Universität Freiburg
Institut für Informatik
Dr. F. Wei
T. Hornung

Georges-Köhler-Allee, Geb. 051
D-79110 Freiburg i. Br.
Freiburg, 15. Juni 2009

Theory I - Exercise sheet 4

Submission due **Tuesday, June 23**

Note: This sheet contains only 3 exercises.

Exercise 1: (6 points) Amortized analysis

A *binary counter* counts up in the binary system. We assume that the cost for flipping one bit is 1. For example, the increment operation counting from 101110 to 101111 has cost 1 (only the last bit has to be changed); the next increment operation, counting from 101111 to 110000, has cost 5 since it requires the change of the last 5 bits. It is obvious that in the worst case all bits required to display the next counter reading need to be flipped, hence the worst-case cost for an increment operation would be in $O(w)$, where w is the width of the counter, i.e. the number of bits.
By *aggregate analysis* (or any other "Amortized Worst Case" method), prove that the *amortized cost* of the increment operation is in $O(1)$, i.e. bounded by a constant (independent of the counter width).
Hints: For a sequence of n increment operations - starting from the initial counter reading 0 - show that the sum of the required costs is bounded by $c \cdot n$, where c is a constant. You may assume that n is a power of 2, i.e. $n = 2^k$ for a non-negative integer k.
Definition: The **Aggregate Method**: Find the worst case running time $T(n)$ for a sequence of n operations. The amortized cost of each operation is $T(n)/n$.

Exercise 2: (6 points) Amortized analysis
Show with the help of the *aggregate analysis* (or any other "Amortized Worst Case" method), that the amortized costs of an insertion step is constant, if in an initial empty dynamic table n keys are inserted. For the expansion strategy the size of the table will be increased by $\lceil \frac{n}{2} \rceil$ fields with n the size of the full table before the expansion.
Hints: For a sequence of n insertion operations - starting from inserting into the empty array - show that the sum of the required costs is bounded by $c \cdot n$, where c is a constant. The costs of an insertion step are 1, if the table is not full, the costs are $k + 1$, if the key has to be inserted into a full table with k elements. You may assume that n is a power of 2, i.e. $n = 2^k$ for a non-negative integer k.

Exercise 3: ($4 + 4$ points) Amortized analysis
A FIFO Queue Q (First-In First-Out) has the operations $dequeue(Q)$ and $enqueue(Q, x)$, which extracts an element of the start of the queue respectively adds an element at the end of the queue. Show that it is possible to implement this data structure with two stacks S and S' in such a way that these operations have constant amortized costs.

a) Realize the operations $dequeue(Q)$ and $enqueue(Q, x)$ by using the following stack operations:
$push(Q, x)$: insert x into the stack Q.
$pop(Q)$: extract the last inserted object from the stack and deliver it as a result.
$empty$: true iff Q is empty.
Please write the method in Pseudo- or Java-Code.

b) Show that the amortized costs are constant by using the aggregate method (or any other "Amortized Worst Case" method).