Theoretische Grundlagen der Informatik
Tutorium XII & XIII (SR 301)
Tut Nr. 9 – Wiederholung, Komplexitätsklassen

David Münch

Karlsruher Institut für Technologie
Institut für Informatik
IKS Müller-Quade

8. Januar 2010

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Organisatorisches

Email: muenchdavid@gmail.com

`https://www.stud.uni-karlsruhe.de/~uhbro/`

Tutorium 12: Donnerstags 8:00 Uhr - Raum 301
Tutorium 13: Donnerstags 9:45 Uhr - Raum 301

Übungsblattabgabe Mittwochs 12:00 Uhr.

Organisatorisches

Deckblatt benutzen: `http://www.stud.uni-karlsruhe.de/~unbdh/deckblatt/index.php?course=6`

Gruppenarbeit erwünscht, aber jeder muss handschriftliche Lösung mit Namen aller Gruppenteilnehmer abgeben.

50% der Punkte sind notwendig für den Schein.

Organisatorisches

Nicht abgeholte Übungsblätter können ab sofort bei Nico Döttling R274 abgeholt werden.
`https://puck.iaks.uka.de/eiss/?id=167`

Wichtig!

Es gibt zum neuen Thema ein vorlesungsbegleitendes Skriptum unter: `https://puck.iaks.uka.de/eiss/fileadmin/User/Info_3/skript.pdf`.

Was wollen wir heute erreichen?

Was wollen wir heute erreichen?

- poly. many-one Reduktionen durchfüren können

Was wollen wir heute erreichen?

- poly. many-one Reduktionen durchfüren können
- Wiederholung Komplexitätstheorie

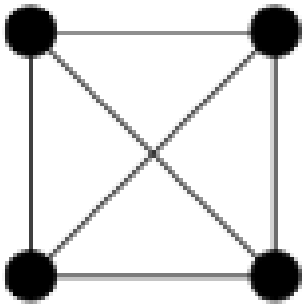
Aufgabe 1

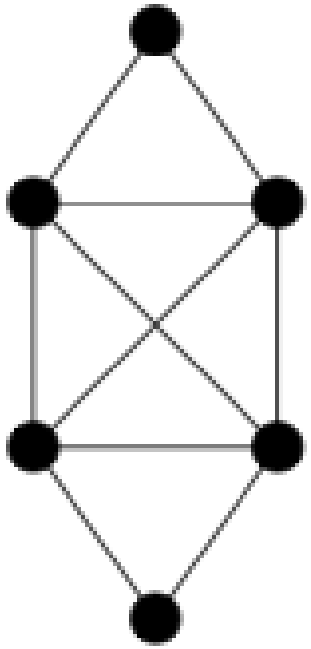
Wir definieren folgendes Problem EC
Gegeben: Ein gerichteter zusammenhängender Graph $G = (V, E)$
Frage: Gibt es eine Folge von Knoten $v_1, \ldots, v_k \in V$, mit der Eigenschaft, dass

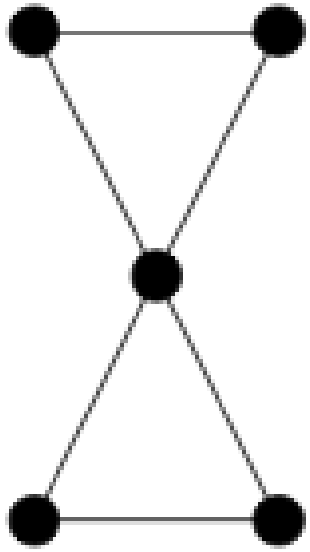
- $v_1 = v_k$
- Für alle $e \in E$ gibt es ein $i \in \{1, \ldots, k\}$ sodass $e = (v_i, v_{i+1})$?

Zeigen Sie: $EC \in \mathcal{P}$.

Das Problem EC ist in der Literatur unter dem Namen Eulerkreis bekannt. Welche der folgenden Graphen haben einen Eulerkreis?







Lösung:
Wir haben hier das gerichtete Eulerkreisproblem.

Lösung:
Wir haben hier das gerichtete Eulerkreisproblem.
Ein Graph G ist eulersch $\Leftrightarrow$ in jedem Knoten stimmen die Anzahl der eingehenden mit der Anzahl der ausgehenden Kanten überein und G ist stark zusammenhängend. Kurzer Beweis oder Beweisidee angeben!

