Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Department Informatik
Markus Püschel David Steurer Peter Widmayer
Chih-Hung Liu
Stefano Leucci

Datenstrukturen & Algorithmen Blatt P11 HS 17

Solution for Exercise P11.1 *Flea Market.*

The problem can be solved by using a dynamic programming algorithm. For $i = 0, \dots, n$, $s = 0, \dots, S$, and $w = 0, \dots, W$ we define $\mathrm{OPT}[i, s, w]$ as the maximum amount of Flops that can be earned from the sale when (i) the sold items must be chosen among the first i items, (ii) the total surface of the selected items is at least s, and (iii) the total weight of the selected items is at most w. If there is no way to satisfy the above constraints, then we let $\mathrm{OPT}[i, s, w]$ be equal to a sufficiently small value that we denote by $-\infty$.

Clearly $\mathrm{OPT}[0, 0, w] = 0 \; \forall w = 0, \dots, W$ and $\mathrm{OPT}[0, s, w] = -\infty \; \forall s = 1, \dots, S \; \forall w = 0, \dots, W$ as the set of available items to choose from is empty.

Consider now on a generic $\mathrm{OPT}[i, s, w]$ with $i > 0$. Notice that an optimal solution either includes item i or it does not. If it does, then $w_i \geq w$ and the number of earned flops is exactly p_i plus the maximum amount of flops that can be earned with the remaining $i-1$ items provided that they free a surface of at least $\max\{0, s - s_i\}$ and have a total weight of at most $w - w_i$. If it does not, then the number of earned flops is exactly the same that can be earned by only considering the first $i - 1$ items. In formulas:

$$\mathrm{OPT}[i, s, w] = \begin{cases} \mathrm{OPT}[i-1, s, w] & \text{if } w_i > w \\ \max\left\{\mathrm{OPT}[i-1, s, w], \mathrm{OPT}[i-1, \max\{0, s - s_i\}, w - w_i]\right\} & \text{if } w_i \leq w \end{cases}$$

Since each $\mathrm{OPT}[i, s, w]$ can be computed in constant time (by considering the values $\mathrm{OPT}[i, s, w]$ in increasing order of i), the overall time required to solve the problem is $O(n \cdot S \cdot W)$. The value of the optimal solution to the input instance is exactly $\mathrm{OPT}[n, S, W]$.

Solution for Exercise P11.2 *Tree Rotations.*

The following is one possible pseudocode for right tree rotations (left rotations are symmetric).

Algorithm: RightRotation(v)

```
u ← v.leftChild
u.parent ← v.parent

if v.parent ≠ null then
    if v.parent.leftChild=v then u.parent.leftChild← u      // v was a left child
    else u.parent.rightChild← u                              // v was a right child
else
    root ← u                                                 // v was the root of the tree

v.leftChild ← u.rightChild
if v.leftChild ≠ null then v.leftChild.parent← v             // u had a right child
u.rightChild← v
v.parent← u
```