Komplexitätstheorie

WiSe 2009/10 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Komplexitätstheorie Gesamtübersicht

- Organisatorisches / Einführung
  Motivation / Erinnerung / Fragestellungen
- Diskussion verschiedener Komplexitätsklassen:
  Zeitkomplexität
  Platzkomplexität
- zugehörige Reduktionsbegriffe
- vollständige Probleme
- Anpassung von Klassenbegriffen und Reduktionen

Die Komplexitätsklassen $\mathtt{FTIME}(t)$, $\mathtt{FSPACE}(s)$, $\mathtt{DTIME}(t)$, $\mathtt{NTIME}(t)$, $\mathtt{DSPACE}(s)$ und $\mathtt{NSPACE}(s)$ sind wie folgt für Funktionen $t, s \colon \mathbb{N} \to \mathbb{N}$ definiert:

Betrachte Funktionen $f \colon W(\Sigma) \to W(\Delta)$:

- Es gilt $f \in \mathtt{FTIME}(t)$, wenn eine det. TM M und eine Konstante c mit $f = f_M$ und $T_M(x) \leq c + c \cdot t(lg(x))$ für alle $x \in W(\Sigma)$ existieren.

- Es gilt $f \in \mathtt{FSPACE}(s)$, wenn eine det. TM M und eine Konstante c mit $f = f_M$ und $S_M(x) \leq c + c \cdot s(lg(x))$ für alle $x \in W(\Sigma)$ existieren.

Komplexitätsklassen (Forts.) Betrachte Mengen $L \subseteq W(\Sigma)$:

- Es gilt $L \in \mathtt{DTIME}(t)$, wenn eine <u>det. TM M</u> und eine Konstante c mit $L = L_M$ und $T_M(x) \leq c + c \cdot t(lg(x))$ für alle $x \in L$ existieren.

- Es gilt $L \in \mathtt{DSPACE}(s)$, wenn eine <u>det. TM M</u> und eine Konstante c mit $L = L_M$ und $S_M(x) \leq c + c \cdot s(lg(x))$ für alle $x \in L$ existieren.

- Es gilt $L \in \mathtt{NTIME}(t)$, wenn eine <u>nichtdet. TM M</u> und eine Konstante c mit $L = L_M$ und $T_M(x) \leq c + c \cdot t(lg(x))$ für alle $x \in L$ existieren.

- Es gilt $L \in \mathtt{NSPACE}(s)$, wenn eine <u>nichtdet. TM M</u> und eine Konstante c mit $L = L_M$ und $S_M(x) \leq c + c \cdot s(lg(x))$ für alle $x \in L$ existieren.

Komplexitätsklassen (Forts.)

Vereinfachte Schreibweisen: statt $\mathtt{FTIME}(t)$ mit $t(n) := n^2$ einfacher $\mathtt{FTIME}(n^2)$

Beispiele der vorigen Vorlesungen:
—$f(x) := xx^r$:

$$f \in \mathtt{FTIME}(n) \cap \mathtt{FSPACE}(n)$$

Modifikation: Spiegelbild der Eingabe durch Rückwärtslauf auf Eingabeband $\Rightarrow$ kein Arbeitsband nötig $\rightsquigarrow f \in \mathtt{FSPACE}(1)$
—$L = \{xx^r \mid x \in W(\{0,1\})\}$:

$$L \in \mathtt{NTIME}(n) \cap \mathtt{NSPACE}(n)$$

—Umwandlung f_{bin} Unär- in Binärdarstellung:

$$f_{bin} \in \mathtt{FTIME}(n) \cap \mathtt{FSPACE}(\log n)$$

Kern der Vorlesung:
Beziehungen zwischen den folgenden Komplexitätsklassen

$$
\begin{aligned}
\mathbf{L} &:= \mathtt{DSPACE}(\log n) \\
\mathbf{NL} &:= \mathtt{NSPACE}(\log n) \\
\mathbf{P} &:= \bigcup_{k \in \mathbb{N}} \mathtt{DTIME}(n^k) \\
\mathbf{NP} &:= \bigcup_{k \in \mathbb{N}} \mathtt{NTIME}(n^k) \\
\mathbf{PSPACE} &:= \bigcup_{k \in \mathbb{N}} \mathtt{DSPACE}(n^k) \\
\mathbf{NPSPACE} &:= \bigcup_{k \in \mathbb{N}} \mathtt{NSPACE}(n^k) \\
\mathbf{EXP} &:= \bigcup_{k \in \mathbb{N}} \mathtt{DTIME}(2^{n^k})
\end{aligned}
$$

Folgerungen aus den Definitionen:

$$\mathbf{L} \subseteq \mathbf{NL}$$

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

$$\mathbf{PSPACE} \subseteq \mathbf{NPSPACE}$$

Bisher bekannt:

$$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE}$$

Einzige bekannte Ungleichheit:

$$\mathbf{NL} \neq \mathbf{PSPACE}$$

Offene Frage: welche Inklusionen sind ansonsten echt?

