Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Dr. Alexander Souza
Thomas Janson

WS 2011/12
Sheet 05

Algorithm Theory

Exercise 1 (Amortized Analysis) [Points: 5]

Recall that a binary min-heap guarantees that the depth is $\Theta(\log n)$ whenever n elements are stored in it. Thus it supports the operations insert and deletemin in $\mathcal{O}(\log n)$ worst case time (for recovery of the heap property as in the lecture).
Define a potential function Φ yielding

- amortized cost $\mathcal{O}(\log n)$ for insert and
- amortized cost $\mathcal{O}(1)$ for deletemin.

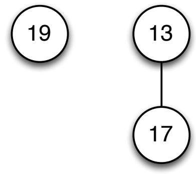
Exercise 2 (Amortized Costs) [Points: 5]

Assume you go skiing K many times, where $K \in \mathbb{N}$ is unknown. Renting skiing equipment for once costs one unit and buying the skiing equipment costs k units.

1. Design an algorithm which never spends more than two times the minimal cost.
2. Give an amortized analysis of your algorithm.

Exercise 3 (Binomial queues) [Points: 5]

Consider the binomial queue given below, and execute the following operations.



1. Q.insert(11), Q.insert(5), Q.insert(18), Q.insert(12), Q.insert(7), Q.decreaseKey(18,1) and Q.deletemin().
2. By using the child-sibling-representation, represent the binomial queue resulting from previous question 1.

Exercise 4 (Priority queues) [Points: 5]

Consider a comparison-based priority queue, i.e. a priority queue where the keys are stored using "$\leq$" comparisons. Prove that inserting n elements in the priority queue and extracting them again with deletemin takes at least $\Omega(n \log n)$.