Exercise sheet 10.

Data structures

SoSe 2020

László Kozma, Katharina Klost

Due 12:00, July 3rd, 2020

Exercise 1 Splay trees *1+4+4 pts*

(a) Try to get some familiarity with the behavior of splay trees. Draw a binary search tree with 5-10 nodes and work out a few splay operations on paper. You can also try some interactive demonstrations of splay tree on the internet. Try to force splay to make costly operations. What happens?

(b) Show that in a splay operation "almost" every node on the search path "roughly" halves its depth. Prove a statement that makes the terms in quotes precise. What happens to the nodes outside the search path?

(c) Consider a "simpler version" of splay, called move-to-root. In move-to-root, instead of the zig-zig and zig-zag operations, we simply rotate the accessed element up using normal rotations until it becomes the root.

Show that move-to-root can be very inefficient: construct an initial tree and an arbitrarily long search sequence that has high cost per search if move-to-root is used instead of splay. Is the choice of initial tree essential in your example?

Exercise 2 Lower bound for dynamic BSTs *4+3 Points*

The goal of this exercise is to show that on *most* search sequences, the $\Theta(\log n)$ cost per search cannot be improved, even if we allow rotations, and even if the search sequence is known in advance.

Consider a search sequence $R = r_1, \ldots, r_n$ with values from $\{1, \ldots, n\}$. *Serving* R means the following: starting from an initial tree T of our choice, for $i = 1, \ldots, n$, we search for r_i in the tree in the usual way (i.e. by moving a pointer from the root to the searched element). After every search we may re-arrange the tree using rotations. The rotations can only be done at the pointer. After every search, the pointer is at the searched element, but we can move the pointer around the tree step-by-step, by moving from the current node to the parent, or to the left/right child. The total cost of serving R is the total number of pointer moves and rotations during this process.

(a) Suppose that R can be served with a total cost c. (Each search has cost at least 1, so $c \geq n$.) Show that the process of serving R can be *encoded* in a binary string of length $O(c)$, so that, from this string, the sequence R can be *reconstructed*.

Hint: Encode explicitly the steps performed in the optimal strategy, as well as the initial tree.

(b) Let $OPT(R)$ denote the smallest possible cost for serving R (i.e. a suitable choice of initial tree and an optimal strategy for pointer moves and rotations that serve R).

Show that there exists a constant $\alpha > 0$, such that we have $OPT(R) > \alpha \cdot n \log n$ for *almost all* possible sequences R. (You can replace *almost all* by a large constant fraction, say 99.9%.)

Hint: Observe that there are n^n possible sequences R. Relate the number of possible sequences and the number of possible encodings.

Exercise 3 Deque sequences *4 Points*

A *deque sequence* is an arbitrarily long sequence of *insert*, *delete*, and *search* operations, such that every *delete* and *search* refers either to the minimum or to the maximum of the currently stored keys, and every *insert* is for a key either smaller than all currently stored keys or larger than all currently stored keys.

Design a data structure based on a binary search tree with rotations (using no extra pointers or annotations) that (starting from an empty tree) can serve an arbitrary deque sequence with constant amortized cost per operation.

Total: 20 points. Have fun with the solutions!