Problem Set 7.5

Data Structures and Algorithms, Fall 2021

Due: November 18, in class.

Problem 1

An ordered stack is a data structure that stores a sequence of items and supports the following operations.

- `OrderedPush`(x) removes all items smaller than x from the beginning of the sequence and then adds x to the beginning of the sequence.
- `Pop`$()$ deletes and returns the first item in the sequence (or `Null` if the sequence is empty).

Suppose we implement an ordered stack with a simple linked list, using the obvious `OrderedPush` and `Pop` algorithms. Prove that if we start with an empty data structure, the amortized cost of each `OrderedPush` or `Pop` operation is $O(1)$.

Problem 2

Recall that a standard FIFO queue maintains a sequence of items subject to the following operations.

- `Push`(x): Add item x to the end of the sequence.
- `Pull`$()$: Remove and return the item at the beginning of the sequence.
- `Size`$()$: Return the current number of items in the sequence.

It is easy to implement a FIFO queue using a doubly-linked list, so that it uses $O(n)$ space (where n is the number of items in the queue) and the worst-case time for each of these operations is $O(1)$.

Consider the following new operation, which removes every tenth element from the queue, starting at the beginning, in $\Theta(n)$ worst-case time.

```
Decimate()
1: n ← Size()
2: for (i ← 0 to n − 1) do
3:     if (i mod 10 == 0) then
4:         Pull()
5:     else
6:         Push(Pull())
```

Prove that in any intermixed sequence of `Push`, `Pull`, and `Decimate` operations, the amortized cost of each operation is $O(1)$.