Vorlesung

Theoretische Informatik II

Bernhard Beckert

Institut für Informatik

UNIVERSITÄT KOBLENZ · LANDAU

Wintersemester 2007/2008

Dank

Diese Vorlesungsmaterialien basieren zum Teil auf den Folien zu den Vorlesungen von

Katrin Erk (gehalten an der Universität Koblenz-Landau)

Jürgen Dix (gehalten an der TU Clausthal)

Christoph Kreitz (gehalten an der Universität Potsdam)

Ihnen gilt mein herzlicher Dank.

– Bernhard Beckert, Oktober 2007

Teil VI

Komplexitätstheorie

Welche Arten von Komplexität gibt es?

- Zeit
- Speicher

DTIME und NTIME

Definition 15.1 (NTIME$(T(n))$, DTIME$(T(n))$)

Basismodell: k-DTM $\mathcal{M}$ (ein Band für die Eingabe).
Wenn $\mathcal{M}$ mit jedem Eingabewort der Länge n höchstens $T(n)$ Schritte macht, dann wird sie $T(n)$**-zeitbeschränkt** genannt.
Die von $\mathcal{M}$ akzeptierte Sprache hat **Zeitkomplexität** $T(n)$ (tatsächlich meinen wir $\max(n+1, \lceil T(n) \rceil)$).

- **DTIME**$(T(n))$ ist die Klasse der Sprachen, die von $T(n)$-zeitbeschränkten, DTMn akzeptiert werden.
- **NTIME**$(T(n))$ ist die Klasse der Sprachen, die von $T(n)$-zeitbeschränkten, NTMn akzeptiert werden.

DSPACE und NSPACE

Definition 15.2 (NSPACE$(S(n))$, DSPACE$(S(n))$)

Basismodell: k-DTM $\mathcal{M}$ davon ein spezielles Eingabeband (**offline DTM**). Wenn $\mathcal{M}$ für jedes Eingabewort der Länge n maximal $S(n)$ Zellen auf den Ablagebändern benutzt, dann heißt $\mathcal{M}$ $S(n)$**-speicherbeschränkt.**
Die von $\mathcal{M}$ akzeptierte Sprache hat **Speicherkomplexität** $S(n)$ (tatsächlich meinen wir $\max(1, \lceil S(n) \rceil)$)

- **DSPACE**$(S(n))$ ist die Klasse der Sprachen, die von $S(n)$-speicherbeschränkten, DTMn akzeptiert werden.
- **NSPACE**$(S(n))$ ist die Klasse der Sprachen, die von $S(n)$-speicherbeschränkten, NTMn akzeptiert werden.

Wichtige Fragen

Wichtige Fragen (1)

Zeit: Wird jede Sprache aus **DTIME**$(f(n))$ von einer DTM entschieden?

Speicher: Wird jede Sprache aus **DSPACE**$(f(n))$ von einer DTM entschieden?

Die Funktionen f sind immer sehr einfache Funktionen, insbesondere sind sie alle berechenbar. In dieser Vorlesung betrachten wir nur Potenzen $f(n) = n^i$.

Wichtige Fragen (2)

Zeit/Speicher: Wie sieht es mit **NTIME**$(f(n))$, **NSPACE**$(f(n))$ aus?

Zeit vs. Speicher: Welche Beziehungen gelten zwischen **DTIME**$(f(n))$, **DSPACE**$(f(n))$, **NTIME**$(f(n))$, **NSPACE**$(f(n))$?

Wichtige Fragen

Halten, hängen nach n Schritten.

Zeitbeschränkt: Was bedeutet es, daß **eine DTM höchstens n Schritte macht**?

Halten?: Streng genommen, müßte sie dann entweder **halten** oder **hängen**. Das bedeutet, im ersten Fall, daß die Eingabe **akzeptiert** wird.

Hängen?: DTM'n auf beidseitig unendlichem Band können aber nicht hängen.

Wichtige Fragen

Stoppen nach n Schritten.

Stoppen: Wir wollen unter **eine DTM macht höchstens n Schritte** folgendes verstehen:

- Sie **hält** (und **akzeptiert** die Eingabe) innerhalb von n Schritten.
- Sie **hängt** (und **akzeptiert** die Eingabe **nicht**) innerhalb von n Schritten.
- Sie **stoppt** innerhalb von n Schritten **ohne in den Haltezustand überzugehen**. Auch hier wird die Eingabe nicht akzeptiert.

