University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Bamberger, P. Schneider

Algorithm Theory
Exercise Sheet 5

Due: Tuesday, 23rd of November, 2021, 4 pm

Exercise 1: Stack with Backup *(8 Points)*

Consider an (initially empty) stack S where every k operations the content of S is copied for backup purposes which takes time $\Theta(|S|)$.

(a) Given an *arbitrary* series of `push` and `pop` operations on S, what is the amortized cost of an operation? *(2 Points)*

(b) Assume we conduct a series of `push` and `pop` such that the size of S never exceeds N. Assign amortized running times (possibly depending on N and k) to the operations that are as small as possible. Prove your claim. *(6 Points)*

Exercise 2: Dynamic Array with Remove *(12 Points)*

In the lecture we saw a dynamically growing array that implements the `append` operation in amortized $\mathcal{O}(1)$ (reads/writes). For an array of size N that is already full, the `append` operation allocates a new array of size $2N$ ($\beta = 2$) before inserting an element at the first free array entry.

Additionally, we introduce another operation `remove`, which writes `Null` into the last non-empty entry of the dynamic array. However, by appending many elements and subsequently removing most of them, the ratio of unused space can become arbitrarily high. Therefore, when the dynamic array of size N contains $\frac{N}{4}$ or less elements after `remove`, we copy each element into a new array of size $\frac{N}{2}$.

(a) Given an array of size N which has at least $\frac{N}{2}$ elements, show that any series of `remove` operations has an amortized cost of $\mathcal{O}(1)$ (reads/writes) per operation. *(4 Points)*

(b) Use the potential function method to show that any series of `append` and `remove` operations has amortized cost of $\mathcal{O}(1)$ (reads/writes) per operation. Assume that the number of elements n in the array is initially 0 and assume that the array never shrinks below its initial size N_0 (we stop allocating smaller arrays in that case). *(8 Points)*

Remarks: You may assume that N is always a power of two. You may also assume that allocating an empty array is free, only copying elements costs one read and one write for each copied element. If you do part (b) absolutely correctly you automatically receive all points for part (a).