Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Institut für Theoretische Informatik
Peter Widmayer
Sandro Montanari
Tobias Pröger

10. April 2013

Datenstrukturen & Algorithmen Programmieraufgabe 7 FS 13

In dieser Aufgabe soll ein Algorithmus der dynamischen Programmierung zur Lösung des *Rucksackproblems* implementiert werden. Für dieses Problem ist neben einer Menge $S = \{1, ..., n\}$ von n Objekten mit den Werten $v_1, ..., v_n \in \mathbb{N}$ und den Gewichten $w_1, ..., w_n \in \mathbb{N}$ noch eine Gewichtsschranke $W \in \mathbb{N}$ gegeben. Eine *Bepackung* ist eine Menge $I \subseteq S$ mit $\sum_{i \in I} w_i \leq W$, und sie besitzt den Wert $\sum_{i \in I} v_i$. Das Ziel ist es, eine Bepackung I mit maximalem Wert zu finden.

Dazu benutzt der Algorithmus eine Tabelle $A[\cdot, \cdot]$ mit $n + 1$ Zeilen und $W + 1$ Spalten. Für $0 \leq i \leq n$ und $0 \leq j \leq W$ repräsentiert der Eintrag $A[i, j]$ den Wert einer optimalen Bepackung, die nur die ersten i Objekte $\{1, ..., i\}$ verwenden darf und höchstens Gewicht j besitzt. Für $i = 0$ wird die leere Menge betrachtet. Daher setzen wir $A[0, j] = 0$ für jedes j, $0 \leq j \leq W$ (werden keine Objekte benutzt, dann ist der maximale Wert jeder Bepackung 0). Ausserdem setzen wir $A[i, 0] = 0$ für jedes i, $1 \leq i \leq n$ (wenn das Maximalgewicht 0 beträgt, dann ist der Wert jeder Bepackung erneut 0). Die verbleibenden Einträge können wie folgt berechnet werden:

$$A[i, j] = \begin{cases} A[i-1, j-w_i] + v_i & \text{falls } w_i \leq j \wedge A[i-1, j-w_i] + v_i \geq A[i-1, j] \\ A[i-1, j] & \text{ansonsten} \end{cases} \quad (1)$$

Nachdem die Tabelle ausgefüllt wurde, enthält der Eintrag $A[n, W]$ genau den Wert einer optimalen Bepackung. Von dort aus kann die optimale Bepackung selbst durch *Backtracking* ermittelt werden. Wenn $w_n \leq W$ gilt und zusätzlich $A[n, W] = A[n-1, W-w_n] + v_n$ erfüllt ist, dann benutzen wir das Objekt n und fahren mit dem Eintrag $A[n-1, W-w_n]$ fort. Ansonsten benutzen wir das Objekt n nicht und fahren mit $A[n-1, W]$ fort. Dieses Vorgehen wird beendet, wenn alle Zeilen von $A[\cdot, \cdot]$ verarbeitet wurden.

Untenstehend findet sich das Beispiel der Tabelle $A[i, j]$, wenn die Objekte $\{1, 2, 3\}$ und eine Gewichtsschranke von $W = 5$ benutzt werden.

	1	2	3
v_i	1	3	2
w_i	2	2	1

$A[i,j]$	**0**	**1**	**2**	**3**	**4**	**5**
$\emptyset$	0	0	0	0	0	0
$\{1\}$	0	0	1	1	1	1
$\{1,2\}$	0	0	3	3	4	4
$\{1,2,3\}$	0	2	3	5	5	6

(2)

Eingabe Die erste Zeile der Eingabe enthält lediglich die Anzahl t der Testinstanzen. Danach folgen genau drei Zeilen pro Testinstanz. Die erste Zeile enthält (in dieser Reihenfolge) die Werte n und W mit $n, W \in \mathbb{N}$, $1 \leq n \leq 30$ and $1 \leq W \leq 200$. Die zweite Zeile enthält die Werte $v_1, ..., v_n$, und die dritte Zeile enthält die Gewichte $w_1, ..., w_n$. Für jedes Objekt $i \in \{1, ..., n\}$ sind $1 \leq v_i \leq 1000$ und $1 \leq w_i \leq 20$.

Ausgabe Für jede Testinstanz soll lediglich eine Zeile ausgegeben werden. Sie enthält den Wert einer optimalen Bepackung und die Objekte einer optimalen Bepackung. Um Mehrdeutigkeiten zu vermeiden, sollen diese Objekte aufsteigend sortiert ausgegeben werden.

Beispiel

Eingabe:

```
2
2 3
1 1
2 5
3 5
1 3 2
2 2 1
```

Ausgabe:

```
1 1
6 1 2 3
```

Abgabe: Bis Mittwoch, den 17. April 2013.