Informatik III - Tutorium XIII & XV (SR -107)

Tut Nr. 7 – Üb6, $\mathcal{NP}$-Vollständigkeit

David Münch

Universität Karlsruhe (TH)
Institut für Informatik
ITI Wagner

12. Dezember 2007

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Organisatorisches

Email: muenchdavid@gmail.com

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

Tutorium 13: Mittwochs 8:00 Uhr - Raum -107
Tutorium 15: Mittwochs 9:45 Uhr - Raum -107

Übungsblattabgabe Donnerstag.

Was wollen wir heute erreichen?

Was wollen wir heute erreichen?

- Beweise für $\mathcal{NP}$-Vollständigkeit verstehen und selbst durchführen können.

Aufgabe 5

Gegeben sei ein Graph $G = (V, E)$ mit $V := \{1, \ldots, n\}$ und $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$. Weiter seien $c : E \to \mathbb{R}^+$ eine Längenfunktion auf den Kanten des Graphen sowie $s \in V$ ein ausgewiesener Start- und $t \in V$ ein ausgewiesener Zielknoten. Ein Pfad von s nach t ist eine Folge von Kanten $p = \{i_1, i_2\}, \{i_2, i_3\}, \ldots, \{i_{k-2}, i_{k-1}\}, \{i_{k-1}, i_k\}$ mit $\{i_j, i_{j+1}\} \in E$ und $i_1 = s$ sowie $i_k = t$.

Die Länge des Pfades sei definiert als die Summe der Längen aller Kanten auf dem Pfad. Betrachten Sie die Menge der Pfade, die von s nach t führen, und formulieren Sie ein naheliegendes Optimierungsproblem.

Formulieren Sie ein zu Ihrem Optimierungsproblem gehörendes Entscheidungsproblem.

Aufgabe 5

Optimierungsproblem:
Finde einen Pfad von s nach t mit minimaler Länge.

Entscheidungsproblem:
Gibt es einen Pfad von s nach t der Länge $\leqslant k$

Aufgabe 6

Seien L_1 die Sprache der ungeraden Zahlen in Dezimaldarstellung über $\Sigma_1 = \{0, \ldots, 9\}$ und L_2 die Sprache der Wörter gerader Länge über $\Sigma_2 = \{a, b\}$. Zeigen Sie, dass L_1 polynomial in L_2 transformiert werden kann ($L_1 \propto L_2$).

Hinweis: Sie müssen die DTM, die die Transformation berechnet, nicht explizit angeben, sondern lediglich kurz in Worten beschreiben, wie sie arbeitet, und begründen, warum ihre Laufzeit polynomial ist.

Aufgabe 6

Definition: polynomiale Transformation

Eine polynomiale Transformation einer Sprache $L_1 \subseteq \Sigma_1^*$ in eine Sprache $L_2 \subseteq \Sigma_2^*$ ist eine Funktion $f : \Sigma_1^* \to \Sigma_2^*$ mit den Eigenschaften:

1. es existiert eine **polynomial deterministische** Turing-Maschine, die f berechnet
2. für alle $x \in \Sigma_1^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$

Wir schreiben dann $L_1 \propto L_2$ (L_1 ist polynomial transformierbar in L_2).

Aufgabe 6

Wir definieren uns eine Funktion $f : \Sigma_1^* \to \Sigma_2^*$

$$f(z) = \begin{cases} aa, & \text{wenn } z \text{ ungerade ist} \\ a, & \text{sonst} \end{cases}$$

Jetzt müssen wir nur noch zeigen, dass f von einer deterministischen Turing-Maschine berechnet werden kann. Eine solche DTM erhält als Eingabe eine Dezimalzahl. Zunächst löscht sie alle Zeichen der Eingabe bis auf das Letzte. Wenn das letzte Zeichen der Eingabe ungerade ist, schreibt unsere TM aa, sonst a. Die Laufzeit unserer DTM ist in $\mathcal{O}(z)$. Somit ist sie auf jeden Fall polynomial beschränkt.
Sei nun $z \in \Sigma_1^*$. Dann ist
$f(z) \in L_2 \Leftrightarrow f(z) = aa \Leftrightarrow z$ ist ungerade $\Leftrightarrow z \in L_1$. Somit ist f eine polynomiale Transformation von L_1 in L_2.

Das Entscheidungsproblem *PRIMES*[a] besteht darin, zu entscheiden, ob es sich bei einer gegebenen natürlichen Zahl $p > 1$ um eine Primzahl handelt. Eine Probleminstanz von *PRIMES* wird also durch eine natürliche Zahl kodiert. Für $b \in \mathbb{N}$ bezeichne s_b das Kodierungsschema, welches Zahlen zur Basis b darstellt.

[a] Erst 2002 konnte gezeigt werden, dass *PRIMES* in $\mathcal{P}$ liegt.

Aufgabe 7

Zeigen Sie, dass die Kodierungsschemata s_a und s_b für $a, b > 1$ bezüglich *PRIMES* äquivalent sind.

