Universität des Saarlandes
FR 6.2 Informatik

Prof. Dr. Kurt Mehlhorn, Dr. Konstantinos Panagiotou **WiSe 2011/2012**

Übungen zu Computational Thinking

http://www.mpi-inf.mpg.de/departments/d1/teaching/ws11/ct/

Blatt 8 Abgabeschluss: 9.1.12 16:00

Regeln: Bis zum Semesterende müssen mindestens 42% der maximal erreichbaren Punkte aller Übungszettel erworben werden.
Programmcode ist elektronisch per E-Mail abzugeben. Zusätzlich müssen die Ausgaben einer exemplarischen Programmausführung mitgeliefert werden.

Bleistiftaufgaben

Aufgabe 1 (*15 Punkte*) Das PARTITION Problem fragt, ob es zu einer gegebenen (Multi-)Menge $\mathcal{S}$ von Zahlen eine Aufteilung in zwei Partitionen A und B gibt, sodass gilt

$$\sum A = \sum B = \frac{1}{2}\sum \mathcal{S}.$$

Für die Menge $\{1, 3, 4, 7, 2, 3\}$ gibt es eine solche Aufteilung: $A = \{7, 3\}$, $B = \{1, 3, 4, 2\}$. PARTITION ist ein NP-vollständiges Problem.

a) Argumentieren Sie, warum PARTITION in NP ist.

b) Das Problem MULTIPROCESSOR SCHEDULING fragt, ob man eine Reihe von Aufgaben $t_1, \ldots, t_k$ mit (ganzzahligen) Längen $\ell(t_1), \ldots, \ell(t_k)$ mit Hilfe von m Prozessoren so abarbeiten kann, dass alle Aufgaben bis zum Zeitpunkt D fertiggestellt sind. Man muss also jeder Aufgabe t_i einen Startzeitpunkt $s(t_i)$ zuordnen, so dass zu keinem Zeitpunkt mehr als m Aufgaben parallel laufen und für alle Aufgaben t_i gilt $s(t_i) + \ell(t_i) \leq D$.

   Argumentieren Sie, warum MULTIPROCESSOR SCHEDULING NP-vollständig ist.

Aufgabe 2 (*15 Punkte*) Wir beschäftigen uns mit dem Rucksackproblem.

a) Argumentieren Sie, warum der Greedy-Algorithmus für das Rucksackproblem, der Gegenstände mit einem hohen Wert pro Gewicht zuerst aufnimmt, zu einem beliebig schlechten Ergebnis führen kann.

b) Nehmen Sie an, es gibt beliebig viele Kopien der Gegenstände. Der Greedy Algorithmus füllt den Rucksack also soweit es geht mit dem Gegenstand mit größten Wert/Gewicht, dann den Rest mit dem zweitwertvollsten Gegenstand usw. Argumentieren Sie, dass der Greedy-Algorithmus höchstens um einen Faktor von zwei schlechtere Ergebnisse liefert als optimal möglich wäre.

Aufgabe 3 (*10 Punkte*) Nehmen Sie an P=NP. Argumentieren Sie, warum das folgende Problem NP-vollständig ist.

Gegeben ein Bit b. Ist $b = 1$?

Python

Aufgabe 4 (*20 Punkte*) Wir wollen ein Programm schreiben, dass das PARTITION Problem löst.

a) Schreiben Sie eine Funktion, die durch Ausprobieren aller Teilmengen eine Lösung für das Problem findet.

b) Angenommen, die Elemente der Menge sind durchnummeriert als $x_1, \ldots, x_n$. Wir definieren uns die folgende Funktion:

$$f(i,s) = \begin{cases} \textit{true} & \text{aus den Elementen } x_1, \ldots, x_i \text{ gibt es eine Menge mit Summe } s \\ \textit{false} & \text{sonst} \end{cases}$$

Wie kann man $f(i+1, s)$ rekursiv ausrechnen?

c) Implementieren Sie eine Funktion, die $f(n, 1/2 \cdot \sum \mathcal{S})$ so ausrechnet, dass jeder Funktionswert nur einmal berechnet wird, d.h. wenn $f(5,7)$ schon einmal im Laufe der Rekursion berechnet wurde, dann wird er kein zweites Mal rekursiv ausgerechnet.

d) Was ist die Laufzeit ihres Programms? Wieso oder wieso nicht haben Sie P=NP bewiesen?