technische universität
dortmund

Beate Bollig, Marc Gillé, Jens Lechner
Lehrstuhl 2, Fakultät für Informatik
Sommersemester 2011

TIfAI Übung – Blatt 6

Ausgabedatum: 10.5.2011 — Abgabedatum: 16.5.2011, 14:00 Uhr

Aufgabe 6.1: NP-Vollständigkeit

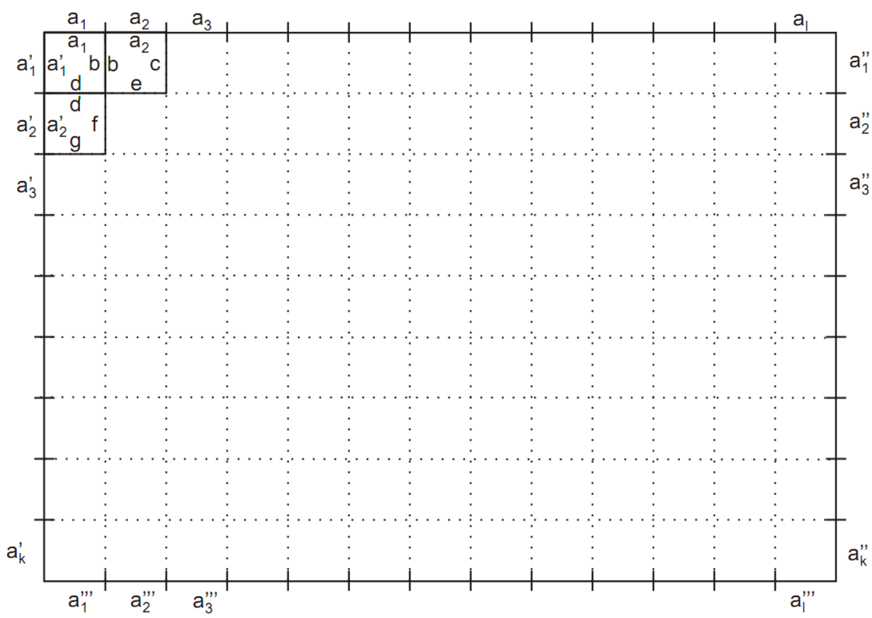
Kurzaufgabe (1 Punkt):

Nenne die drei prinzipiellen Typen von polynomiellen Reduktionen und jeweils eine Beispielreduktion von jedem Typ.

Hauptaufgabe (4 Punkte):

Betrachte folgendes Puzzle-Problem PZ mit folgender Eingabe:

- A ist eine endliche Menge von Beschriftungen und $k, l \in \mathbb{N}$ sind zwei natürliche Zahlen.
- Die rechteckige Grundfläche besteht aus $k \cdot l$ quadratischen und gleichgroßen 1×1 Einzelfeldern mit k Zeilen und l Spalten. Jeder Randabschnitt einer Zeile bzw. einer Spalte hat eine Beschriftung $a \in A$.
- Die Menge der Puzzle-Teile besteht aus $n \geq k \cdot l$ quadratischen 1×1 Puzzle-Teilen. Jedes Puzzle-Teil hat an jeder Seite eine Beschriftung $a \in A$ und kann nicht gedreht werden, d.h. es muss so in das Puzzle-Feld gesetzt werden, dass die Beschriftung des oberen Rands wirklich oben liegt.



Eine Eingabe von PZ wird genau dann akzeptiert, wenn es möglich ist, das Puzzle-Feld komplett mit Puzzle-Teilen zu besetzen, so dass alle angrenzenden Beschriftungen gleich sind.

Aus der Vorlesung ist bekannt, dass 3-SAT NP-vollständig ist.

Zeige, dass auch PZ NP-vollständig ist.

Aufgabe 6.2: Approximationen für TSP

Kurzaufgabe (1 Punkt):

Ein ungerichteter *Multigraph* G ist ein ungerichteter Graph, bei dem eine Kante zwischen Knoten v und w mehrfach vorhanden sein kann. Ein *Eulerkreis* in G ist ein Kreis, der jede Kante genau einmal enthält (d.h. er kann Knoten mehrfach besuchen). Sei G ein zusammenhängender Multigraph. Zeige, dass G genau dann einen Eulerkreis hat, wenn alle Knoten in G geraden Grad haben.

Hauptaufgabe (4 Punkte):

Falls P $\neq$ NP gilt, wissen wir aus der Vorlesung, dass es keine deterministischen polynomiellen Algorithmen für NP-schwere Probleme gibt. Da diese Probleme in der Praxis trotzdem vorkommen, braucht man andere Methoden, um eine zufriedenstellende Lösung zu bekommen. Im Falle von Optimierungsproblemen benutzt man häufig sogenannte *Approximationsalgorithmen*. Ein Approximationsalgorithmus ist ein Algorithmus, der in polynomieller Zeit eine Näherungslösung berechnet, dessen Wert „nah“ an dem Wert einer optimalen Lösung ist. Im folgenden betrachten wir nur Minimierungsprobleme, das bedeutet, wir suchen eine gültige Lösung mit minimalem Wert. Die *Güte* von einem Approximationsalgorithmus A für eine Eingabe x ist

$$\frac{v_A(x)}{v_{OPT}(x)},$$

wobei $v_A(x)$ der Wert der Lösung von A bei Eingabe x ist und $v_{OPT}(x)$ der optimale Wert der Lösung bei Eingabe x. Wir nennen A eine c-Approximation, falls $\frac{v_A(x)}{v_{OPT}(x)} \leq c$ für alle Eingaben x gilt.

