Technische Universität München
Fakultät für Informatik
Prof. Dr. M. Broy

SS 2003
Übungsblatt 8
13. Juni 2003

Übungen zur Vorlesung Einführung in die Informatik IV

Aufgabe 33 Erfüllbarkeitsproblem

Wir betrachten das Erfüllbarkeitsproblem der Aussagenlogik, eines der klassischen NP-vollständigen Probleme. Der Einfachheit halber beschränken wir uns auf die logischen Konnektive $\neg$ (nicht) und $\vee$ (oder), mit denen alle anderen Konnektive dargestellt werden können.

Zur Wiederholung: Für eine gegebene Menge V aussagenlogischer Variablen, ist die entsprechende Sprache der Aussagenlogik A_V wie folgt definiert:

- Jede aussagenlogische Variable aus V ist in A_V.
- Wenn F und G in A_V sind, dann auch $\neg F$ und $(F \vee G)$.

Eine *Belegung* für eine Variablenmenge V ist eine Abbildung $\iota : V \to \{\mathbf{t}, \mathbf{f}\}$. Jede Belegung kann fortgesetzt werden auf Formeln aus A_V wie folgt:

- $\iota(\neg F) = \mathbf{w}$ gdw $\iota(F) = \mathbf{f}$
- $\iota((F \vee G)) = \mathbf{w}$ gdw $\iota(F) = \mathbf{w}$ oder $\iota(G) = \mathbf{w}$

Eine aussagenlogische Formel F ist *erfüllbar*, wenn es eine Belegung gibt, sodass $\iota(F) = \mathbf{w}$. Eine solche Belegung nennt man auch *Modell* für die Formel F.

(a) Geben sie ein nichtdeterministisches Programm in Pseudo-Code an (unter Benutzung des nichtdeterministischen Verzweigungsoperators []), sodass für jede Formel F der obigen Form als Eingabe eine Berechnung existiert, die mit dem Wert 1 terminiert gdw F erfüllbar ist.

(b) Beschreiben Sie eine nichtdeterministische Turingmaschine mit zwei Bändern, die genau die erfüllbaren Formeln der Aussagenlogik akzeptiert.

Aufgabe 34 Rundreiseproblem

Gegeben sei eine Menge von Knoten (Städte) $V = \{1, .., n\}$ und eine Abstandsfunktion dist: $V^2 \to \mathbb{N}$ mit dist(i, j) ≥ 0 und dist(i, j) = dist(j,i). Gesucht ist nun eine Rundreise, d.h. eine zyklische Permutation $(\pi(1), ..., \pi(n))$ der Knoten in V, so dass

$$(\sum_{i=1}^{n} dist(\pi(i), \pi(i+1))$$

minimal wird.

(a) Formulieren Sie einen Algorithmus zur exakten Berechnung einer derartigen Rundreise und diskutieren Sie den Aufwand für diesen Algorithmus!

(b) Formulieren Sie einen Algorithmus, der mit geringerem Aufwand eine Näherungslösung für das Rundreiseproblem berechnet.

Aufgabe 35 Rundreiseproblem vs. Erfüllbarkeitsproblem

Zeigen Sie, dass das Rundreiseproblem (in der Version: es gibt eine Rundreise mit $\leq k$ Kosten) NP-vollständig ist, d.h dass

(1) das Rundreiseproblem in NP ist und

(2) sich jedes Problem in NP polynomiell auf das Rundreiseproblem reduzieren lässt. (Hinweis: Entwickeln Sie dazu eine polynomielle Reduktion des Erfüllbarkeitsproblems auf das Rundreiseproblem.)

Aufgabe 36 (P) Rundreiseproblem

In dieser Aufgabe soll eine Lösung des Rundreiseproblems in Java implementiert werden. Hierbei kann der im Verlauf einer Rundreise zu durchlaufende Graph aus n Knoten als quadratische Matrix D mit n Zeilen und Spalten repräsentiert werden, deren Elemente $d_{ij}(0 \leq i, j \leq n)$ die Distanz zwischen den Knoten i und j im Graph wiedergeben. Ein Pfad durch einen solchen Graphen entspricht somit einer n-elementigen Sequenz von Knoten-Indizes, während seine Länge durch die Summe der Einzeldistanzen zwischen aufeinander folgenden Elementen der Sequenz bestimmt ist (zyklischen Schluss beachten!).

(a) Entwickeln Sie eine Klasse `TSP`, die für einen fest vorgegebenen Graphen aus 3 Knoten eine Lösung des Rundreiseproblems berechnet. Entwickeln Sie hierfür neben einer geeigneten Repräsentation für Graphen und Pfade in Java eine Methode `pathLength()`, welche die Länge eines gegebenen Pfades ermittelt. Diese Funktionalität kann von einer Methode `shortestPathStatic()` genutzt werden, um sämtliche möglichen Pfade für das statische Beispiel hinsichtlich ihrer Länge zu untersuchen und den kürzesten Pfad zu bestimmen.

(b) Erweitern Sie die Klasse `TSP` um die Möglichkeit, beliebige Graphen zur Laufzeit einzulesen und die jeweils kürzeste Rundreise zu berechnen. Hierfür liest die Methode `readGraph()` die Beschreibung des Graphen in einem geeigneten Format aus einer Textdatei, während die Methode `shortestPath()` dynamisch sämtliche möglichen Pfade generiert und hinsichtlich ihrer Länge untersucht. Für die Generierung der Pfade kann eine im Rahmen der Lösungsvorlage bereitgestellte Klasse `PermutationGenerator` verwendet werden.

(c) Testen Sie die korrekte Umsetzung der in Teilaufgabe a) und b) entwickelten Funktionalität mit verschiedenen Graphen und dokumentieren Sie ihre Ergebnisse. Schätzen Sie den erforderlichen Aufwand zur Ermittlung einer Lösung für eine wachsende Anzahl von Knoten ab.