Prof. Dr. E. Wanke, Y. Schmitz Düsseldorf, 12.04.2021

Übung zur Vorlesung

Algorithmische Komplexitätstheorie

Die Übungen werden dienstags und mittwochs jeweils von 10:30 - 12:00 Uhr online stattfinden. In den Übungen werden die Aufgaben besprochen (wo gab es Probleme, welche Ansätze haben (nicht) funktioniert, etc.), die Aufgaben werden jedoch **nicht** vorgerechnet. Es wird empfohlen die Aufgaben als Vorbereitung auf die Übungen schriftlich zu bearbeiten.

Übungstermin	Aufgaben
20.04. & 21.04.	1,2
27.04. & 28.04.	3,4
04.05. & 05.05.	5,6
11.05. & 12.05.	7,8
18.05. & 19.05.	9,10
25.05. & 26.05.	11
01.06. & 02.06.	12,13
08.06. & 09.06.	Fragestunde
15.06. & 16.06.	14,15
22.06. & 23.06.	16
29.06. & 30.06.	17,18
06.07. & 07.07.	19,20
13.07. & 14.07.	21
20.07. & 21.07.	Fragestunde

Aufgabe 1:

Gegeben sei die Turingmaschine $M = (\{s, q_1, q_2, q_3\}, \{a, X, \triangleright, \#\}, \delta, s)$ mit folgendem Programm δ:

δ	a	X	$\#$	$\triangleright$
s	$(q_1, X, \rightarrow)$	$(s, X, \rightarrow)$	$(q_2, \#, \leftarrow)$	$(s, \triangleright, \rightarrow)$
q_1	$(s, a, \rightarrow)$	$(q_1, X, \rightarrow)$	$(q_3, \#, \leftarrow)$	$(q_1, \triangleright, \perp)$
q_2	$(q_2, a, \leftarrow)$	$(q_2, X, \leftarrow)$	$(q_2, \#, \perp)$	$(s, \triangleright, \rightarrow)$
q_3	$(q_3, a, \perp)$	$(q_3, X, \leftarrow)$	$(q_3, \#, \perp)$	$(h, \triangleright, \perp)$

(a) Geben Sie die Berechnung von M auf der Eingabe $w_1 = aaaa$ an.

(b) Geben Sie die Sprache L an, die M akzeptiert.

(c) Zeigen Sie, dass $w_2 = aaaaa$ nicht von M akzeptiert wird.

Aufgabe 2: Gegeben sei die Sprache $L = \{(ab)^n c^n \mid n \geq 0\} \subseteq \{a, b, c\}^*$.

(a) Geben Sie eine deterministische Turingmaschine M an, die L akzeptiert.

(b) Geben Sie eine Berechnung von M auf der Eingabe $w = ababcc$ an.

Aufgabe 3:

Definieren Sie die Relation $\overset{M}{\longmapsto}$ für deterministische k-Band-Turingmaschinen.

Aufgabe 4:

Seien $M_1 = (Q_1, \Sigma, \delta_1, s_1)$ und $M_2 = (Q_2, \Sigma, \delta_2, s_2)$ zwei deterministische 1-Band-Turingmaschinen. Geben Sie eine deterministische 2-Band-Turingmaschine M an, sodass $L(M) = L(M_1) \cup L(M_2)$. Dabei sind $L(M), L(M_1), L(M_2)$ die Sprachen, die von den entsprechenden Turingmaschinen akzeptiert werden.

Aufgabe 5:

Zeigen Sie analog zum Beweis des Speed-Up Theorems:
Sei $L \in \text{SPACE}(f(n))$. Für jedes $\epsilon > 0$ ist $L \in \text{SPACE}(\epsilon \cdot f(n) + \mathcal{O}(1))$.

Aufgabe 6:

Gegeben sei die Sprache $L = \{a^n b^m c^n \mid n, m \geq 0\} \subseteq \{a, b, c\}^*$.

