Exercise sheet 1.

Data structures SoSe 2020

László Kozma, Katharina Klost

Due 12:00, May 1st, 2020

Exercise 1 Stacks and variants 3×4 *Points*

Recall the array-based stack implementation from the lecture, with operations *make_stack*, *push*, *pop*, supported in constant amortized time.

(a) A *double-ended queue* ("deque") is a more powerful data structure that allows adding and removing elements at both ends of a list, i.e. it supports the operations *make_deque*, *push-right*, *push-left*, *pop-right*, *pop-left*.

Design a deque that uses memory efficiently and in which all operations take constant amortized time. Analyze the data structure rigorously with any method you prefer. You may re-use the stack construction from the lecture as a black box, or you may use arrays.

(b) Suppose you wanted a stack where all operations take *actual* constant time (as opposed to amortized). Describe how to modify the array-based design to achieve this. (A linked list would also work, but we insist on using arrays.) *Hint:* The only "bad case" was when the array had to be doubled and elements had to be copied. Can you "spread" this work across multiple operations?

(c) The *waste* of a data structure is the difference between the number of memory cells in use and the number n of items stored in the data structure. The stack implementation discussed in the lecture has a waste of $O(n)$. Improve the design to reduce the waste to $O(\sqrt{n})$ at all times. (Make sure you account for all the extra pointers and other bookkeeping you may need in the design.)

Exercise 2 Lower bounds 2×4 *Points*

(a) Recall that if a heap supports one of the operations *insert* and *extract-min* in time $o(\log n)$, the other type of operation must take time $\Omega(\log n)$.

Similarly argue that if *meld* takes time $O(n^{1-\varepsilon})$ for arbitrary constant $\varepsilon > 0$, then *extract-min* must take time $\Omega(\log n)$.

(b) Recall that we showed how to do median filtering on a sequence of length n with window size k in time $O(n \log k)$. Argue that this running time is optimal in both n and k. (Hint: sorting.)

Total: 20 points. Have fun with the solutions!