Computergrundlagen
Komplexitätstheorie

Axel Arnold

Institut für Computerphysik
Universität Stuttgart

Wintersemester 2010/11

Zeitschranken

Definition

Sei M eine Turingmaschine.
T heißt Zeitschranke für M, falls M für alle Eingaben ω nach $\mathcal{O}(T(|\omega|))$ Schritten anhält.

- $|\omega|$ bezeichnet die Länge der Eingabe, „abcde“ hat etwa die Länge 5, die Zahl 123 die Länge 3 (oder 7 in Binärdarstellung)
- Da wir nur an der Größenordnung interessiert sind, ist der Faktor durch die Kodierung egal
- Kann auch über die Anzahl Schritte eines Algorithmus oder Python-Programms definiert werden, wenn nur die Grundrechenarten und Schleifen benutzt werden dürfen
- Die niedrigste Schranke T heißt auch die **Komplexität** eines Algorithmus

Entscheidungsprobleme

Definition

Ein Entscheidungsproblem ist Untermenge L des Alphabets Σ^*, für die es eine Turingmaschine M gibt, so dass M für alle Eingaben ω anhält und genau dann „ja“ ausgibt, wenn $\omega \in L$.
Man sagt, dass M das Problem L entscheidet, das Problem L ist entscheidbar.

Beispiele

- Die Menge aller Primzahlen ist entscheidbar
- Die Menge aller sortierten Listen ist entscheidbar
- Das Halteproblem ist *nicht* entscheidbar

Entscheidungsprobleme

Definition

Ein Entscheidungsproblem ist Untermenge L des Alphabets Σ^*, für die es eine Turingmaschine M gibt, so dass M für alle Eingaben ω anhält und genau dann „ja“ ausgibt, wenn $\omega \in L$.
Man sagt, dass M das Problem L entscheidet, das Problem L ist entscheidbar.

- Ein Entscheidungsproblem ist immer eine Frage: „Ist die Eingabe p eine Primzahl?“
- Alle terminierenden Algorithmen können auch als Entscheidungsproblem formuliert werden

Komplexitätszeitklassen

Definition

$TIME(T)$ sei die Klasse aller Probleme L, für die es eine Turingmaschine mit Zeitschranke T gibt, die L entscheidet.

$P := \bigcup_{p \text{ Polynom}} TIME(p)$ sei die Klasse aller in polynomieller Zeit lösbaren Probleme.
$EXP := \bigcup_{p \text{ Polynom}} TIME(2^p)$ sei die Klasse aller in exponentieller Zeit lösbaren Probleme.

- Probleme in *EXP* gelten als schwer lösbar
- Probleme in *P* als lösbar, da die Exponenten meist klein sind
- Kryptographie beruht darauf, dass einige Probleme (vermutlich) nicht in *P* liegen

Komplexitätszeitklassen

Definition

$TIME(T)$ sei die Klasse aller Probleme L, für die es eine Turingmaschine mit Zeitschranke T gibt, die L entscheidet.

$P := \bigcup_{p \text{ Polynom}} TIME(p)$ sei die Klasse aller in polynomieller Zeit lösbaren Probleme.
$EXP := \bigcup_{p \text{ Polynom}} TIME(2^p)$ sei die Klasse aller in exponentieller Zeit lösbaren Probleme.

- Jedes Problem kann in polynomieller Zeit in ein Entscheidungsproblem übersetzt werden
- Das Entscheidungsproblem zu einem beliebigen polynomiellen Algorithmus ist also selber polynomiell

Beispiel: Grundrechenarten sind in *P*

- Addition ist in *P*

Beweis

Betrachte zwei n- und m-stellige Zahlen a und b.
Stellenweise Addition benötigt $2\max(n, m) + 1$ Schritte

- Multiplikation ist in *P*

Beweis

Für jede Stelle von a wird b um einige Stellen verschoben einmal zu sich selber addiert. Diese Addition braucht für die oberen Stellen $2(m + n) + 1$ Schritte, insgesamt benötigt die Multiplikation $\mathcal{O}(m^2 \cdot n)$ Schritte.