(a) Geben Sie eine deterministische 2-Band-Turingmaschine M an, die L in einer minimalen Anzahl an Schritten akzeptiert.

(b) Geben Sie eine möglichst gute, obere Zeitschranke $f : \mathbb{N}_0 \to \mathbb{N}_0$ für M an.

Aufgabe 7:

Geben Sie eine nichtdeterministische 2-Band-Turingmaschine $M = (Q, \Sigma, \Delta, s)$ an, die für eine Eingabe $u \in \Sigma^*$ eine Binärzahl der Länge $|u|$ auf das zweite Band schreibt. Dabei soll es für jede mögliche Binärzahl der Länge $|u|$ einen Berechnungspfad in M geben, d.h. es soll gelten:

$$\forall u \in \Sigma^* \; \forall n \in \{0, \ldots, 2^{|u|} - 1\} : (s, \triangleright, u, \triangleright, \epsilon) \overset{M^*}{\longmapsto} (h, \triangleright u, \epsilon, \triangleright n, \epsilon)$$

Aufgabe 8:

Sei Σ eine Alphabet und seien $u, v \in \Sigma^*$. Die *Teilwortrelation* auf Σ^* ist definiert durch

$$u \sqsubseteq v \Leftrightarrow \exists v_1, v_2 \in \Sigma^* : \; v_1 u v_2 = v.$$

Das Wort u wird als *Teilwort* von v bezeichnet.

(a) Sei $L_1 := \{w \mid 0110 \sqsubseteq w\}$ eine Sprache über dem Alphabet $\{0, 1\}$. Geben Sie eine nichtdeterministische k-Band-Turingmaschine M_1 für ein k Ihrer Wahl an, die L_1 entscheidet.

(b) Sei $L_2 := \{u2v \mid u, v \in \{0, 1\}^*, u \sqsubseteq v\}$ eine Sprache über dem Alphabet $\{0, 1, 2\}$. Geben Sie eine nichtdeterministische k-Band-Turingmaschine M_2 für ein k Ihrer Wahl an, die L_2 entscheidet.

Aufgabe 9:

Sei $x \in \{0\}\{0,1\}^*\{0\}$. Ein *Einserblock* in x ist ein Wort $y \in \{1\}^*$ mit $0y0 \sqsubseteq x$. Sei M_x die Multimenge der Einserblöcke in einem Wort x und sei $f(M_x) = \sum_{x \in M_x} |x|$ die Anzahl der Einsen in der Menge M_x.
Beispiel: Sei $x = 011101110110110101010$, dann ist $M_x = \{111, 111, 11, 11, 1, 1, 1\}$ und $f(M_x) = 13$.

Sei $L = \{x \in \{0\}\{0,1\}^*\{0\} \mid \exists M'_x \subseteq M_x : 2 \cdot f(M'_x) = f(M_x)\}$, d.h. es existiert eine Teilmenge der Einserblöcke, die genau die Hälfte der Einsen in x enthält. Geben Sie eine nichtdeterministische k-Band-Turingmaschine M für ein k Ihrer Wahl an, die L entscheidet.

Aufgabe 10:

Sei A die Menge der natürlichen Zahlen, die keine Primzahlen sind. Beschreiben Sie eine mögliche Arbeitsweise einer nichtdeterministischen Turingmaschine M, die A entscheidet. Sie brauchen dabei das Programm von M nicht explizit anzugeben.

Aufgabe 11: Betrachten Sie die folgende nichtdeterministische Turingmaschine:

$M = (Q, \Sigma, \Delta, s)$ mit $Q = \{s\}$, $\Sigma = \{0, \#, \triangleright\}$ und
$\Delta = \{(s, \triangleright, s, \triangleright, \rightarrow), (s, 0, s, 0, \rightarrow), (s, \#, h_y, \#, \leftarrow), (s, \#, h_n, \#, \bot)\}$. Betrachten Sie die SAT-Instanz aus dem Beweis von Cooks Theorem.