Aufgabe 7

Sei I eine Instanz von PRIMES, d.h. $I \in \mathbb{N}, I > 1$. Da wir kein Vorzeichen benötigen, gilt $|s_a(I)| = \lfloor \log_a I \rfloor + 1$ und analog $|s_b(I)| = \lfloor \log_b I \rfloor + 1$. Damit gilt:

$$|s_a(I)| = \lfloor \log_a I \rfloor + 1 = \lfloor \frac{\ln I}{\ln a} \rfloor + 1 \leq \frac{\ln I}{\ln a} + 1 = \frac{\ln b}{\ln b} \frac{\ln I}{\ln a} + 1$$

$$= \frac{\ln b}{\ln a} \log_b I + 1 \leq \frac{\ln b}{\ln a} (\lfloor \log_b I \rfloor + 1) + 1 = \frac{\ln b}{\ln a} |s_b(I)| + 1 =: p_a(|s_b(I)|)$$

mit $p_a(x) := \frac{\ln b}{\ln a} x + 1$.
Analog haben wir: $p_b(x) := \frac{\ln a}{\ln b} x + 1$.
Damit haben wir gezeigt, dass es Polynome p_a und p_b gibt mit $|s_a(I)| \leq p_a(|s_b(I)|)$ und $|s_b(I)| \leq p_b(|s_a(I)|)$ für alle Instanzen I von PRIMES. Also sind s_a und s_b äquivalent.

Aufgabe 8

Betrachten Sie $L_1 := L[\mathit{PRIMES}, s_1]$. Ein naiver Algorithmus für *PRIMES* könnte alle Zahlen $2, 3, \ldots, n-1$ darauf hin überprüfen, ob sie die gegebene Zahl p teilen. Beschreiben Sie kurz in Worten die Arbeitsweise einer **deterministischen** Turing-Maschine, die diesen Algorithmus implementiert und damit L_1 entscheidet. Geben Sie die Laufzeit Ihrer Turing-Maschine asymptotisch an. Ist L_1 in $\mathcal{P}$? Begründen Sie gegebenenfalls, warum L_1 in $\mathcal{P}$ ist.

Aufgabe 8

Die Eingabe der DTM ist eine Folge von n Einsen.

(1) Initialisiere den Teilerkandidaten:

Gehe zum Ende der Eingabe und schreibe $\$1$. $\rightarrow \mathcal{O}(n)$

(2) Inkrementiere den Teilerkandidaten um Eins:

Gehe zum Ende des Teilerkandidaten und schreibe eine 1. $\rightarrow \mathcal{O}(n)$

(3) Abbruchkriterium:

Überprüfe ob die Länge des Teilerkandidaten der Länge der Eingabe entspricht. Falls ja, dann halte in akzeptierendem Zustand: Dazu werden die Einsen der Eingabe und dem Teilerkandidaten abwechselnd mit $*$ überschrieben. Falls die Eingabe noch Einsen enthält, überschreibe auch die restlichen Einsen jeweils mit $*$, ansonsten halte in akzeptierendem Zustand. $\rightarrow \mathcal{O}(n^2)$

weil n Symbole der Eingabe überschrieben und der Kopf der TM für jedes Symbol max. $2n + 2$ Schritte auf dem Band machen muss. Es wurden alle Einsen auf dem Band mit $*$ überschrieben.

Aufgabe 8

(4) Dividiere die Eingabe durch den aktuellen Teiler
Solange die Eingabe noch $*$ enthält, ersetze für jeden $*$ des Teilers einen $*$ der Eingabe wieder durch eine Eins. Falls der Teilerkandidat keinen $*$ mehr enthält, die Eingabe aber noch einen $*$ enthält, überschreibe die Einsen des Teilers wieder mit $*$ und überschreibe weiter für jeden $*$ des Teilerkandidaten einen $*$ der EIngabe mit einer Eins. Falls die Division aufgeht (Divisor und Divident haben keine $*$ mehr), halte in nichtakzeptierendem Endzustand, weil Teiler gefunden. $\rightarrow \mathcal{O}(n^2 + n)$
(5) Weiter mit (2)

(2)-(5) liegt in $\mathcal{O}(n^2)$ und wird n mal ausgeführt. $\rightarrow \mathcal{O}(n^3)$.
$\Rightarrow L_1 \in \mathcal{P}$, weil es eine DTM mit polynomial beschränktem Aufwand gibt.

Aufgabe 9

Betrachten Sie die Sprache $L_{10}^c := L[\mathit{PRIMES}, s_{10}]^c$ der *zusammengesetzten* Zahlen in Dezimaldarstellung. Beschreiben Sie grob in Worten die Arbeitsweise einer **nichtdeterministischen** Turing-Maschine, die L_{10}^c entscheidet. Ein Unterprogramm für die Division von Dezimalzahlen mit polynomieller Laufzeit sei gegeben und kann von Ihrer Turing-Maschine aufgerufen werden. Ist L_{10}^c in $\mathcal{NP}$? Begründen Sie gegebenenfalls, warum L_{10}^c in $\mathcal{NP}$ ist.

