Eidgenössische
Technische Hochschule
Zürich

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

Departement of Computer Science
Markus Püschel, David Steurer
Karel Kubicek, Johannes Lengler

14 October 2019

Algorithms & Data Structures Programming task 7 HS 19

The solutions for this sheet need to be submitted to the judge by Sunday, 10 November, 23:59:59.

Exercise 7.1 *Heap and Heap sort* **(2 bonus point)**.

Your task is to implement some operations for the data structure maxheap. Those are `isMaxHeap` and `heapSort`. To realise them, you also need to implement the auxiliary functions `heapify`, and `siftDown`. If you wish, you can also implement `siftUp`. All auxiliary functions are tested in the Eclipse project, but not by Judge. For debugging, the tests visualize the heap in Graphviz format, to render it, you can use offline `dot`, or online `http://www.webgraphviz.com/` tools.

`isMaxHeap` checks if an array is a maxheap, i.e., if it satisfies the maxheap condition everywhere.

`heapSort` sorts the array via the heapsort algorithm.

`heapify` builds a maxheap from the input array.

`siftDown` checks whether the element with a given index has a too high index to satisfy the heap property. If not, then it restores the heap property for this element.

`siftUp` checks whether the element with a given index has a too low index to satisfy the heap property. If not, then it restores the heap property for this element.

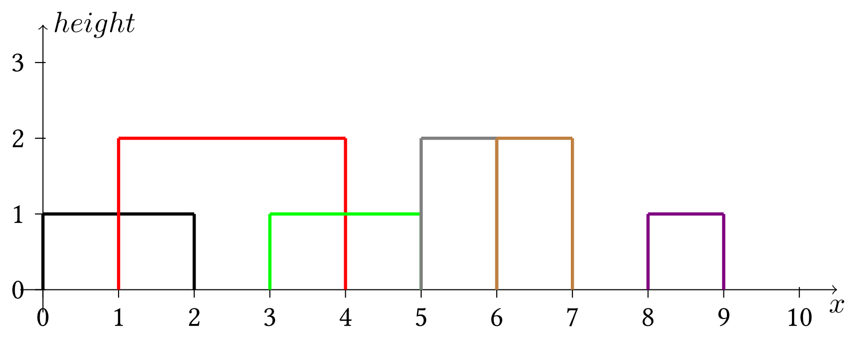
Please find a more detailed description of the task in the `Main.java` file. The methods that you have to implement are denoted by `TODO` and description of the task. The data format is described by the method `Main.read`. The Eclipse project also contains a set of basic tests, execute them by running `JUnitTest.java`.

You get one point for each passing test set. The test set `basic` is smaller, so it does not test time complexity. That is doing test set `complex`.

Submission: Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=28784&problem=AD19H7P1`. The enrollment password is "`asymptotic`".

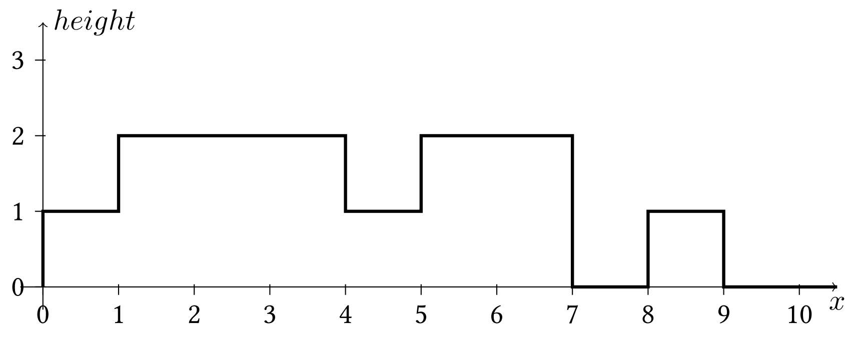
Exercise 7.2 *Computing overlapping buildings silhouette* (**2 bonus point**).

In this exercise, you are going to develop an algorithm to compute the silhouette of a given set of rectangular buildings. Every building is determined by its beginning and end on the x axis, and its height. The input has the form of a set of triples of (beginning, height, end) $\in \mathbb{N}^3$. A city is a set of buildings. For example,



is represented by input: `0, 1, 2, 1, 2, 4, 3, 1, 5, 5, 2, 6, 6, 2, 7, 8, 1, 9`. Note that the order of buildings does not have to satisfy any order.

For a given set of buildings, your goal is to compute the overall silhouette of the whole city. Graphically, it is:



which we represent as silhouette $\underline{0}, \overline{1}, \underline{1}, \overline{2}, \underline{4}, \overline{1}, \underline{5}, \overline{2}, \underline{7}, \overline{0}, \underline{8}, \overline{1}, \underline{9}, \overline{0}$. This array consists of pairs of x axis positions where the height of silhouette changes (underscored) and the height itself (overscored). Every silhouette is therefore terminated by height 0.

Please find more detailed description in the `Main.java` file. The task is to implement methods `Main.evaluate`. The data format is described by method `Main.read`. The Eclipse project also contains a set of basic tests, run them by running `JUnitTest.java`.

The judge is again testing two test sets. To pass `simple`, it is fine if your solution works for all buildings of height 1. But you have to do so in $\mathcal{O}(n \log(n))$. The test set `generic` requires you to evaluate an arbitrary silhouette in $\mathcal{O}(n \log(n))$. Hint (in white text):

Submission: Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=28784&problem=AD19H7P2`. The enrollment password is "`asymptotic`".