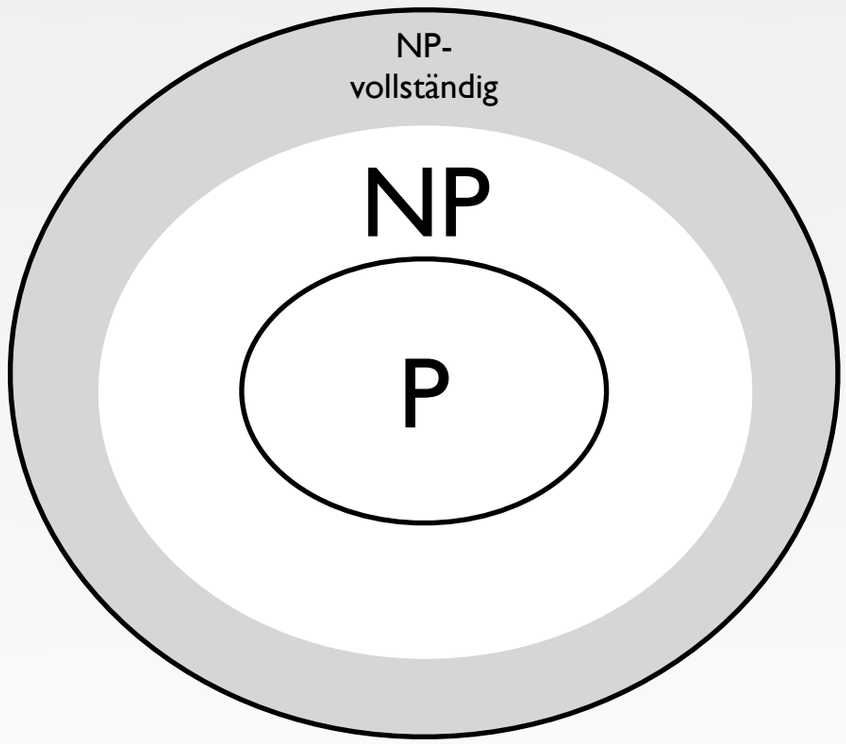
"I can't find an efficient algorithm, because no such algorithm is possible!"



"I can't find an efficient algorithm, but neither can all these famous people."

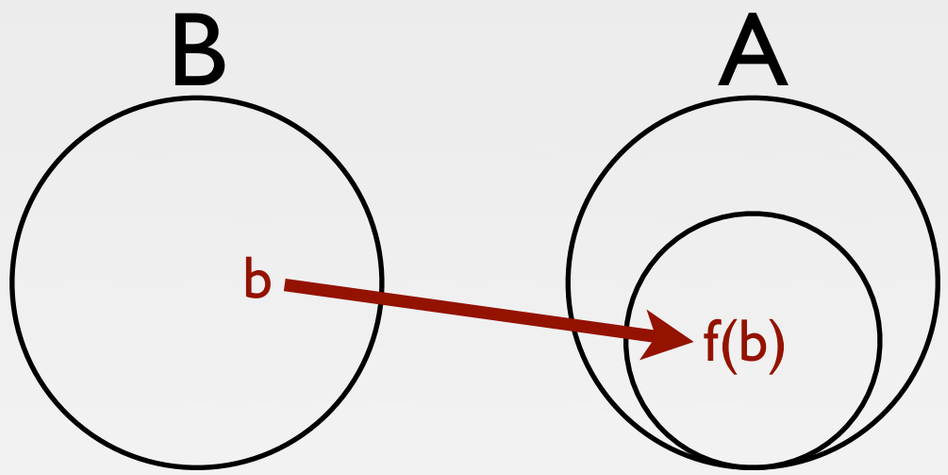
Algorithmen und Komplexität

- P (in polynomieller Zeit lösbar auf einer deterministischen Turingmaschine)
  - Für Probleme in P existieren effiziente Algorithmen.
- NP (in polynomieller Zeit lösbar auf einer nicht-deterministischen Turingmaschine)
  - In polynomieller Zeit verifizierbar
  - Für NP-*schwere* Probleme existieren keine effizienten Algorithmen wenn P≠NP.



Schwere und einfache Probleme

- Wie zeigt man, dass Problem A
  - in P ist?
    - polynomieller Algorithmus für A
  - NP-schwer ist?
    - Reduktionsbeweis
    - Formuliere A als Entscheidungsproblem.
    - Wähle ein NP-schweres Entscheidungsproblem B.
    - Gebe eine effizient berechenbare Funktion f an, die zu jeder Instanz eines B-Problems b eine Instanz eines A-Problems f(b) liefert, so dass f(b) genau dann *wahr* ist, wenn b *wahr* ist.



- SAT (Erfüllbarkeit einer prädikatenlogischen Formel)
  - NP-vollständig, insbesondere für konjunktive Normalform, selbst wenn eine Klausel nur drei Literale enthält (3SAT).

$$(v_1 \vee \bar{v}_2 \vee v_3) \wedge (\bar{v}_1 \vee v_2 \vee \bar{v}_3) \wedge \ldots$$

NP-schwere Probleme

- NP-schwer
  - mindestens so schwer wie alle Probleme in NP (bezüglich polynomieller Reduktionen)
- NP-vollständig
  - NP-schwer und in NP
- Was macht man mit NP-schweren Problemen?
  - Problemklasse einschränken
  - Parametrisierung
  - Heuristik
  - Randomisierung
  - Approximation

Approximation

- Bei schwierigen Optimierungsproblemen bietet es sich an, die Lösung zu approximieren.
- Maximierungsproblem
  - $Alg(I) \geq c \cdot Opt(I)$
- Minimierungsproblem
  - $Alg(I) \leq c \cdot Opt(I)$
- Beispiele für c-Approximationen
  - Maximale Clique (=vollständig verbundener Teilgraph)
    - 1/|V|-Approximation
  - MAX-SAT
    - 1/2-Approximation