Viele Beispiele entstammen der Graphentheorie:

gerichteter Graph G: Paar (V, E)

—V endliche Menge (Ecken / Knoten, englisch: *vertex*)

—$E \subseteq V \times V$ (gerichtete Kanten / Bögen, englisch: *edge, arc*)

—Bei Kante $(x, y) \in E$: x Vorgänger von y, y Nachfolger von x

ungerichteter Graph G: Paar (V, E)

—V endliche Menge

—jedes $e \in E$ zweielementige Teilmenge von V

—Bei Kante $\{x, y\} \in E$: x und $y$sind Nachbarn

Mehr zu Graphen...

gerichtete Graphen:

—jede Kante hat Orientierung
$(x, y) \in E$ meist als $x \longrightarrow y$ dargestellt

—möglicher Sonderfall: $(x, x) \in E$ (*‘Schlinge’*)

ungerichtete Graphen:

—Kante $\{x, y\} \in E$ ohne Orientierung,
durch x—y dargestellt

—Schlingen sind nicht möglich(Menge $\{x, x\}$ hat nur *ein* Element...)

Eine Datenstruktur für Graphen

Adjazenzmatrix: ‘Datenstruktur’ zur Darstellung von Graphen $G = (V, E)$
O.B.d.A. $V = \{1, \ldots, n\}$.
Adjazenzmatrix von G ist $\underline{n \times n\text{-Matrix}}$ mit Einträgen 0 oder 1
Eintrag $m_{ij} = 1$ zeigt $(i, j) \in E$ an(bzw. $\{i, j\} \in E$ bei ungerichteten Graphen).
Darstellung der Matrix: zeilenweise, als Wort über $\{0, 1, (,)\}$:

$$(m_{11}m_{12}\ldots m_{1n})\ldots(m_{n1}m_{n2}\ldots m_{nn})$$

n Knoten $\Rightarrow$ Darstellung der Matrix hat Länge $n \cdot (n+2)$

Beispiel: Problem `CLIQUE(k)` (zu fest vorgegebenem $k \in \mathbb{N}$, also k ist nicht Teil der Eingabe) definiert als:

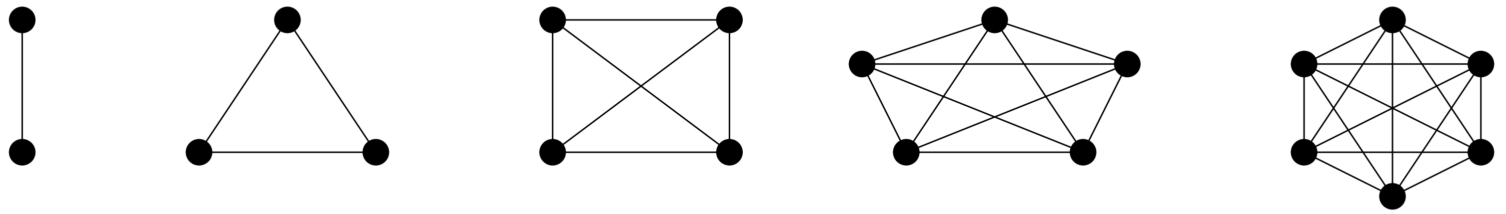
- Gegeben: ungerichteter Graph $G = (V, E)$ (durch Adjazenzmatrix)

- Frage: Gibt es in G eine Menge $B \subseteq V$ von k Knoten,
  die jeweils paarweise benachbart sind,
  d.h. wo zu $x, y \in B$ mit $x \neq y$ stets $\{x, y\} \in B$ gilt?
  (Eine solche Menge B in G heißt eine **k-Clique**.)

Problem `CLIQUE(k)` Formal exakt(er):

`CLIQUE(k)` ist Menge aller Worte über $\{0, 1, (,)\}$,
die Adjazenzmatrix eines gerichteten Graphen darstellen,
der eine k-Clique enthält.