- Bemerkung: es reicht nicht, a-mal b zu sich selber zu addieren, da dies Komplexität $\mathcal{O}(a \cdot m) = \mathcal{O}(2^n m)$ hat

Beispiel: Probleme in $EXP \setminus P$

- Die Exponentialfunktion $\exp(x) = 2^x$ ist in $EXP \setminus P$

Beweis

Betrachte n-stellige Zahl a. Dann kann 2^a durch a-malige Multiplikation von 2 mit sich selber berechnet werden. Also $\exp \in EXP$.
Andererseits hat 2^a $a/\log(2)$ Stellen. Jeder Algorithmus für exp muss jede Stelle einmal schreiben, hat also wenigstens $\mathcal{O}(a) = 2^n$ Schritte. Also $\exp \notin P$.

- Fakultät ist entsprechend auch nicht in P
- P und EXP sind also echt verschiedene Komplexitätsklassen
- Gibt es noch Klassen dazwischen?

Nichtdeterministische Touringmaschinen

Definition

Eine nichtdeterministische Turingmaschine M entscheidet ein Problem L genau dann, wenn sie bei allen Eingaben anhält und gilt:
$\omega \in L \Leftrightarrow$ es existiert ein $y \in \Sigma^*$, so dass M bei Eingabe (ω, y) „ja“ antwortet.

***NTIME*(T) sei die Klasse aller Probleme L, für die es eine nichtdeterministische Turingmaschine mit Zeitschranke T gibt, die L löst.**
NP $:= \bigcup_{p \text{ Polynom}}$ ***NTIME*(p).**

- Es gibt sozusagen einen Dämonen, der der Turingmaschine hilft.
- Oder: die Turingmaschine darf raten, wie sie weitermachen soll.

$P \subseteq NP \subseteq EXP$

- $P \subseteq NP$

Beweis

Klar, da eine deterministische Turingmaschine die Zusatzeingabe einfach ignorieren kann.

- $NP \subseteq EXP$

Beweis

Sei M eine nd. TM und T eine polynomielle Zeitschranke für M. Diese kann bei Eingabe ω nur Zusatzeingaben bis Länge $p(|\omega|)$ lesen, von denen es $\mathcal{O}(2^{p(|\omega|)})$ gibt. Durchprobieren liefern eine deterministische Turingmaschine mit exponentieller Zeitschranke.

- $P \neq NP$ ist eine zentrale Frage der theoretischen Informatik

NP: das HANDELSREISENDEN-Problem

Gegeben

m Orte, eine $m \times m$-Matrix E mit den Abständen der Orte voneinander und eine Schranke S.

Frage

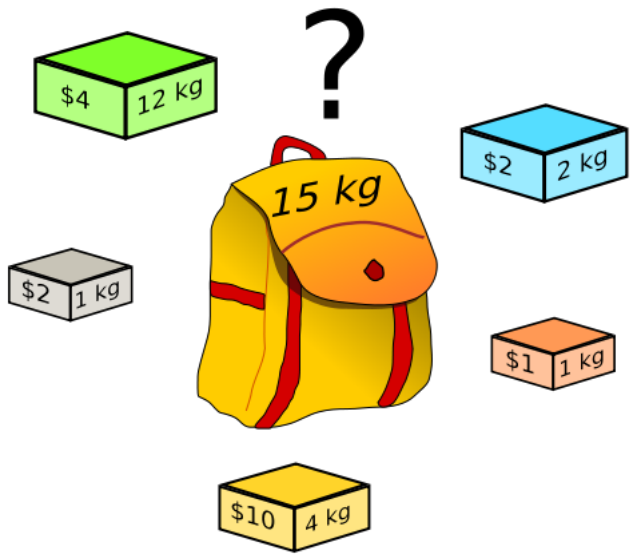
Gibt es eine Rundreise durch alle m Orte, die jeden Ort genau einmal besucht und kürzer als S ist?

- Idee: Ein Handelsreisender soll in 2 Tagen alle großen Städte in Deutschland besuchen. E enthält dann die Reisezeiten.
- Im allgemeinen müssen die Abstände nicht euklidisch sein, da es ja für eine bestimmte Verbindung nur die Bahn geben könnte, für andere einen Flug.

NP: das RUCKSACK-Problem

Gegeben

Zwei Mengen Werte $w_1, ..., w_m$ und Gewichte $g_1, ..., g_m$ sowie Schranken W und G.



Frage

Gibt es eine Menge $I \subseteq \{1, ..., m\}$, so dass $\sum_{i \in I} w_i \geq W$ und $\sum_{i \in I} g_i \leq G$?