(a) Geben Sie die Menge C_M für M an. (Es wird empfohlen die 4-Tupel zusammenzufassen.)

(b) Sei $x = 00$ eine Eingabe für M. Wie viele Variablen enthält X? Wie viele Klauseln enthält $\mathcal{C}$ vor Schritt 5 des Beweises (d.h. ohne die Klauseln die mithilfe der Menge C_M gebildet werden)?

Lösungsvorschlag für (a):
Die Menge C_M enthält 4-Tupel der Form (a, b, c, d), wenn in der Konfiguration β_i a, b, c drei aufeinanderfolgende Symbole sind und in der Konfiguration β_{i+1} das Symbol d auf der gleichen Position stehen kann wie b in der Konfiguration β_i. D.h. wenn a, b, c drei Symbole in einem Schritt i sind, soll im nächsten Schritt $i + 1$ das Symbol d statt dem Symbol b stehen dürfen. (Anschaulich gesprochen: Wenn auf dem Band a, b, c steht, wird das b im nächsten Schritt zu d.)
Zur vereinfachten Notation sei im Folgenden $\Sigma' = \Sigma \cup \{\#'\}$ (also alle Symbole, die keine Programmschritte sind) und sei $\Sigma'' = \Sigma' \cup \{(s, \#, h_y, \#, \leftarrow), (s, \#, h_n, \#, \bot)\}$.

Als erstes ist zu beobachten, dass die Symbole "auf dem Band" nur durch den Kopf verändert werden können. D.h. wenn a, b, c keine Programmschritte sind ($a, b, c \in \Sigma'$), wird b nicht verändert. Folglich sind alle 4-Tupel der Form (a, b, c, b) mit $a, b, c \in \Sigma'$ in C_M.
Wir betrachten nun, wann ein Symbol aus $\overline{\Sigma}$ "auf dem Band" stehen kann, bzw. welche Zeichen in dem Schritt davor "auf dem Band" gestanden haben müssen. D.h. wir suchen alle 4-Tupel die "erlaubt", also in C_M sind.
Zuerst betrachten wir die 4-Tupel der Form $(a, b, c, \#')$. Das Trennsymbol $\#'$ steht nur "am Anfang jeder Zeile", d.h. es steht nur an Positionen, an denen im Schritt zuvor auch das Trennsymbol stand. Demnach kann das d nur dann das Trennsymbol sein, wenn b auch eines ist. Außerdem läuft der Kopf niemals auf oder über das Trennsymbol, d.h. a und c können keine Programmschritte sein, durch die der Kopf im nächsten Schritt auf dem Trennsymbol stehen würde. Die Tupel mit $b = \#'$ werden allerdings laut Skript nicht in C_M aufgenommen.
Als zweites betrachten wir die 4-Tupel der Form $(a, b, c, \triangleright)$. Das initiale Symbol steht immer nur am Anfang des Bandes und wird nicht überschrieben, d.h. im vorherigen Schritt steht auch das initiale Symbol. Außerdem steht links neben dem initialen Symbol immer das Trennsymbol, d.h. $a = \#'$. Die hier betrachtete Maschine hat zudem nur einen Programmschritt, bei dem sich der Kopf nach links bewegt. Allerdings erreicht die Maschine durch diesen Programmschritt einen Haltezustand und da die Berechnung endet sobald ein Haltezustand erreicht wurde, kann das c in allen 4-Tupeln beliebig gewählt werden. Beliebig soll hier und im Folgenden bedeuten, dass ein beliebiges Zeichen aus $\overline{\Sigma}$ gewählt werden kann, solange ein für die Maschine sinnvolles Tupel erzeugt wird, also z.B. a, b, c enthalten nicht zwei Programmschritte, zwei Trennsymbole oder zwei initiale Symbole. Folglich sind alle Tupel der Form $(\#', \triangleright, c, \triangleright)$ mit c beliebig in C_M enthalten. Zusätzlich gilt, dass wenn der Programmschritt $(s, \triangleright, s, \triangleright, \rightarrow)$ ausgeführt wird, im nächsten Schritt das initiale Symbol "auf dem Band" steht. D.h. alle Tupel der Form $(\#', (s, \triangleright, s, \triangleright, \rightarrow), c, \triangleright)$ mit c beliebig sind ebenfalls in C_M enthalten.
Als nächstes betrachten wir die 4-Tupel der Form $(a, b, c, \#)$. Aus dem Programm der Maschine geht hervor, dass die Leerzeichen nicht überschrieben werden und auch keine neuen Leerzeichen geschrieben werden. D.h. das d kann nur ein Leerzeichen sein, wenn das b auch eines ist, also auf der betrachteten Position im vorherigen Schritt schon ein Leerzeichen stand. a und c können aufgrund von ähnlichen Argumentationen wie vorher beliebig gewählt werden, wobei es zwei Ausnahmen gibt. Wenn a einer der Programmschritte $(s, \triangleright, s, \triangleright, \rightarrow)$ oder $(s, 0, s, 0, \rightarrow)$ ist, würde der Kopf im nächsten Schritt eine Position weiter rechts stehen und dort einen weiteren Programmschritt anwenden. D.h. wenn das a einer dieser beiden Programmschritte ist, ist das d auch ein Programmschritt (der der als nächstes nach a von der Maschine ausgeführt werden würde) und somit kein Leerzeichen. Folglich sind alle Tupel der Form $(a, \#, c\#)$ mit $a \in \Sigma''$ und c beliebig in C_M enthalten.
Nun betrachten wir die 4-Tupel der Form (a, b, c, d) mit $d \in \{(s, \#, h_y, \#, \leftarrow), (s, \#, h_n, \#, \perp)\}$. Wenn die Maschine einen Haltezustand erreicht, bleibt sie in diesem. D.h. wenn b ein Programmschritt mit Haltezustand ist, ist d der gleiche Programmschritt. Also sind alle Tupel der Form (a, b, c, b) mit a, c beliebig und $b \in \{(s, \#, h_y, \#, \leftarrow), (s, \#, h_n, \#, \perp)\}$ in C_M enthalten. Außerdem erreicht die Maschine einen Haltezustand, wenn sie da erste Mal