Bsp: k-Cliquen für $k = 2, 3, 4, 5, 6$:



Clique(k) liegt in $\mathbf{L}$

Eingabe: Ein Graph mit n Knoten
$\rightsquigarrow$ Eingabe benötigt $\mathcal{O}(n^2)$ Platz
Grundidee: einen einzelnen Knoten kann man in $\log(n)$ speichern.
Wir richten k Zähler (k konstant (!)) auf Arbeitsband ein; jeder Zähler enthalte $\log(n)$ Bits.
Durchprobieren aller Zahlen-k-Tupel von 1 bis n ist in also deterministisch in $\mathcal{O}(k \log(n))$ Platz möglich und in $\mathcal{O}(n^k)$ Zeit:
Für jedes k-Tupel teste, ob die entsprechende Knotenmenge der Größe k ein k-Clique bildet.

Platzkomplexitätsklassen:
Zahl der Arbeitsbänder wegen der Konstanten in der Definition der Komplexitätsklassen ohne Belang.

Zeitkomplexitätsklassen:
Zahl der Bänder hat Einfluss auf die Komplexität von Algorithmen (ohne Beweis!), aber für polynomiale Laufzeit ist Zahl der Bänderzahl unwichtig (Zeit n^j auf k Bändern wird $c_2 \cdot n^{2j} + c_2$ auf einem Arbeitsband).

$$\Longrightarrow$$

Untersuchung von $\mathbf{P}$ und $\mathbf{NP}$ und der Platzkomplexitätsklassen unabhängig von Zahl der Arbeitsbänder!

Mit Lemma über Zusammenhang
zwischen Zeit- und Platzkomplexität:

Lemma 1

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

$$\mathbf{NL} \subseteq \mathbf{NP} \subseteq \mathbf{NPSPACE}$$

Im Wesentlichen: $T_M(x) \leq c^{S_M(x)} \leq c^{\log(lg(x))} = lg(x)^d$.
Noch fehlend:

Zusammenhang zwischen Nichtdeterminismus und Determinismus

Hauptresultat:
Platzkomplexität bei der det. Simulation von nichtdet. Maschinen?

Hauptresultat:

Satz 2 **Savitch, 1970** *Sei* $s\colon \mathbb{N} \to \mathbb{N}$ *mit* $\log \in \mathcal{O}(s)$ *gegeben. Dann gilt*

$$\mathtt{NSPACE}(s) \subseteq \mathtt{DSPACE}(s^2)$$

Satz von Savitch — Beweis

Wir können uns auf nichtdeterministische 3-Band-TM M beschränken, wobei (da Sprachakzeptanz) das Ausgabeband unbenutzt bleibt. Konfigurationen werden daher durch

$$(z, \mathrm{bin}(i), w_\ell, a, w_r)$$

beschrieben; $\mathrm{bin}(i)$ zeigt binär kodiert die Kopfposition auf dem Eingabeband.
Beobachte: Ist $\lg(w_\ell a w_r) \in \Theta(\log(n))$, wobei n die Eingabelänge ist, so lässt sich jede "sinnvolle" Konfiguration von M mit $\Theta(\log(n))$ Bits beschreiben.
Ziel: Deterministischer Algorithmus, der mit "geringem Platzaufwand" feststellt, ob für zwei Konfigurationen K, K' von M gilt: $K \xrightarrow[M]{*} K'$.
Boolesche Hilfsfunktion: TEST(K, K', i): überprüft, ob $K \xrightarrow[M]{n} K'$ für $n \leq 2^i$ gilt.
Wir zeigen:
(1) TEST benötigt für alle sinnvollen Eingaben logarithmischen Platz für die "eigene Arbeit".
(2) TEST arbeitet rekursiv und der Rekursionsstack (auf dem wegen (1) höchsten Objekte logarithmischer Größe liegen) ist höchstens logarithmisch tief.

Die Prozedur TEST

Für $i = 0$ liefert TEST(K, K', i) wahr gdw. $K = K'$ oder $K \xrightarrow[M]{1} K'$.

Für $i > 0$ sucht TEST unter allen Konfigurationen K'' mit $S_M(K'') \le S_M(K')$ [bzgl. Eingabe x] (rekursiv) eine Konfiguration, für die gilt:

$$\text{TEST}(K, K'', i-1) \text{ UND } \text{TEST}(K'', K', i-1).$$

Die Rekursionen bauen einen Stack auf, der ebenfalls gespeichert werden muss.
Das Hauptprogramm zählt nun i von 1 an immer höher und sucht dann unter allen Endkonfigurationen K' (zur Eingabe x) mit $S_M(K') \le i$ eine solche, die TEST$(I_M(x), K', i)$ wahr macht; im positiven Fall hält die simulierende Maschine mit $i = T_M(x) \le c^{S_M(x)}$.