Als Eingabe für das allgemeine *Traveling Salesman Problem* (TSP) erhält man einen vollständigen Graphen $G = (V, E)$ und eine Distanzfunktion $d : V \times V \to \mathbb{N}_0$, die die Distanzen zwischen je zwei Städten angibt. Gesucht ist die kürzeste Rundreise, die alle n Knoten (Städte) besucht.

Eine Variante des TSP ist das *metrische TSP*, bei der die Distanzfunktion eine Metrik ist, d.h. es gilt $d(i, j) = d(j, i)$ für alle $i \neq j$ und $d(i, j) \leq d(i, k) + d(k, j)$ für alle $i, j, k \in V$.

1. Zeige, dass es keine c-Approximation für das allgemeine TSP mit $c \leq 2^n$ gibt, wenn nicht P = NP ist.

   Hinweis: Betrachte die Reduktion $HC \leq_p TSP_{dec}$.

2. Zeige, dass der folgende Algorithmus eine 2-Approximation für das metrische TSP ist:

   (a) Berechne einen minimalen Spannbaum $T = (V, E_T)$ auf G.

   (b) Erzeuge einen Multigraphen $G' = (V, E')$ mit $E' = \{e, e \mid e \in E_T\}$, d.h. G' enthält alle Kanten von T und fügt noch eine Kopie von jeder Kante aus T ein.

   (c) Berechne einen Eulerkreis K auf G'.

   (d) Berechne eine Rundreise, in dem K durchlaufen wird und ein mehrfaches Vorkommen von einem Knoten entfernt wird.

Aufgabe 6.3: Satz von Cook

Kurzaufgabe (1 Punkt):

Begründe ausführlich die Korrektheit des Beweises von dem Satz von Cook (Folie 230). Beschreibe also, wie man eine erfüllende Belegung konstruiert, wenn $x \in \mathrm{L}$ ist, und wie man aus einer erfüllenden Belegung einen akzeptierenden Rechenweg bekommt.

Hauptaufgabe (4 Punkte):

Die folgende Tabelle beschreibt die Übergangsfunktion δ einer deterministischen (d.h. die Variablen $Z(t)$ (siehe Folie 220) werden nicht benötigt), stereotypen Turingmaschine M mit dem Eingabealphabet $\Sigma = \{0, 1\}$ und der Zustandsmenge $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$.

δ	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
0	(q_1, B)	$(q_1, 0)$	$(q_2, 0)$	(q_5, B)	(q_7, B)	$(q_5, 0)$	$(q_6, 0)$	$(q_7, 0)$
1	(q_2, B)	$(q_1, 1)$	$(q_2, 1)$	(q_7, B)	(q_5, B)	$(q_5, 1)$	$(q_6, 1)$	$(q_7, 1)$
B	(q_6, B)	$(q_3, B$	(q_4, B)	(q_7, B)	(q_7, B)	(q_0, B)	(q_6, B)	(q_7, B)

Der einzige akzeptierende Zustand ist q_6 und der einzige verwerfende Zustand ist q_7.

M akzeptiert alle Wörter der Form ww^R mit $w \in \{0, 1\}^*$, d.h. M überprüft, ob die Eingabe ein Palindrom ist. Dazu liest und merkt sich die TM das Zeichen am linken Ende, überschreibt es mit einem B und überprüft, ob das Zeichen am rechten Ende der Eingabe mit diesem Zeichen übereinstimmt und überschreibt auch das Zeichen mit einem B. Danach sucht die TM das nächste Zeichen ungleich B am linken Ende und setzt die Berechnung rekursiv fort. Falls irgendwann ein falsches Zeichen vorkommt, wechselt die TM in den verwerfenden Zustand. Wenn alle Zeichen ohne Fehler abgearbeitet sind, wechselt die TM in den akzeptierenden Zustand.

Für eine Eingabe der Länge 2 seien die Funktionen $t(j)$ und $N(t)$ aus dem Beweis des Satzes von Cook gegeben:

	0	1	2
$t(j)$	1	2	3

	1	2	3	4	5	6	7
$N(t)$	5	4	-	6	-	7	-

$N(t) =$ - bedeutet, dass die Speicherstelle, die in Schritt t angeschaut wird, nach Schritt t von der TM nicht wieder besucht wird.

Gib explizit die Klauselmenge an, die für M mit der Eingabe 11 im Satz von Cook konstruiert wird. Beschränke dich für die Codierung der Übergangsfunktion (Folien 226f) nur auf die Zustände q_0, q_2 und q_4, d.h. gib für den Teil nur die Klauseln an, die die korrekte Rechnung codieren, wenn die TM zum Zeitpunkt t im Zustand q_0, q_2 oder q_4 ist und ein Zeichen liest.

Testfragen:

1. Wie kann man nichtdetermistische Algorithmen in deterministische Algorithmen umformen?
2. Warum interessieren wir uns für die NP-Vollständigkeitstheorie?
3. Was sind die wesentlichen Ideen für den Beweis vom Satz von Cook?