

Grundlagen der Theoretischen Informatik

Komplexitätstheorie (II)

11.07.2019

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Übersicht

1. Motivation
2. Terminologie
3. Endliche Automaten und reguläre Sprachen
4. Kellerautomaten und kontextfreie Sprachen
5. Turingmaschinen und rekursiv aufzählbare Sprachen
6. Berechenbarkeit, (Un-)Entscheidbarkeit
7. Komplexitätsklassen P und NP

Komplexitätstheorie

Welche Arten von Komplexität gibt es?

- Zeit
- Speicher

DTIME und NTIME

Definition [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 DTMs akzeptiert werden.
- **NTIME($T(n)$)** ist die Klasse der Sprachen, die von $T(n)$ -zeitbeschränkten NTMs akzeptiert werden.

DSPACE und NSPACE

Definition [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 DTMs akzeptiert werden.
- **NSPACE($S(n)$)** ist die Klasse der Sprachen, die von $S(n)$ -speicherbeschränkten NTMs akzeptiert werden.

DSPACE und NSPACE

Wieso eine *offline*-Turing-Maschine?

Bandbeschränkung von weniger als linearem Wachstum.

Beispiel Zu welcher Zeit-/Speicherkomplexitätsklasse gehört

$$\mathcal{L}_{\text{mirror}} := \{wcw^R : w \in (0 + 1)^*\},$$

also die Menge aller Wörter, die um den mittleren Buchstaben c gespiegelt werden können?

DSPACE und NSPACE

Beispiel (Forts.)

Zeit: $\text{DTIME}(n+1)$. Die Eingabe rechts vom c wird einfach in umgekehrter Reihenfolge kopiert. Wenn das c gefunden wird, wird einfach der übrige Teil (das w) mit der Kopie von w auf dem Band verglichen.

Speicher: Die gerade beschriebene Maschine liefert eine Schranke von $\text{DSPACE}(n)$.

Geht es noch besser?

Ja: $\text{DSPACE}(\lg n)$. Wir benutzen zwei Bänder als Binär-Zähler. Die Überprüfung der Eingabe auf das Auftreten von genau einem c benötigt keinen Speicher (kann mit einigen Zuständen getan werden). Als zweites prüfen wir Zeichen für Zeichen auf der linken und auf der rechten Seite: dazu müssen die zu prüfenden Positionen gespeichert werden (sie werden auf den beiden Bändern kodiert).

Man kommt auch mit einem einzigen Zähler aus (und einem Band).

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?

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

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's 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 **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 **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

- $\text{DTIME}(f(n)) \subseteq \text{NTIME}(f(n))$ und
- $\text{DSPACE}(f(n)) \subseteq \text{NSPACE}(f(n))$.

Versucht man aber eine NTM durch eine DTM zu simulieren, braucht man (vermutlich) exponentiell mehr Zeit.

Für die Speicherkomplexität kann man zeigen:

- $\text{NSPACE}(f(n)) \subseteq \text{DSPACE}(f^2(n))$.

Antworten

Antworten (informell)

Zeit vs. Speicher: Trivial sind

- $\text{DTIME}(f(n)) \subseteq \text{DSPACE}(f(n))$ und
- $\text{NTIME}(f(n)) \subseteq \text{NSPACE}(f(n))$.

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

Bandkompression/Zeitbeschleunigung

Konstante Faktoren werden ignoriert

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

Bandkompression

Theorem [Bandkompression]

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

$$\mathbf{DSPACE}(S(n)) = \mathbf{DSPACE}(cS(n))$$

$$\mathbf{NSPACE}(S(n)) = \mathbf{NSPACE}(cS(n))$$

Beweis Eine Richtung ist trivial. Die andere geht, indem man den Inhalt einer festen 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 [Zeitbeschleunigung]

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

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

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

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

Wachstumsrate von DTIME, DSPACE

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

(a) Es sei $f(n) \in \mathbf{O}(T(n))$. Dann gilt: $\mathbf{DTIME}(f(n)) \subseteq \mathbf{DTIME}(T(n))$.

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

Wachstumsrate von DTIME und DSPACE

Definition [P, NP, PSPACE]

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

Wachstumsrate von DTIME und DSPACE

Definition [P, NP, PSPACE]

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

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**.

Wachstumsrate von DTIME und DSPACE

Definition [P, NP, PSPACE]

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

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 $f : \mathbb{N} \rightarrow \mathbb{N}$ ist in \mathbf{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?

Beziehungen zwischen den Komplexitätsklassen

Frage:

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

$$P \subseteq NP \subseteq PSPACE$$

Beziehungen zwischen den 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?

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 (Polynomial-Zeit-Reduzibilität)

Seien L_1, L_2 Sprachen.

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

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

Reduktion

Lemma [Polynomial-Zeit-Reduktionen]

1. Sei L_1 Polynomial-Zeit-reduzibel auf L_2 ($L_1 \preceq_{\text{pol}} L_2$). Dann gilt

Wenn L_2 in **NP** ist dann ist auch L_1 in **NP**

Wenn L_2 in **P** ist dann ist auch L_1 in **P**

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