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

10th April 2013

Datenstrukturen & Algorithmen Programming Exercise 7 FS 13

In this exercise we are going to implement a dynamic programming algorithm for solving the *Knapsack* problem. For this problem, a set $S = \{1, ..., n\}$ of n objects with values $v_1, ..., v_n \in \mathbb{N}$ and weights $w_1, ..., w_n \in \mathbb{N}$ is provided along with a weight limit $W \in \mathbb{N}$. A subset $I \subseteq S$ with $\sum_{i \in I} w_i \leq W$ is called *packing* and has the value $\sum_{i \in I} v_i$. The goal is to find a packing I with maximum value.

The algorithm creates and fills a table $A[\cdot, \cdot]$ with $n+1$ rows and $W+1$ columns. For $0 \leq i \leq n$ and $0 \leq j \leq W$, an entry $A[i, j]$ represents the value of an optimum packing that uses only the first i objects $\{1, ..., i\}$ and has a weight of at most j. Note that for $i = 0$, the set with no objects is considered. Thus, we set $A[0, j] = 0$ for every j, $0 \leq j \leq W$ (when no objects are used, the maximum value of any packing is 0). Furthermore we set $A[i, 0] = 0$ for every i, $1 \leq i \leq n$ (if the weight limit is 0, the maximum value of any packing again is 0). The other entries can now be computed as follows:

$$A[i,j] = \begin{cases} A[i-1, j-w_i] + v_i & \text{if } w_i \leq j \ \wedge \ A[i-1, j-w_i] + v_i \geq A[i-1, j] \\ A[i-1, j] & \text{otherwise} \end{cases} \tag{1}$$

After the table has been filled, the entry $A[n, W]$ contains the value of an optimal packing. From there we can reconstruct the optimal packing using *backtracking*. If $w_n \leq W$ and $A[n, W] = A[n-1, W-w_n] + v_n$, we use the object n and continue with the entry $A[n-1, W-w_n]$. Otherwise, we do not use the object n and continue with $A[n-1, W]$. We stop when all the rows of $A[\cdot, \cdot]$ have been processed.

Below, an example of the table $A[i, j]$ is shown when 3 objects $\{1, 2, 3\}$ and a weight limit $W = 5$ are used.

	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)

Input The first line contains only the number t of test instances. After that, we have exactly three lines per test instance. The first line contains the values n and W in this order, with $n, W \in \mathbb{N}$, $1 \leq n \leq 30$ and $1 \leq W \leq 200$. The second line contains the values $v_1, ..., v_n$, and the third line the weights $w_1, ..., w_n$. For each object $i \in \{1, ..., n\}$, we have $1 \leq v_i \leq 1000$ and $1 \leq w_i \leq 20$.

Output For every test instance we output only one line. This line contains the value of an optimum packing and the objects that are part of the optimum packing computed by the above algorithm. To avoid ambiguity, the indices must be sorted in ascending order.

Example

Input:

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

Output:

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

Hand-in: until Wednesday, 17th April 2013.