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

20. November 2017

Datenstrukturen & Algorithmen Blatt P11 HS 17

Hand-in: Bis Sonntag, 17. Dezember 2017, 23:59 Uhr via Online Judge (nur Source Code). Fragen zur Aufgabenstellung oder Übersetzung werden wie üblich im Forum beantwortet.

Exercise P11.1 *Flea Market.*

You have the bad habit of leaving items on the floor of your basement, which is now completely full. You decide to clean it up by selling some of the items in the flea market of the city of Algo. There are n items in your basement and the i-th item occupies a surface of s_i m^2, weighs $w_i \geq 1$ Grahams (the weight unit of the city of Algo), and can be sold for a price of p_i Flops (the currency of the city of Algo). You want to free an area of least S m^2 from your basement but you can only carry at most W Grahams to the flea market.

Your task is to design an algorithm that computes the maximum amount of Flops that you can earn from the sale (subject to the above conditions).

Input The input consists of a set of instances, or *test-cases*, of the previous problem. The first line of the input contains the number T of test-cases. The first line of each test-case contains the three positive integers n, S and W. The next n lines each describe one item: the i-th line contains the three integers s_i, w_i and p_i.

Output The output consists of T lines, each containing a single integer. The i-th line is the answer to the i-th test-case, i.e., it contains the total value V of the items to sell at the flea market. More precisely, $V = \max_{X \in \mathcal{I}} \sum_{i \in X} p_i$ where $\mathcal{I}$ contains all the sets of items that have a total area of at least S and a total weight of at most W, i.e., $\mathcal{I} = \{X \subseteq \{1, \ldots, n\} : \sum_{i \in X} s_i \geq S \text{ and } \sum_{i \in X} w_i \leq W\}$.

Grading You get 3 bonus points if your program works for all inputs. Your algorithm should require $O(n \cdot S \cdot W)$ time (with reasonable hidden constants). Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=18997&problem=DA17P4.7`. The enrollment password is "`asymptotic`".

Subtask 1: You get 1 point (out of the 3 total points) if you program correctly solves instances in which $S = 1$ and $s_i = 1$ for every $i = 1, \ldots, n$.

Example

Input:

```
1
6 10 12
1 4 10
3 5 8
7 10 5
5 2 3
1 1 1
3 4 2
```

Output:

```
22
```

Notes For this exercise we provide an archive on the lecture website containing a program template that will load the input and write the output for you. The archive also contains additional test cases (which differ from the ones used for grading). Importing or using classes that are not in `java.lang.*` is **not allowed** (with the exception of the already imported `java.util.Scanner` class).

Exercise P11.2 *Tree Rotations.*

Your task is to implement simple tree rotations in an AVL tree supporting insertions. Most of the implementation of the AVL tree is already provided by the template (reading the input, inserting a new element, output).

The tree is stored as a collection of `Node` objects. Each `Node` object v has five fields:

<`parent`, `leftChild`, `rightChild`, `value`, `balanceFactor`>

A pointer `parent` to the parent of v in the tree (or `null` if v is the root of the tree), two pointers `leftChild` and `rightChild` to the left and right child of v, respectively (or `null` if no such child exists), the integer `value` associated with v, and the `balanceFactor` of v (i.e., the height of the subtree rooted at the right child of v minus the height of the subtree rooted at the left child of v).

Notice that for every pointer, `leftChild`, `rightChild`, or `parent`, from one vertex v to another vertex u, there is a corresponding pointer from u to v. The provided AVL tree implementation also contains an additional pointer, named `root`, to the current root node of the tree (`root` is `null` when the tree is empty).

Your task is to complete the code of the functions `rotateLeft(Node v)` and `rotateRight(Node v)` to perform a single left and a single right tree rotation around v, respectively. Here v is the deepest vertex in the AVL tree that is temporarily unbalanced as a consequence of an insertion operation.

To solve the task it is sufficient to replace the `TODO` commentS in the template with your Java code. We strongly recommend not to change the rest of the template (even though it is not forbidden). You should only take care of updating the pointers `leftChild`, `rightChild`, and `parent` that change as a result of the rotation, as well as the pointer `root` (if needed). Your solution *should not* modify any `balanceFactor` field: they are already updated elsewhere in the provided template code. Note that the functions `rotateLeft(Node v)` and `rotateRight(Node v)` keep track of the number of left and right rotations performed: these numbers are the output of your program and they are needed by the judge to verify the correctness of your solution. We strongly advise against changing them.

The numbers being inserted are unique integers between 0 and 1 000 000.

Input The input consists of a set of instances, or *test-cases*. The first line of the input contains the number T of test-cases. Each test case consists of two lines. The first line contains n, i.e., the number of integers to be inserted into a (initially empty) AVL tree. The second line contains the n integers to be inserted, separated by spaces.

Output The output consists of T lines, each containing a single integer. The i-th line corresponds to the i-th test-case and contains two integers separated by a space. The first (resp. second) integer is the total number of left (resp. right) rotations performed as a consequence of the n insertions operations.

Grading You get 3 bonus points if your program works for all inputs. The time required to perform a single rotation should be constant. Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=18997&problem=DA17P4.8`. The enrollment password is "`asymptotic`".

Example

Input:

```
1
9
3 1 6 9 4 8 10 11 12
```

Output:

```
2 0
```

Notes For this exercise we provide an archive on the lecture website containing a program template that will load the input and write the output for you. The template already implements most of the functionality. Your task is to complete the code of the functions `rotateLeft(Node v)` and `rotateRight(Node v)`.

The archive also contains additional test cases (which differ from the ones used for grading). Importing or using classes that are not in `java.lang.*` is **not allowed** (with the exception of the already imported `java.util.Scanner` class).