Antworten

Antworten (informell)

Zeit: Jede Sprache aus $\mathbf{DTIME}(f(n))$ ist entscheidbar: Man warte einfach solange wie $f(n)$ angibt. Falls bis dahin kein “Ja” gekommen ist, ist die Antwort eben “Nein”.

Speicher: Jede Sprache aus $\mathbf{DSPACE}(f(n))$ ist entscheidbar. Denn es gibt nur **endlich viele verschiedene Konfigurationen**. Falls also die DTM nicht terminiert (was passiert), macht man folgendes: man schreibt alle Konfigurationen auf (die komplette Rechnung). Falls die DTM nicht terminiert, muss sie in eine Schleife laufen (eine Konfiguration erreichen, die sie schon einmal hatte). Das lässt sich feststellen.

Antworten

Antworten (informell)

NTM vs. DTM: Trivial sind $\mathbf{DTIME}(f(n)) \subseteq \mathbf{NTIME}(f(n))$ und $\mathbf{DSPACE}(f(n)) \subseteq \mathbf{NSPACE}(f(n))$. Versucht man aber eine NTM durch eine DTM zu simulieren, braucht man (vermutlich) exponentiell mehr Zeit.
Für die Speicherkomplexität kann man zeigen:
$\mathbf{NSPACE}(f(n)) \subseteq \mathbf{DSPACE}(f^2(n))$.

Antworten

Antworten (informell):

Zeit vs. Speicher: Trivial sind $\mathbf{DTIME}(f(n)) \subseteq \mathbf{DSPACE}(f(n))$ und $\mathbf{NTIME}(f(n)) \subseteq \mathbf{NSPACE}(f(n))$.

Aber $\mathbf{DSPACE}(f(n))$, $\mathbf{NSPACE}(f(n))$ sind viel größer.

Bandkompression

Konstante Faktoren werden ignoriert

Nur **die funktionale Wachstumsrate einer Funktion in Komplexitätsklassen zählt**: Konstante Faktoren werden ignoriert.

Theorem 15.3 (Bandkompression)

Für jedes $c \in \mathbb{R}^+$ und jede Speicherfunktion $S(n)$ gilt:

$$\textbf{DSPACE}(S(n)) = \textbf{DSPACE}(cS(n))$$
$$\textbf{NSPACE}(S(n)) = \textbf{NSPACE}(cS(n))$$

Bandkompression

Beweis

Eine Richtung ist trivial. Die andere geht, indem man eine feste Anzahl r ($> \frac{2}{c}$) von benachbarten Zellen auf dem Band als **ein neues Symbol** darstellt. **Die Zustände der neuen Maschine simulieren die alten Kopfbewegungen als Zustandsübergänge** (innerhalb des neuen Symbols). D.h. für r Zellen der alten Maschine werden nun nur maximal 2 benutzt: im ungünstigsten Fall, wenn man von einem Block in den benachbarten geht.

Zeitbeschleunigung

Theorem 15.4 (Zeitbeschleunigung)

Für jedes $c \in \mathbb{R}^+$ *und jede Zeitfunktion* $T(n)$ *mit* $\lim_{n\to\infty} \frac{T(n)}{n} = \infty$ *gilt:*

$$\mathbf{DTIME}(T(n)) = \mathbf{DTIME}(cT(n))$$
$$\mathbf{NTIME}(T(n)) = \mathbf{NTIME}(cT(n))$$

Zeitbeschleunigung

Beweis

Eine Richtung ist trivial. Der Beweis der anderen läuft wieder über die Repräsentation einer festen Anzahl r ($> \frac{4}{c}$) benachbarter Bandzellen durch **neue Symbole**. Im Zustand der neuen Maschine wird wieder kodiert, welches Zeichen und welche Kopfposition (der simulierten, alten Maschine) aktuell ist. **Wenn die alte Maschine simuliert wird, muss die neue nur** 4 **statt** r **Schritte machen**: 2 um die neuen Felder zu drucken und weitere 2 um den Kopf auf das neue und wieder zurück auf das alte (im schlimmsten Fall) zu bewegen.

