Algorithmen und Berechnungskomplexität I WS 15/16

Universität Bonn, Institut für Informatik, Abteilung I

4. Aufgabenblatt zur Vorlesung **Abgabe: 17.11.** (12^{30})

Aufgabe 13: O-Notation (4 Punkte)

1. Zeigen Sie, dass aus $f(n) \in O(g(n))$ und $g(n) \in O(h(n))$ folgt, dass $f(n) \in O(h(n))$.

2. Geben Sie zwei Funktionen $f(n)$ und $g(n)$ an, sodass weder $f(n) \in O(g(n))$ noch $g(n) \in O(f(n))$ gilt.

3. Ähnlich zur in der Vorlesung definierten O- bzw. Ω-Notation wird die Klein-o-Notation definiert:

$$f(n) \in o(g(n)) \Leftrightarrow \begin{array}{l} \text{zu jedem } C > 0 \text{ existiert ein } n_0 > 0, \\ \text{sodass } f(n) \leq Cg(n) \text{ für alle } n > n_0. \end{array}$$

   Zeigen Sie, dass aus $f(n) \in o(g(n))$ folgt, dass $f(n) \in O(g(n))$. Zeigen Sie außerdem, dass die Umkehrung im Allgemeinen nicht gilt.

Aufgabe 14: Rekursionsgleichungen (4 Punkte)

Lösen Sie die folgende Rekursionsgleichung exakt für $n > 1$:

$$\begin{aligned} S(1) &= 1 \\ S(n) &= \sum_{i=1}^{n-1} i \cdot S(i) \qquad \text{für } n > 1. \end{aligned}$$

Aufgabe 15: Das Rucksackproblem (4 Punkte)

Wir betrachten das Rucksackproblem. Gegeben sind 5 Gegenstände

$$\{a_1 = (1, 4), a_2 = (3, 5), a_3 = (4, 6), a_4 = (6, 9), a_5 = (7, 10)\}.$$

Der erste Wert bezeichnet das Gewicht, der Zweite den Wert der Gegenstände; das Gewicht von a_1 ist beispielsweise 1 und dessen Wert ist 4.

Bestimmen Sie für einen Rucksack mit Kapazität 10 eine optimale Lösung für das Rucksackproblem. Geben Sie an, welche Gegenstände den Rucksack optimal füllen. Stellen Sie hierzu, wie in der Vorlesung besprochen, eine Tabelle von Teillösungen auf um dieses Rucksackproblem mit Hilfe der Dynamischen Programmierung zu lösen. Die Tabelle soll sowohl den für die Teilergebnisse jeweils den erzielten Maximalwert als auch die verwendeten Gegenstände beinhalten.

Aufgabe 16: Dynamische Programmierung (4 Punkte)

Gegeben sei eine Familie M von n Münzen mit Werten $v_1, v_2, ..., v_n \in \mathbb{N}$. Sie sollen nun einen bestimmten Betrag S mit Hilfe dieser Münzen zusammensetzen, d.h eine Teilmenge $U \subseteq 1, 2, ..., n$ finden, so dass $S = \sum_{i \in U} v_i$. Jede Münze darf dabei nur einmal verwendet werden.

Geben Sie ein Verfahren an, das mittels Dynamischer Programmierung die minimale Anzahl an Münzen aus M bestimmt, die dazu nötig ist.