ein Leerzeichen erreicht. Das d kann demnach ein Programmschritt mit Haltezustand sein, wenn b ein Leerzeichen ist. Zusätzlich muss in diesem Fall jedoch das a ein Programmschritt sein, durch den der Kopf eine Position nach rechts, also auf das Leerzeichen, verschoben wird. Demnach sind alle Tupel der Form $(a, \#, c, d)$ mit $a \in \{(s, \triangleright, s, \triangleright, \rightarrow), (s, 0, s, 0, \rightarrow)\}$, c beliebig und $d \in \{(s, \#, h_y, \#, \leftarrow), (s, \#, h_n, \#, \bot)\}$ in C_M enthalten.
Weiter betrachten wir die 4-Tupel der Form $(a, b, c, 0)$. Es existieren zwei Möglichkeiten, wann das d eine 0 sein kann: Erstens wenn der Kopf im Schritt vorher auf dem b steht, eine 0 schreibt und sich dann nach links oder rechts bewegt. Dies entspricht in der betrachteten Maschine $b = (s, 0, s, 0, \rightarrow)$, also sind alle Tupel der Form $(a, (s, 0, s, 0, \rightarrow), c, 0)$ mit a, c, beliebig in C_M enthalten. Zweitens wenn die 0 im Schritt zuvor schon auf dem Band stand und nicht verändert wird. Dies entspricht in der betrachteten Maschine $b = 0$ und $a \in \Sigma''$. Wäre $a \in \{(s, \triangleright, s, \triangleright, \rightarrow), (s, 0, s, 0, \rightarrow)\}$, würde der Kopf im nächsten Schritt auf die Position von b bewegt werden und dort ein Programmschritt ausführen, d.h. d wäre ein Programmschritt und nicht 0. Demnach sind alle Tupel der Form $(a, 0, c, 0)$ mit $a \in \Sigma''$ und c beliebig in C_M enthalten.
Anschließend betrachten wir die 4-Tupel der Form $(a, b, c, (s, 0, s, 0, \rightarrow))$. Der Programmschritt $(s, 0, s, 0, \rightarrow)$ kann nur angewendet werden, wenn im Schritt zuvor eine 0 auf dem Band stand, d.h. wenn $b = 0$. Außerdem muss der Kopf im Schritt zuvor auf diese 0 bewegt werden, d.h. $a \in \{(s, \triangleright, s, \triangleright, \rightarrow), (s, 0, s, 0, \rightarrow)\}$. Demnach sind alle Tupel der Form $(a, 0, c, (s, 0, s, 0, \rightarrow))$ mit $a \in \{(s, \triangleright, s, \triangleright, \rightarrow), (s, 0, s, 0, \rightarrow)\}$ und c beliebig in C_M enthalten.
Zuletzt betrachten wir die 4-Tupel der Form $(a, b, c, (s, \triangleright, s, \triangleright, \rightarrow))$. Der Programmschritt $(s, \triangleright, s, \triangleright, \rightarrow)$ kann nur einmal am Anfang der Berechnung ausgeführt werden, da die Maschine danach das initiale Symbol nicht mehr betrachtet. Demnach kommen Tupel mit $d = (s, \triangleright, s, \triangleright, \rightarrow)$ nicht in C_M vor (keine Konfiguration sorgt dafür, dass im nächsten Schritt dieser Programmschritt angewendet wird).

