Technische Universität Braunschweig

Algorithmen und Datenstrukturen II

12. Vorlesung

Linda Kleist, 17.07.2019

Vorlesungsinhalte

algorithmische Techniken

Greedy

dynamische Programmierung

Branch & Bound

Approximations-algorithmen

...

weiterführende Datenstrukturen

Hashing

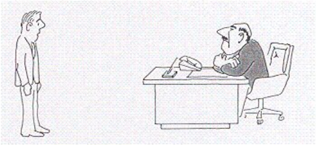
...

leicht und *schwer*?

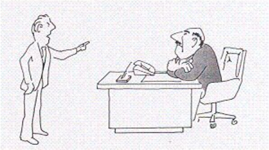
Komplexitäts-Theorie

Vorlesungsinhalte

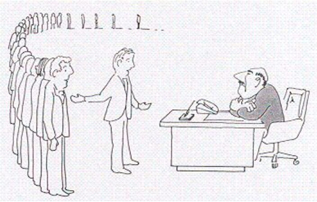
How to explain to your boss as to why your program is so slow…



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.

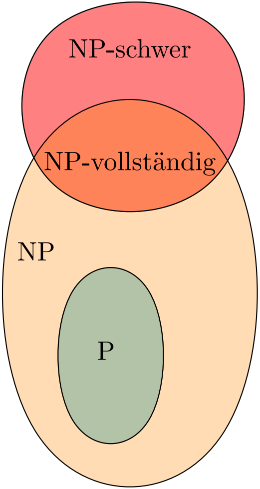
Komplexitätsklassen

Ein Problem Prob ist

- in P, wenn ein Algorithmus existiert, der
  - für jede Instanz
  - in polynomieller Zeit
  - eine korrekte Lösung liefert.
- in NP, wenn ein Algorithmus existiert, der
  - für jede Instanz
  - in polynomieller Zeit
  - eine Lösung verifiziert.

Ein Entscheidungsproblem Prob ist

- NP-schwer, wenn sich 3-Sat auf Prob reduzieren lässt.
- NP-vollständig, wenn es sowohl in NP liegt als auch NP-schwer ist.

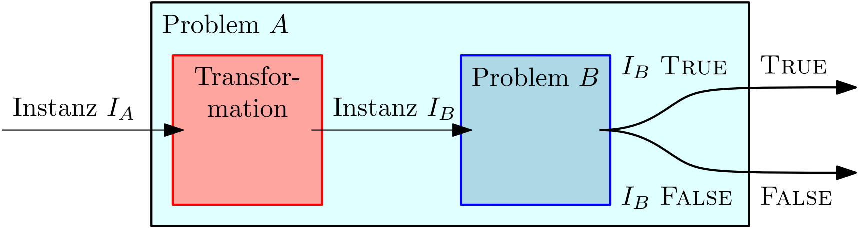


Reduktion

Ein Entscheidungsproblem A lässt sich zu einem Entscheidungsproblem B reduzieren, wenn

- ein Polynomialzeit-Algorithmus existiert, der
- jede Instanz I_A von Problem A in eine Instanz I_B von B transformiert,
- sodass I_A genau dann wahr ist wenn I_B wahr ist.

Wir schreiben $A \leq_p B$.



Reduktion

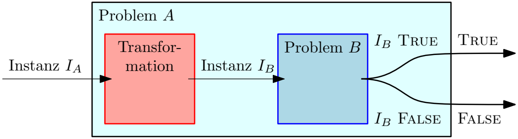
Ein Entscheidungsproblem A lässt sich zu einem Entscheidungsproblem B reduzieren, wenn

- ein Polynomialzeit-Algorithmus existiert, der
- jede Instanz I_A von Problem A in eine Instanz I_B von B transformiert,
- sodass I_A genau dann wahr ist wenn I_B wahr ist.

Wir schreiben $A \leq_p B$.

Fakt

$A \leq_p B$ und $B \in \mathrm{P}$
$\implies A \in \mathrm{P}$



Reduktion

Ein Entscheidungsproblem A lässt sich zu einem Entscheidungsproblem B reduzieren, wenn

- ein Polynomialzeit-Algorithmus existiert, der
- jede Instanz I_A von Problem A in eine Instanz I_B von B transformiert,
- sodass I_A genau dann wahr ist wenn I_B wahr ist.

Wir schreiben $A \leq_p B$.

Fakt

$A \leq_p B$ und $B \in \mathrm{P}$
$\implies A \in \mathrm{P}$

Fakt

$A \in \mathrm{NP}$ und B ist NP-schwer
$\implies A \leq_p B$

Reduktion

Ein Entscheidungsproblem A lässt sich zu einem Entscheidungsproblem B reduzieren, wenn

- ein Polynomialzeit-Algorithmus existiert, der
- jede Instanz I_A von Problem A in eine Instanz I_B von B transformiert,
- sodass I_A genau dann wahr ist wenn I_B wahr ist.

Wir schreiben $A \leq_p B$.

Fakt

$A \leq_p B$ und $B \in \mathrm{P}$
$\implies A \in \mathrm{P}$

Fakt

$A \in \mathrm{NP}$ und B ist NP-schwer
$\implies A \leq_p B$

Folgerung

$B \in \mathrm{P}$ und B ist NP-schwer
$\implies \mathrm{P} = \mathrm{NP}$.

Rucksack-Probleme

teilen möglich

ganz oder gar nicht

Rucksack-Probleme

	Fractional Knapsack	Maximum Knapsack				01- Knapsack	
Typ	Optimierungs-problem	Optimierungsproblem				Entscheidungs-problem	
Kom-plexität	P	NP-schwer				NP-vollständig	
Algo-rithmen	Greedy	Greedy_0	Greedy_k	DP	B&B	DP	B&B
Be-merkung	optimal	beliebig schlecht	$\frac{k}{k+1}$-Approx	optimal	optimal	–	–
Laufzeit	$O(n \log n)$	$O(n \log n)$	$O(n^{k+2})$	$O(nZ)$	$O(n2^n)$	$O(nZ)$	$O(n2^n)$

- Subset Sum: NP-vollständig, DP
- Partition: NP-vollständig, DP
- Bin Packing: NP-schwer, c-Approximation mit $c < \frac{3}{2} \implies \mathrm{P} = \mathrm{NP}$

Hörsaal-Probleme

Problem	Hörsaal-Belegung[1]			Hörsaal-Auslastung
Typ	Optimierungsproblem			Optimierungsproblem
Komplexität	P			P
Algorithmen	Greedy-Intervalllänge	Greedy-Überlappung	Greedy-Endzeit	DP
Bemerkung	$\frac{1}{2}$-Approx	$\frac{1}{2}$-Approx	optimal	optimal
Laufzeit	$O(n^2)$	$O(n^2)$	$O(n \log n)$	$O(n^2)$

- Variante: Hörsaal-Zuweisung/ Färbung von Intervallgraphen

[1] Das Problem fragt nach einer größten unabhängigen Menge eines Intervallgraphen.

Graphen-Probleme

Problem	(EUKLIDISCHE) RUNDREISE			MIN VERTEX COVER
Typ	Optimierungsproblem			Optimierungsproblem
Komplexität	NP-schwer			NP-schwer
Algorithmen	Brute Force	B&B	DP	inklusions-maximales Matching
Bemerkung	optimal	optimal	optimal	2-Approximation
Laufzeit	$O(n!)$	$O(n^2 2^{n^2})$	$O(n^2 2^n)$	$O(n^2)$

- unabhängige Menge
- Clique
- k-Färbung

...Klausursituation...