Wachstumsrate von DTIME und DSPACE

Lemma 15.5 (Wachstumsrate von DTIME, DSPACE)

Es sei $T(n)$ eine Zeitfunktion mit $\lim_{n\to\infty} \frac{T(n)}{n} = \infty$, und es sei $S(n)$ eine Speicherfunktion.

(a) *Es sei $f(n) = \mathbf{O}(T(n))$. Dann gilt:*
$\mathbf{DTIME}(f(n)) \subsetneqq \mathbf{DTIME}(T(n))$.

(b) *Es sei $g(n) = \mathbf{O}(S(n))$. Dann gilt:*
$\mathbf{DSPACE}(g(n)) \subsetneqq \mathbf{DSPACE}(S(n))$.

P, NP, PSPACE

Definition 15.6 (P, NP, PSPACE)

$$\begin{aligned} \mathbf{P} &:= \bigcup_{i \geqslant 1} \mathbf{DTIME}(n^i) \\ \mathbf{NP} &:= \bigcup_{i \geqslant 1} \mathbf{NTIME}(n^i) \\ \mathbf{PSPACE} &:= \bigcup_{i \geqslant 1} \mathbf{DSPACE}(n^i) \end{aligned}$$

P, NP, PSPACE

Intuitiv

- Probleme in **P** sind effizient lösbar, jene aus **NP** können in exponentieller Zeit gelöst werden.
- **PSPACE** ist eine sehr große Klasse, weit größer als **P** oder **NP**.

Komplexitätsklassen für Funktionen

Komplexitätsklassen für Funktionen

Eine Funktion $f : \mathbb{N} \to \mathbb{N}$ ist in **P**, falls es eine DTM $\mathcal{M}$ und ein Polynom $p(n)$ gibt, so daß für jedes n der Funktionswert $f(n)$ in höchstens $p(\text{länge}(n))$ Schritten von $\mathcal{M}$ berechnet wird. Dabei gilt $\text{länge}(n) = \lg n$, denn man braucht $\lg n$ Zeichen um die Zahl n binär darzustellen.

Analog funktioniert dies für alle anderen Komplexitätsklassen.

Beziehungen zwischen den Komplexitätsklassen

Frage

Was sind die genauen Beziehungen zwischen den Komplexitätsklassen P, NP, PSPACE?

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$$

Komplexitätsklassen

Frage

Wie zeigen wir, daß ein gegebenes Problem in einer bestimmten Klasse ist?

Antwort

Reduktion auf ein bekanntes!

Wir brauchen eines, mit dem wir anfangen können: SAT

Komplexitätsklassen

Frage

Können wir in **NP** Probleme finden, die **die schwierigsten in NP** sind?

Antwort

Es gibt mehrere Wege, ein schwerstes Problem zu definieren. Sie hängen davon ab, welchen **Begriff von Reduzierbarkeit** wir benutzen.

Für einen gegebenen Begriff von Reduzierbarkeit ist die Antwort: **Ja**. Solche Probleme werden **vollständig in der gegebenen Klasse** bezüglich des Begriffs der Reduzierbarkeit genannt.

Reduktion

Definition 15.7 (Polynomial-Zeit-Reduzibilität)

Seien L_1, L_2 Sprachen.

Polynomial-Zeit: L_2 ist Polynomial-Zeit reduzibel auf L_1, bezeichnet mit $L_2 \preceq_{\text{pol}} L_1$, wenn es eine **Polynomial-Zeit beschränkte DTM** gibt, die für jede Eingabe w eine Ausgabe $f(w)$ erzeugt, so daß

$$w \in L_2 \text{ gdw } f(w) \in L_1$$

Reduktion

Lemma 15.8 (Polynomial-Zeit-Reduktionen)

1. *Sei L_2 Polynomial-Zeit-reduzibel auf L_1. Dann gilt*

   L_2 *ist in* **NP** *wenn* L_1 *in* **NP** *ist*
   L_2 *ist in* **P** *wenn* L_1 *in* **P** *ist*

2. *Die Komposition zweier Polynomial-Zeit-Reduktionen ist wieder eine Polynomial-Zeit-Reduktionen.*