EFFIZIENTE ALGORITHMEN

Übungsblatt 8

Prof. Dr. Woeginger, PD Dr. Unger, Prof. Dr. Rossmanith
Dennis Fischer
Lehrstuhl für Informatik 1
RWTH Aachen

WS 18/19
6. Dezember
Abgabe: **20. Dezember 18:00**

- Die Übungsblätter sollen in Gruppen von 3-5 Studierenden abgegeben werden.
- Die abgegebenen Lösungen mit Namen und Matrikelnummern aller Teammitglieder und der Übungsgruppe beschriften.
- Um zur Klausur zugelassen zu werden müssen 50% aller möglichen Übungspunkte erreicht werden.

Hinweis zu Programmieraufgaben: Es werden nur Abgaben akzeptiert in vernünftigen Programmiersprachen (Beispiele: C, C++, C#, Java, Haskell, Python, ...). Die Programme müssen gut kommentiert und verständlich sein.

Aufgabe 1 (4 Punkte)

Betrachte die Heuristik *Nearest Insertion* zur Berechnung einer minimalen Tour für das metrische TSP:

Die Starttour besteht aus einer Tour der Länge 0 in einem beliebigen Knoten. Diese wird schrittweise folgendermaßen erweitert: Bestimme den noch nicht genutzten Knoten v mit dem geringsten Abstand zu einem der benutzten Knoten w auf der Tour. Füge v in die Tour nach w ein. Wiederhole, bis alle Knoten benutzt sind.

Zeige, dass die Heuristik Nearest Insertion eine 2-Approximation für das metrische TSP berechnet.

Aufgabe 2 (4 Punkte)

Das euklidische TSP ist eine Einschränkung des metrischen TSP. Die Knoten sind nun Punkte $(x, y) \in \mathbb{R}^2$ und die Distanzfunktion ist der euklidische Abstand $\operatorname{dist}((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Wir betrachten die Nearest Insertion Heuristik aus Aufgabe 1. Wir wollen eine untere Schranke für den Approximationsfaktor zeigen.

Zeigen Sie, dass es eine unendliche Menge von Instanzen $\mathcal{I} = \{I_1, I_2, \ldots\}$ mit $\operatorname{opt} I_i < \operatorname{opt} I_{i+1}$ gibt mit

$$\lim_{n\to\infty} \frac{\mathsf{A}_{\mathrm{NN}}(I_n)}{\operatorname{opt}(I_n)} \geq \frac{3}{2}.$$

Um die untere Schranke zu zeigen hätte es gereicht, eine einzelne Instanz anzugeben. Was für einen Vorteil hat es, eine solche Folge von Instanzen zu betrachten?

Aufgabe 3 (2 Punkte)

Das *Rationale Knapsack Problem* ist folgendes Optimierungsproblem: Gegeben Werte $p_1, \ldots, p_n$, $w_1, ..., w_n$ und K.

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} p_i x_i \\ \text{subject to:} & \sum_{i=1}^{n} w_i x_i \leq K \\ & 0 \leq x_i \leq 1 \qquad i = 1 \ldots n \end{array}$$

Geben Sie einen Algorithmus für dieses Problem mit Laufzeit $\mathcal{O}(n \log n)$ an.

Aufgabe 4 (4 Punkte)

Das *Knapsack Problem* ist folgendes Optimierungsproblem: Gegeben Werte $p_1, \ldots, p_n$, $w_1, ..., w_n$ und K.

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} p_i x_i \\ \text{subject to:} & \sum_{i=1}^{n} w_i x_i \leq K \\ & x_i \in \{0, 1\} \qquad i = 1 \ldots n \end{array}$$

Geben Sie einen Algorithmus an, der das Knapsack Problem mit Backtracking löst. Was sind geeignete *bounding functions* B_j?

Aufgabe 5 (2 Punkte)

Modellieren Sie das Subset Sum Problem als ILP. Sind die Ecken des (relaxierten) Raumes der zulässigen Lösungen stets ganzzahlig? Nutzen Sie ein komplexitätstheoretisches Argument.

Aufgabe 6 (10 Punkte)

Wie viele Teilmengen von

$$\{163, 475, 160, 674, 85, 916, 461, 307, 349, 86, 198, 466, 709, 973, 981, 374, 766, 473, 342, 191, 393, 300, 11\}$$

gibt es, deren Summe genau 3652 ist?

(a) Schreiben Sie ein Programm, welches dieses Problem mittels Backtracking löst.

(b) Schreiben Sie ein Programm, welches dieses Problem mittels Dynamischer Programmierung löst.

(c) Betrachten Sie die Eingabeinstanz

$$\{1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009,$$
$$1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019,$$
$$1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029,$$
$$1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039\}$$

mit Zielwert 20000 (und 19362). Wie ist dort die Anzahl der Lösungen? Welchen der Algorithmen aus (a) und (b) haben Sie genutzt?

Aufgabe 7 $(\star)$ (8 Bonus Punkte)

Eine Familie ist im Urlaub und geht an den (1km langen) Strand und stellt dort am ersten Tag ihren Sonnenschirm auf. Dieser soll dort für den restlichen Urlaub stehen bleiben und kann nicht versetzt werden. In den darauffolgenden Tagen kommt jeweils eine weitere Familie und macht das gleiche. Dabei soll der Strand gleich unter den Familien aufgeteilt werden. Das heißt an Tag i, falls man den Strand in i gleichgroße Teile aufteilt, steht in jedem Teil genau ein Sonnenschirm.

Wie viele Tage lang kann das maximal funktionieren?

(a) Schreibt ein Programm welches die maximale Anzahl von Tagen berechnet.

(b) Visualisiere das Ergebnis so, dass leicht zu sehen ist, dass die Lösung korrekt ist.

Abgabefrist: Die Lösungen müssen bis zum **20. Dezember 18:00** in der Vorlesung oder im Abgabekasten vor dem Lehrstuhl i1 abgegeben werden.