Die Zähler für K' bzw. i benötigen daher binär höchstens $\mathcal{O}(S_M(x))$ Platz.
Diese Werte sind auch für jeden Rekursionsaufruf zu speichern.
$\leadsto \mathcal{O}(S_M(x)^2)$ Platzbedarf, da $S_M(x)$ mindestens wie $\log(|x|)$ wächst.

Satz von Savitch — Folgerungen

Folgerung 3

$$\mathbf{NL} = \mathtt{NSPACE}(\log n) \subseteq \mathtt{DSPACE}(\log^2 n) \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE}$$

—Offene Frage: Verhältnis $\mathtt{DSPACE}(\log^2 n)$ zu $\mathbf{P}$ und $\mathbf{NP}$

—Inklusion $\mathtt{NSPACE}(\log n) \subseteq \mathbf{P}$ erst später!

Folgerung aus dem Satz von Savitch für die Rekursionstheorie:
Nichtdet. TM sind bei Akzeptieren grundsätzlich **(ohne genaue Komplexitätsbetrachtungen)** nicht mächtiger sind als det. TM !

Folgerung 4
Zu jeder n.det. TM M gibt es eine det. TM M' mit $L_M = L'_M$.

Beweis: wähle

$$s(n) := \max\{\log_n, S_M(x) \mid x \in L_M, lg(x) = n\}$$

dann sicherlich $L_M \in \mathtt{NSPACE}(s) \subseteq \mathtt{DSPACE}(s^2)$.

Reduktionen—Motivation

Oft: Exakte Komplexität eines Problems unbekannt

Gesucht: (grobe) Einordnung durch Vergleich mit anderen Problemen

Grundidee: Transformation zwischen verschiedenen Problemen:

Gegeben: Probleme $A \subseteq W(\Sigma)$, $B \subseteq W(\Delta)$

Funktion $f\colon W(\Sigma) \to W(\Delta)$ *transformiert* A in B,
wenn für alle $x \in W(\Sigma)$ gilt

$$x \in A \iff f(x) \in B$$

Many-one: die Idee

Untersuchung von $x \in A$ durch:
(1) Bestimmung von $f(x)$
(2) Untersuchung der Frage $f(x) \in B$

Offensichtlich: Falls
(a) Problem B leicht lösbar und
(b) f nicht zu kompliziert auszurechnen
$\Rightarrow$ A leicht lösbar

Andere Formulierung:
Gibt es einfache Transformation f von A in B,
so ist Problem A nicht (viel) schwerer zu lösen als Problem B.

Man hat die Aufgabe darauf **reduziert**, (nur) noch B zu lösen...

$\rightsquigarrow$ Transformation ermöglicht An-/Ein-Ordnung von Problemen nach Komplexität.

Zwei konkrete (many-one) Reduktionen

Gegeben $A \subseteq W(\Sigma)$, $B \subseteq W(\Delta)$

- A heißt *Polynomzeit-reduzierbar* auf B (in Zeichen $A \leq_p B$), wenn für ein $k \in \mathbb{N}$ eine Funktion $f(n) \in \mathtt{FTIME}(n^k)$ mit

$$\forall x \in W(\Sigma)\,(x \in A \Leftrightarrow f(x) \in B)$$

  existiert.

- A heißt *mit logarithmischem Platz reduzierbar* oder *logspace-reduzierbar* auf B (in Zeichen $A \leq_{\log} B$), wenn eine Funktion $f \in \mathtt{FSPACE}(\log n)$ mit

$$\forall x \in W(\Sigma)\,(x \in A \Leftrightarrow f(x) \in B)$$

  existiert.

Beispiel: $\Sigma = \Delta = \{0, 1\}$

$A = \{x \in W(\Sigma) \mid x = x^r\} \qquad B = \{xx \mid x \in W(\Sigma)\}$

Betrachte f mit $f(x) := xx^r$
$\Rightarrow f \in \mathtt{FSPACE}(\log n)$ (sogar $f \in \mathtt{FSPACE}(1)$).

Dann:

$$x \in A \Leftrightarrow x = x^r \Leftrightarrow xx^r = xx \Leftrightarrow xx^r = f(x) \in B$$

also $A \leq_{\log} B$.