Aufgabe 12:

Definieren Sie äquivalent zur NP-Vollständigkeit den Begriff der P-Vollständigkeit. Hierzu ist es notwendig Reduktionen auf logarithmischen Platz zu definieren. Erläutern Sie ebenfalls, warum $\leq_p$ zu diesem Zweck unzureichend ist.

Aufgabe 13:

Zeigen Sie, dass das folgende Problem NP-vollständig ist.

4-OMNI-SAT	
Gegeben:	Eine Menge von Boolschen Variablen X und eine Menge C von Klauseln über X, sodass (a) $\forall c \in C : \ \lvert c \rvert = 4$ (b) $\exists x \in X : \ \forall c \in C : \ (x \in c \vee \bar{x} \in c)$
Frage:	Ist C erfüllbar?

Ansatz: Reduktion von 3-SAT: Erzeuge zwei Kopien der 3-SAT Instanz und eine neue Variable x, füge einmal x und einmal $\overline{x}$ zu jeder Klausel hinzu.

Aufgabe 14:

Sei X eine Menge von Boolschen Variablen und C eine Menge von Klauseln über X. Der Klausel-Variablen-Graph $G_{(X,C)}$ hat einen Knoten x für jede Variable $x \in X$ und einen Knoten c für jede Klausel $c \in C$. Es gibt eine Kante $\{x, c\}$ zwischen einem Variabelknoten x und Klauselknoten c genau dann, wenn c das Literal x oder $\bar{x}$ enthält.

Zeigen Sie, dass das folgende Problem NP-vollständig ist.

CONNECTED SAT	
Gegeben:	Eine Menge von Boolschen Variablen X und eine Menge von Klauseln C über X, so dass $G_{(X,C)}$ zusammenhängend ist.
Frage:	Ist C erfüllbar?

Ansatz: Erzeuge zwei Kopien der SAT Instanz und eine neue Variable x, füge einmal x und einmal $\overline{x}$ zu jeder Klausel hinzu und erzeuge eine Klausel, die alle Literale enthält.

Aufgabe 15:

Zeigen Sie, dass das folgende Problem NP-vollständig ist.