Lösung:
Wir haben hier das gerichtete Eulerkreisproblem.
Ein Graph G ist eulersch $\Leftrightarrow$ in jedem Knoten stimmen die Anzahl der eingehenden mit der Anzahl der ausgehenden Kanten überein und G ist stark zusammenhängend. Kurzer Beweis oder Beweisidee angeben!
Zur Bestimmung des Aufwandes gibt man beispielsweise einen Algorithmus an. Die o.g. Bedingung kann durch ein einfaches Durchgehen aller Knoten überprüft werden. Somit liegt der Aufwand in $\mathcal{O}(|Knoten| \cdot |Kanten|)$

Aufgabe 2

a) Sei $\mathcal{L} \in NP$. $\mathcal{L}$ sei über dem Alphabet $\Sigma = \Gamma = \{0,1\}^*$ definiert.
Zeigen Sie: Es gibt ein Polynom $p(n)$ und eine deterministische Turingmaschine $\mathcal{T}$ sodass $\mathcal{T}$ die Sprache $\mathcal{L}$ in TIME$(O(2^{p(n)})$ erkennt.

b) Zeigen Sie: Ist $\mathcal{L} \in NP$, dann ist auch $\mathcal{L}^* \in NP$.

Lösung:
Da $\mathcal{L} \in NP$, gibt es eine nichtdeterministische Turingmaschine $\mathcal{M}$, die $\mathcal{L}$ in NTIME$(q(n))$ für ein Polynom $q(n)$ erkennt. Eine Trace von $\mathcal{M}$ läßt sich also mit maximal $\mathcal{O}(q(n)^2)$ Zeichen darstellen. Dies gilt, da $\mathcal{M}$ für jede Eingabe w nach spätestens $q(n)$ Schritten hält und damit auch nie mehr als $q(n)$ Bandstellen beschreiben kann. Wir können eine Trace also als Abfolge von $q(n)$ Konfigurationen von $\mathcal{M}$ codieren, sodass jede Konfiguration eine Größe von höchstens $\mathcal{O}(q(n))$ Bits hat. Maschine $\mathcal{T}$ arbeitet nun folgendermaßen: Sie erzeugt alle möglichen Traces von $\mathcal{M}$ und prüft ob eine davon $\mathcal{M}$ akzeptieren lässt. Wir konstruieren $\mathcal{T}$ folgendermaßen:

$\mathcal{T}$ bei Eingabe w:

- Zähle alle möglichen Traces $t = c_0 \# c_1 \# \ldots \# c_{q(n)}$ der Länge $c \cdot q(n)^2$ auf. (c ist eine Konstante die von der Codierung der Traces abhängt)
  - Prüfe ob c_0 eine korrekte Startkonfiguration codiert, also ob $\mathcal{M}$ sich im Anfangszustand befindet und w auf dem Band steht. Falls nicht fahre mit der Zählschleife fort.
  - Prüfe für alle $i = 0, \ldots, q(n)$ ob c_{i+1} eine zulässige Nachfolgekonfiguration von c_i ist, also ob es einen nichtdeterministischen Zustandsübergang gibt der c_i in c_{i+1} überführt. Falls nicht setze Zählschleife fort.
  - Prüfe, ob $c_{q(n)}$ sich in einem akzeptierenden Zustand befindet. Falls ja, akzeptiere. Falls nein, setze Zählschleife fort.
- Lehne die Eingabe ab.

Es ist offensichtlich, dass $\mathcal{T}$ die selbe Sprache wie $\mathcal{M}$ akzeptiert, denn wenn es eine akzeptierende Trace von $\mathcal{M}$ gibt, dann findet $\mathcal{T}$ diese. Es gibt maximal $\mathcal{O}(2^{c \cdot q(n)^2})$ Traces die von der Zählschleife aufgezählt werden. Zur Überprüfung einer Trace werden je zwei Konfigurationen verglichen. Es ergibt sich also eine Aufwandsabschätzung von $\mathcal{O}(q(n)^3)$. Insgesamt also hat der Algorithmus $\mathcal{T}$ einen Aufwand von $\mathcal{O}(q(n)^3 2^{c \cdot q(n)^2}) \subseteq \mathcal{O}(2^{c \cdot q(n)^2 + n})$. Wir setzen $p(n) = c \cdot q(n)^2 + n$, welches wieder ein Polynom ist, und erhalten die gewünschte Laufzeitabschätzung.

