Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
H. Ghodselahi, Y. Maus November 24, 2016

Algorithm Theory, Winter Term 2016/17
Problem Set 4

hand in (hard copy or electronically) by 09:55, Thursday December 8, 2016,
tutorial session will be on December 12, 2016

Exercise 1: Amortization (using Accounting) (8 points)

Suppose we perform a sequence of n operations on an (unknown) data structure in which the i-th operation costs i if i is an exact power of 2, and 1 otherwise.

Use the **accounting** method to determine the amortized cost per operation.

Exercise 2: Amortization (using Potential Function) (8 points)

We are given a data structure $\mathcal{D}$, which supports the operations `put` and `flush`. The operation `put` stores a data item in $\mathcal{D}$ and has a running time of 1. Further, if $\mathcal{D}$ contains $k \geq 0$ items, the operation `flush` deletes $\lceil k/2 \rceil$ of the k data items stored in $\mathcal{D}$ and its running time is equal to k.

Prove that both operations have constant amortized running time by using the **potential function** method.

Exercise 3: Fibonacci Heaps (12 points)

Fibonacci heaps are only efficient in an **amortized** sense. The time to execute a single, individual operation can be large. Show that in the worst case, the `delete-min` and `decrease-key` operations can require time $\Omega(n)$ (for any heap size n).

Hint: *Describe an execution in which there is a delete-min operation that requires linear time. Also, describe an execution in which there is a decrease-key operation that requires linear time.*

Exercise 4: Union-Find (12 points)

Assume that we are given a union-find data structure which is implemented as a disjoint-set forest. In the lecture, we have seen that when using path compression and the union-by-rank heuristics, the total running time of any m operations is $\Theta\left(m \cdot \alpha(m, n)\right)$ (where $\alpha(m, n)$ is the inverse of the Ackermann function and n is the number of `make-set` operations).

We now consider any sequence of m union-find operations, where all the `make-set` and `union` operations appear before any of the `find-set` operations. Let f be the number of `find-set` operations. Show that the total running time of the f `find-set` operations is only $\mathcal{O}(f + n)$ if both path compression and union-by-rank heuristics are used. What happens in the same situation if we use only the path compression heuristic (without union-by-rank)?

Remark: In the union-by-rank heuristic, each tree of the disjoint-forest representation has a rank which is computed as follows. When a tree of size 1 is created in a `make-set` operation, its rank is 0. Further, whenever two trees T_1 and T_2 are merged in a `union` operation, the tree of smaller rank is attached to the tree of larger rank. If T_1 and T_2 have different ranks, the rank of the combined tree is equal to the larger of the two ranks of T_1 and T_2. Otherwise, if they both have the same rank, the rank of the combined tree is the rank of the two trees plus 1.