TRAVELING SALESMAN PROBLEM (TSP)	
Gegeben:	Ein vollständiger Graph $K_n = (V, E)$ mit n Knoten, eine Kantengewichtsfunktion $c : E \to \mathbb{R}_+$ und eine Zahl $k \in \mathbb{R}_+$.
Frage:	Gibt es einen Hamiltonkreis $v_1, \ldots, v_n$, so dass $$c(\{v_n, v_1\}) + \sum_{i=1}^{n-1} c(\{v_i, v_{i+1}\}) \leq k \ ?$$

Ansatz: Reduktion von Hamiltonkreis: $k = n$, alle vorhandenen Kanten bekommen Gewicht 1, füge alle fehlenden Kanten zum Graphen hinzu und gib ihnen Gewicht $k + 1$.

Aufgabe 16:

Betrachten Sie das folgende Problem:

PARTITION IN k MENGEN	
Gegeben:	Werte $a_1, \ldots, a_n \in \mathbb{N}$.
Frage:	Gibt es k Indexmengen $R_1, R_2, \ldots, R_k \subseteq \{1, \ldots, n\}$ mit $R_1 \cup R_2 \cup \ldots \cup R_k = \{1, \ldots, n\}$ und $R_i \cap R_j = \emptyset$, $i \neq j$, $i, j \in \{1, \ldots, n\}$ sodass $\sum_{i \in R_1} a_i = \sum_{i \in R_2} a_i = \ldots = \sum_{i \in R_k} a_i$?

(a) Zeigen Sie, dass PARTITION IN 2 MENGEN NP-vollständig ist.

(b) Zeigen Sie, dass PARTITION IN 3 MENGEN NP-vollständig ist.

(c) Geben Sie einen pseudopolynomiellen Algorithmus für PARTITION IN 2 MENGEN an.

Ansatz: (a) Reduktion von Knapsack, wobei $a_i = g_i$ und $A = G$ für alle Objekte: Füge ein neues Objekt mit Gewicht $|2 \cdot A - S|$ hinzu, wobei $S = \sum a_i$.
(b) Reduktion von Partition in 2 Mengen: Füge ein neues Objekt mit Gewicht $\frac{1}{2} \cdot \sum a_i$ hinzu.

Aufgabe 17:

Formulieren Sie die Optimierungsprobleme MINIMUM VERTEX COVER und MAXIMUM KNAPSACK, indem Sie jeweils folgendes angeben:

- Die Menge aller zulässigen Eingaben D,
- die Menge aller Lösungen $S(I)$ für eine Instanz $I \in D$,
- eine Bewertungsfunktion f, die jedem $s \in S(I)$ einen positiven Wert zuordnet.

Aufgabe 18:

Führen Sie den FIRST-FIT Algorithmus mit den folgenden Instanzen für das Optimierungsproblem MINIMUM BIN PACKING durch und bestimmen Sie jeweils den Wert $R_A(I)$.

(a) $I = (U, g)$ mit $U := \{o_1, \ldots, o_{16}\}$ und $g(o_i) := \begin{cases} \frac{1}{8}, & i \text{ gerade} \\ \frac{1}{2}, & i \text{ ungerade} \end{cases}$

(b) $I = (U, g)$ mit $U := \{o_1, \ldots, o_{18}\}$ und $g(o_i) := \begin{cases} 0.15, & 1 \leq i \leq 6 \\ 0.34, & 7 \leq i \leq 12 \\ 0.51, & 13 \leq i \leq 18 \end{cases}$

Aufgabe 19:

Zeigen Sie:
Seien P und P' zwei Probleme. Wenn P' FPT ist und P parametrisiert auf P' transformiert werden kann, dann ist P FPT.

Aufgabe 20:

Zeigen Sie, dass das folgende Problem FPT ist:

p-SAT	
Gegeben:	Eine Menge X von Variablen, eine Menge C von Klauseln.
Parameter:	$p = \|X\|$
Frage:	Gibt es eine Belegung der Variablen, die alle Klauseln erfüllt

Aufgabe 21:

Zeigen Sie, dass das Problem 2-SAT in P liegt.