Lösung:

Sei $\mathcal{L} \in NP$, $R_{\mathcal{L}}$ die zugehörige Zeugenrelation, so definieren wir

$$R_{\mathcal{L}^*} = \{x, ((x_1, w_1), \ldots, (x_k, w_k)) | x = x_1 x_2 \ldots x_k \\ \text{und } \forall i = 1, \ldots, k : (x_i, w_i) \in R_{\mathcal{L}}\}.$$

$R_{\mathcal{L}^*}$ ist polynomiell entscheidbar, denn die Länge der Zeugen $((x_1, w_1), \ldots, (x_k, w_k))$ ist polynomiell beschränkt in der Länge von x (höchstens $|x|$ Paare (x_i, w_i)). Es gilt

$$\begin{aligned} x \in \mathcal{L}^* &\Leftrightarrow \exists x_1, \ldots, x_k \in \mathcal{L} : x = x_1 \ldots x_k \\ &\Leftrightarrow \exists x_1, \ldots, x_k \exists w_1, \ldots, w_k : x = x_1 \ldots x_k \\ &\quad \text{und } \forall i = 1, \ldots, k : (x_i, w_i) \in R_{\mathcal{L}} \\ &\Leftrightarrow \exists (x_1, w_i), \ldots, (x_k, w_k) : (x, ((x_1, w_i), \ldots, (x_k, w_k))) \in R_{\mathcal{L}^*} \end{aligned}$$

Wiederholung Komplexitätstheorie

Sei M eine TM mit Eingabealphabet A. Die Rechenzeit von M wird durch die Funktion $t_M : A^* \to \mathbb{N}_0$ beschrieben.

$$t_M(\alpha) = \begin{cases} \text{minimale Zahl von Rechenschritten,} & \text{falls } \alpha \in L(M) \\ \text{die } M \text{ braucht, um } \alpha \text{ zu akzeptieren,} & \\ 0, & \text{falls } \alpha \notin L(M). \end{cases}$$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

NTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine nichtdeterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

NTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine nichtdeterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

P $= \bigcup_{k=1}^{\infty} DTIME(n^k)$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

NTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine nichtdeterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

P $= \bigcup_{k=1}^{\infty} DTIME(n^k)$

NP $= \bigcup_{k=1}^{\infty} NTIME(n^k)$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

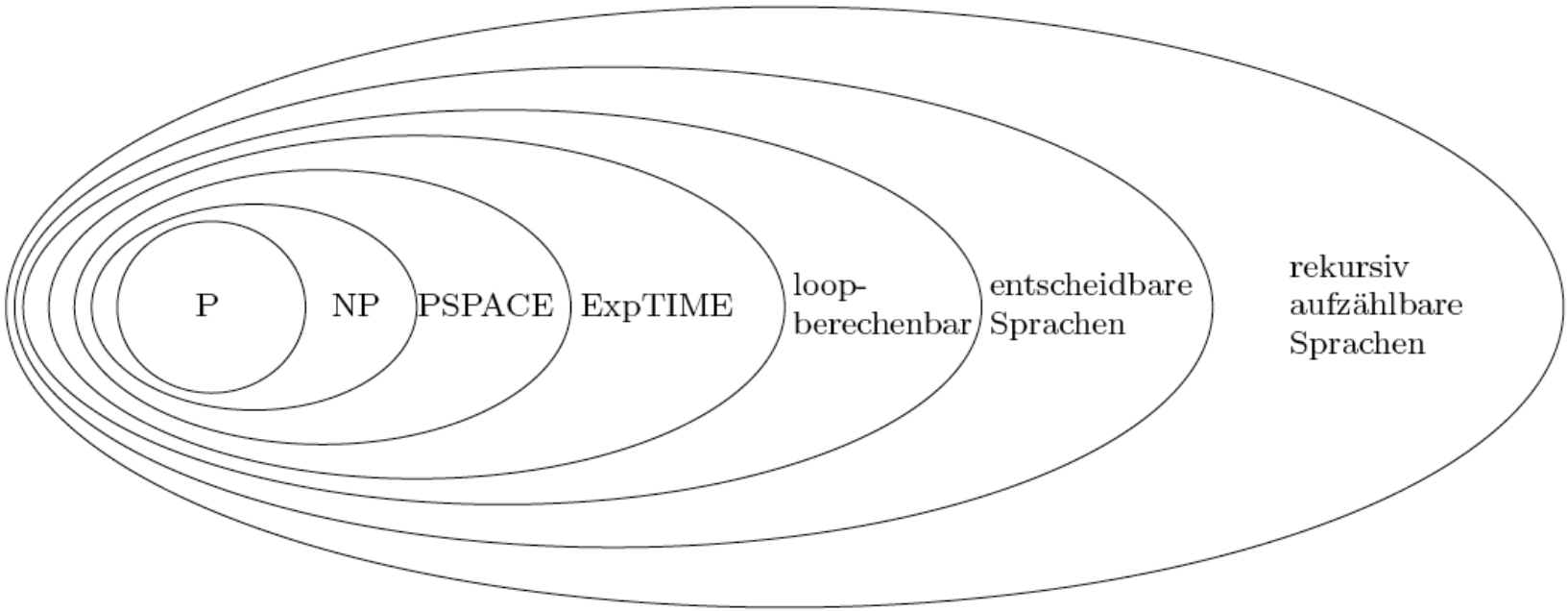
NTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine nichtdeterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

P $= \bigcup_{k=1}^{\infty} DTIME(n^k)$

NP $= \bigcup_{k=1}^{\infty} NTIME(n^k)$

EXPTIME $= \bigcup_{k=1}^{\infty} DTIME(2^n)$

Zentrale Frage: $\mathcal{P} = \mathcal{NP}$?