- Idee: Wir packen (kleine) Gegenstände in einen Rucksack, der maximal G kg tragen kann. Können wir damit einen Wert von mindestens W€ mitnehmen?
- Dieses Problem tritt zum Beispiel bei der Optimierung von Investment-Portfolios auf.

NP: das 3SAT-Problem

Gegeben

Ein boolescher Ausdruck

$$F = (a_{11} \vee a_{12} \vee a_{13}) \wedge (a_{21} \vee a_{22} \vee a_{23}) \wedge \ldots \wedge (a_{n1} \vee a_{n2} \vee a_{n3})$$

mit $a_{ij} \in \{a_1, \ldots, a_m, \neg a_1, \ldots, \neg a_m\}$.

Frage

Gibt es eine Belegung von $a = (a_1, \ldots, a_n)$ mit Wahrheitswerten, so dass F wahr ist?

- 2SAT mit nur 2 Variablen pro Term ist in P.
- Eine nichtdeterministische Turingmaschine für 3SAT interpretiert Zusatzeingabe als a und berechnet F. Ist F wahr, so antwortet sie „ja“, sonst „nein“.

Reduktionen

Definition

Seien $L, M \subset \Sigma^*$. L heißt polynomiell reduzierbar auf M, wenn es eine in polynomieller Zeit berechenbare Funktion f gibt, so dass $\omega \in L \Leftrightarrow f(\omega) \in M$.
Man schreibt kurz $L \leq_p M$.

- Ist $L \leq_p M$ und $M \leq_p N$, so ist $L \leq_p N$ (Transitivität)
- Ist $L \in P$ und $M \leq_p L$, so ist $M \in P$ (Abgeschlossenheit)
- Entsprechend: Ist $L \in NP$ und $M \leq_p L$, so ist $M \in NP$

NP-Vollständigkeit

Definition

Ein Problem L heißt *NP*-vollständig, falls $L \in NP$ und für alle Probleme $M \in NP$ gilt: $M \leq_p L$.

- Kann ich ein *NP*-vollständiges Problem effizient lösen, dann alle Probleme in *NP*
- Ist ein *NP*-vollständiges Problem in *P*, dann ist $P = NP$
- Gibt es ein *NP*-vollständiges Problem, das nicht in *P* liegt, so liegt kein *NP*-vollständiges Problem in *P*
- Ist $L \in NP$ und $M \leq_p L$ für ein *NP*-vollständiges Problem *M*, dann ist *L* *NP*-vollständig

NP-Vollständigkeit

Definition

Ein Problem L heißt *NP*-vollständig, falls $L \in NP$ und für alle Probleme $M \in NP$ gilt: $M \leq_p L$.

- Satz von Cook: 3SAT ist *NP*-vollständig

Beweis

Wir simulieren eine TM mit (sehr vielen) booleschen Variablen. Es gibt eine Variable für jedes Paar aus Zustand und Zeitschritt, Bandposition und Triplets aus Zeichen, Bandposition und Zeitschritt. Die booleschen Terme repräsentieren alle möglichen Übergänge.

- Durch Reduktionen auf 3SAT kennt man mittlerweile *viele* *NP*-vollständige Probleme

NP-Vollständigkeit

Definition

Ein Problem L heißt *NP*-vollständig, falls $L \in NP$ und für alle Probleme $M \in NP$ gilt: $M \leq_p L$.

NP-vollständig sind:

- das HANDELSREISENDEN-Problem
- das RUCKSACK-Problem
- Minesweeper: Gegeben ein Feld mit Anzahlen benachbarter Minen, gibt es eine gültige Belegung mit Minen?
- Es gibt aber auch Probleme in *NP*, die nicht *NP*-vollständig sind...
- Die Faktorisierung ist vermutlich nicht *NP*-vollständig

Die Klasse BQP

Definition

Die Klasse *BQP* (bounded error quantum polynomial time) beschreibt alle Probleme, die sich mit Hilfe eines Quantencomputers in polynomieller Zeit berechnen lassen mit einer Fehlerwahrscheinlichkeit $< \frac{1}{4}$.

- Durch Wiederholen der Berechnung lässt sich der Fehler beliebig verringern, da die Fehlerwahrscheinlichkeit exponentiell abnimmt mit der Zahl der Messungen
- Klar: $P \subset BQP$
- Unklar: $BQP \subseteq NP$, $NP \subseteq BQP$
- Faktorisierung ist in *BQP* (Algorithmus von Shor)