Aufgabe 9

Programm unserer NTM:
Orakel schreibt z auf das Band. Wir müssen überprüfen ob $z \in \{2, \ldots, p-1\}$:
z eine Folge über $\{0, \ldots, 9\}$ und nicht länger als Eingabe und weder 1 noch p ist.$\rightarrow \mathcal{O}(n^2)$
Falls z nicht unseren Anforderungen entspricht, hält unsere TM in einem nichtakzeptierenden Zustand.
Andernfalls ruft unsere TM ein Unterprogramm (z.B. Aufgabe 8, polynomial beschränkt!) auf, um zu überprüfen ob $z|p$. Falls dies der Fall ist, dann hält die TM in einem akzeptierennden, sont nichtakzeptierenden Zustand.
$\rightarrow \mathcal{O}(n^3)$
Somit ist $L_{10}^c \in \mathcal{NP}$

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: $L' \propto L$

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: $L' \propto L$

Definition: polynomiale Transformation

Eine polynomiale Transformation einer Sprache $L_1 \subseteq \Sigma_1^*$ in eine Sprache $L_2 \subseteq \Sigma_2^*$ ist eine Funktion $f : \Sigma_1^* \to \Sigma_2^*$ mit den Eigenschaften:

1. es existiert eine **polynomial deterministische** Turing-Maschine, die f berechnet
2. für alle $x \in \Sigma_1^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$

Wir schreiben dann $L_1 \propto L_2$ (L_1 ist polynomial transformierbar in L_2).

Problem **DOUBLE-SAT**:
Gegeben: Eine Menge $U = \{u_1, \ldots, u_m\}$ von booleschen Variablen, eine Menge C von Klauseln über U.

Frage: Existieren *zwei verschiedene* Wahrheitsbelegungen von U, sodass C erfüllt wird, d.h., dass alle Klauseln aus C den Wahrheitswert $wahr$ annehmen?

Problem **DOUBLE-SAT**:
Gegeben: Eine Menge $U = \{u_1, \ldots, u_m\}$ von booleschen Variablen, eine Menge C von Klauseln über U.

Frage: Existieren *zwei verschiedene* Wahrheitsbelegungen von U, sodass C erfüllt wird, d.h., dass alle Klauseln aus C den Wahrheitswert $wahr$ annehmen?

Aufgabe

Zeige, dass das Problem **DOUBLE-SAT** $\mathcal{NP}$-vollständig ist.

Problem **HAMILTONIAN CIRCUIT**:
Gegeben: Ein Graph $G = (V, E)$
Frage: Existiert ein einfacher Kreis in G, der alle Knoten in V enthält, also eine Folge $(v_1, \ldots, v_n)$ von paarweise verschiedenen Knoten $v_i \in V (i = 1, \ldots, n)$ mit $n := |V|$ und $\{v_j, v_{j+1}, \{v_n, v_1\} \in E (j = 1, \ldots, n-1)\}$?
Hinweis: HAMILTONIAN CIRCUIT ist $\mathcal{NP}$-vollständig.

Problem **HAMILTONIAN CIRCUIT**:
Gegeben: Ein Graph $G = (V, E)$
Frage: Existiert ein einfacher Kreis in G, der alle Knoten in V enthält, also eine Folge $(v_1, \ldots, v_n)$ von paarweise verschiedenen Knoten $v_i \in V (i = 1, \ldots, n)$ mit $n := |V|$ und $\{v_j, v_{j+1}, \{v_n, v_1\} \in E (j = 1, \ldots, n-1)\}$?
Hinweis: HAMILTONIAN CIRCUIT ist $\mathcal{NP}$-vollständig.

Problem **RURAL POSTMAN**:
Gegeben: Ein Graph $G = (V, E)$, Teilmenge $E' \subseteq E$ und Parameter $K \in \mathbb{N}$
Frage: Existiert ein (nicht notwendigerweise einfacher) Kreis der Länge höchstens K in G, der alle Kanten aus E' enthält, also eine Folge $(v_1, \ldots, v_l)$ mit $v_i \in V (i = 1, \ldots, l), l <= K$ und $E' \subseteq \{\{v_j, v_{j+1} \mid j = 1, \ldots, l-1\} \cup \{\{v_l, v_1\}\} \subseteq E$?
Hinweis: Der Graph G ist nicht notwendigerweise schleifenfrei, d.h. er kann auch Kanten $\{v, v\}$ enthalten.

Aufgabe

Zeige, dass das Problem **RURAL POSTMAN** $\mathcal{NP}$-vollständig ist.

Reflexion

Was haben wir heute gelernt?

Reflexion

Was haben wir heute gelernt?

- $\mathcal{NP}$-Vollständigkeit

Noch Fragen?

Vorschau

Vorschau

- Tasse nicht vergessen!!!

Bis zum nächsten Mal