P
NP
PSPACE
ExpTIME
loop-
berechenbar
entscheidbare
Sprachen
rekursiv
aufzählbare
Sprachen

$\mathcal{NP}$-vollständig

Definition: $\mathcal{NP}$-vollständig

Eine Sprache L heißt $\mathcal{NP}$-**vollständig**, falls gilt:

1. $L \in \mathcal{NP}$ und
2. für alle $L' \in \mathcal{NP}$ gilt: $L' \leq_p L$

Hinweis:
$A \leq_p B$: A ist in Polynomialzeit many-one reduzierbar auf B.
$w \in A \Leftrightarrow f(w) \in B$ mit f in Polynomialzeit berechenbare Abbildung.

Problem: PARTITION:

Gegeben: Natürliche Zahlen $a_1, ..., a_n \in \mathbb{N}$ $(n \in \mathbb{N})$

Gesucht: Gibt es eine Teilmenge $J \subseteq \{1, ..., n\}$ mit $\sum_{1 \leq i \leq n, i \in J} a_i = \sum_{1 \leq i \leq n, i \notin J} a_i$?

Problem: BIN PACKING:

Gegeben: Eine Behältergröße $b \in \mathbb{N}$, die Anzahl der Behälter $k \in \mathbb{N}$ und Objekte $a_1, ..., a_n$ $(n \in \mathbb{N})$ mit $a_i \in \mathbb{N}, a_i \leq b$ für alle $i \in \{1, ...,$

Gesucht: Können die n Objekte so auf die k Behälter verteilt werden, dass kein Behälter überbeladen ist?
(Das heißt: Existiert eine Abbildung $f : \{1, ..., n\} \to \{1, ..., k\}$, sodass für alle $j \in \{1, ..., k\}$ gilt: $\sum_{1 \leq i \leq k, f(i) = j} a_i \leq b$?)

Aufgabe

a) Zeigen Sie, dass BIN PACKING $\mathcal{NP}$-hart ist, wobei PARTITION als $\mathcal{NP}$-vollständig vorausgesetzt werden darf!

b) Gegeben seien die Objekte der PARTITION-Probleminstanz $(a_1, a_2, a_3, a_4, a_5, a_6) = (1, 1, 2, 3, 4, 5)$. Zeigen oder widerlegen Sie, ob das transformierte und das ursprüngliche Problem eine Lösung besitzen!

Problem: 4-COLOR
Gegeben: Ein ungerichteter Graph $G = (V, E)$
Gesucht: Gibt es eine Färbung der Knoten V, sodass je zwei durch eine Kante aus E miteinander verbundene Knoten unterschiedlich gefärbt sind, wenn nur vier unterschiedliche Farben zur Verfügung stehen?

Aufgabe

Zeigen Sie, dass 4-COLOR $\mathcal{NP}$-vollständig ist!

Hinweis: Es kann hilfreich sein, wenn Sie die $\mathcal{NP}$-Vollständigkeit des Dreifärbbarkeitsproblems 3-COLOR verwenden.

Es ist bekannt, dass sowohl das Erfüllbarkeitsproblem der Aussagenlogik SAT als auch 3-SAT, das Erfüllbarkeitsproblem mit Beschränkung auf Klauseln mit nur 3 Literalen, $\mathcal{NP}$-vollständig sind.
Das Problem 3-CLIQUE ist wie folgt definiert:

Problem: 3-CLIQUE
Gegeben: Ein ungerichteter Graph $G = (V, E)$
Gesucht: Gibt es eine Clique (vollständig verbundener Teilgraph) der Größe 3 in G?

Aufgabe

Zeigen Sie, dass 3-CLIQUE **nicht** $\mathcal{NP}$-vollständig ist!

Aufgabe

Beweisen Sie folgende Aussagen:

a) Die Klasse $\mathcal{NP}$ ist unter Schnittbildung abgeschlossen.

b) Die Klasse $\mathcal{NP}$ ist unter Vereinigung abgeschlossen.

Dabei nehmen wir an, dass alle Sprachen über dem binären Alphabet $\Sigma = \{0, 1\}$ definiert sind.

Reflexion

Was haben wir heute gelernt?

Reflexion

Was haben wir heute gelernt?

- Polynomiale Transformierbarkeit kennen gelernt.

Noch Fragen?

Vorschau

Vorschau

- weitere Komplexitätsklassen

Bis zum nächsten Mal