Your name
CS 261, Spring 2021
Homework 9
Due Monday, May 31

1. We have seen that for persistent stacks implemented as singly linked lists, path-copying persistence is a better than fat nodes: it has constant time and space overhead per persistent operation, while the overhead for the flat trees in fat nodes is larger. It is also possible to implement queues (data structures which maintain a sequence of elements, with "enqueue" operations that add them at one end of the sequence and "dequeue" operations that remove them at the other), using singly linked lists, by maintaining pointers to the nodes at both ends of a list, with the dequeue removing an element from the start of the list and the enqueue adding an element to the end of the list. If we want to make a queue, implemented in this way, into a persistent queue, would path-copying or fat nodes be a better choice? Explain your answer.

■

2. Suppose that we have represented a fully persistent binary search tree, with a pointer into a single node of the tree, using zippers. Our representation provides access to this representation via the following operations:

- Test whether the node at the pointer has a left child, a right child, or a parent
- If it has a parent, test whether the node at the pointer is the left or the right child of its parent
- Move the pointer to the left child, the right child, or the parent, and return a new version of the structure representing the tree with the moved pointer. (This operation can be performed by making new copies of two nodes, the ones pointed to before and after the move.)
- Return the key at the current node.

Using only these operations, describe how to implement an operation that moves the pointer to the next key in sorted order in the tree and that returns a new version of the structure representing the tree with the moved pointer. You may describe it either in English or pseudocode, but use a standard style of programming, not functional programming, for your answer. Your solution should detect the case that there is no next key, and in that case it should raise an error condition.

■

3. In the method for point location by persistent search trees, suppose that we relax one of the simplifying assumptions, and allow the boundary between two regions to be vertical, while keeping all of the other simplifying assumptions. How does this relaxation affect the ways that the vertical ordering of regions can change from one slab to the next?

■

4. Describe an API for a fully retroactive queue, like the one for a stack in the "Fully retroactive stack API" slide of the lecture notes.

■