Beispiel: Problem `CLIQUE(k)` (zu fest vorgegebenem $k \in \mathbb{N}$, also k ist nicht Teil der Eingabe) definiert als:

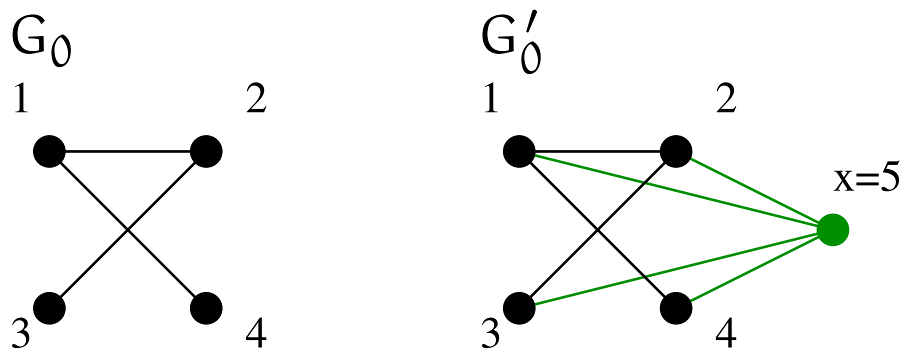
- Gegeben: ungerichteter Graph $G = (V, E)$ (durch Adjazenzmatrix)

- Frage: Gibt es in G eine Menge $B \subseteq V$ von k Knoten,
die jeweils paarweise benachbart sind,
d.h. wo zu $x, y \in B$ mit $x \neq y$ stets $\{x, y\} \in B$ gilt?
(Eine solche Menge B in G heißt eine **k-Clique**.)

Beispiel: für alle $k \in \mathbb{N}$ gilt

$$\texttt{CLIQUE}\,(k) \leq_{\log} \texttt{CLIQUE}\,(k+1)$$

Konstruktion:
Gegeben $G = (V, E)$ ungerichteter Graph, setze $G' := (V', E')$
mit $V' = V \dot\cup \{x\}$ und $E' = E \cup \big\{\{x, y\} \mid y \in V\big\}$
für einen *neuen* Knoten $x \notin V$, etwa:



Beispiel (Forts.): für alle $k \in \mathbb{N}$ gilt

$$\texttt{CLIQUE}\,(k) \leq_{\log} \texttt{CLIQUE}\,(k+1)$$

$\{i_1, \cdots i_k\}$ k-Clique in $G \Rightarrow \{i_1, \cdots, i_k, x\}$ k+1-Clique in G'

Andererseits:
$\{i_1, \cdots, i_{k+1}\}$ k+1-Clique in $G' \Rightarrow B$ k-Clique in G mit

$$B := \begin{cases} \{i_1, \cdots, i_{k+1}\} \setminus \{x\}, & \text{falls } x \in \{i_1, \cdots, i_{k+1}\}, \\ \{i_1, \cdots, i_k\} & \text{sonst} \end{cases}$$

Also: G hat k-Clique $\Leftrightarrow G'$ hat k+1-Clique.

Beispiel (Forts.): für alle $k \in \mathbb{N}$ gilt

$$\texttt{CLIQUE}\ (k) \leq_{\log} \texttt{CLIQUE}\ (k+1)$$

Komplexität der Konstruktion von G' aus G:

Transformation auf Adjazenzmatrizen über $\Sigma = \{0, 1, (,)\}$

Dazu $f : W(\Sigma) \to W(\Sigma)$ wie folgt:

—Für Argument w ist $f(w)$ i.W. Kopie von w, jedoch:
—Überall in w wird '$)$' durch '$1)$' ersetzt.
—An das Ende wird '$(\underbrace{11 \cdots 1}_{n} 0)$' angefügt,

mit $n =$ Abstand zwischen ersten '$($' und erstem '$)$' in w
(bzw. n=0 falls kein Klammerpaar in w existiert).

Bsp. (Forts.):

Adjazenzmatrix von G_0:

$$w_0 = (0101)(1010)(0100)(1000)$$

Adjazenzmatrix von G_0':

$$f(w_0) = (01011)(10101)(01001)(10001)(11110)$$

Damit $w \in \texttt{CLIQUE}(k) \Longleftrightarrow f(w) \in \texttt{CLIQUE}(k+1)$

Da $f \in$ FSPACE $(\log n)$ (wir benötigen logarithmischen Platz zur Generierung der Darstellung der letzten Zeile der Adjazenzmatrix), gilt

$$\texttt{CLIQUE}(k) \leq_{\log} \texttt{CLIQUE}(k+1)$$