CpSc 8400: Design and Analysis of Algorithms

Instructor: Dr. Brian Dean — Spring 2014
Webpage: `http://www.cs.clemson.edu/~bcdean/` — TTh 11:00-12:15
Handout 8: Quiz #1. 25 possible points. — Vickery 100

Some of the following questions ask you to design an algorithm. Please do so *clearly* and also *concisely*, in English (pseudocode is generally not necessary). For each algorithm, please also briefly justify its running time. Standard results from class or the textbook can be used as "black boxes" without extra elaboration. Partial credit will be awarded for valuable insight or slightly slow but correct solutions, as long as they are not overly complicated. Unless otherwise stated, use the RAM model of computation. For some problems, overly-complicated solutions will receive fewer points than simple solutions. Be sure to keep an eye on the clock and pace yourself well. Good luck!

1. Common Elements (4 points). You are given two n-element balanced binary search trees (AVL trees) as input. Please describe a fast comparison-based algorithm for deciding whether there is any key that appears in both trees.

2. Recurrences (1+1+1 points). Please give asymptotic $\Theta(\cdot)$ solutions to the following recurrences. As a base case for each one, $T(n) = O(1)$ if $n = O(1)$.

$T(n) = 2T(n/2) + \Theta(\log n)$

$T(n) = 3T(n/2) + \Theta(\log n)$

$T(n) = 2T(n/3) + \Theta(\log n)$

3. Min-Quack (8 points). Recall from lecture that we can easily build a *min-stack* supporting the operations *push*, *pop*, and *find-min* (which reports but does not remove the minimum element in structure) all in $O(1)$ worst-case time. In this problem, we use min-stacks to build a more powerful data structure which we call a *min-quack*.

We all know that a queue is a sequential data structure supporting insertion at one end and removal from the other, and a stack is a sequence supporting insertion and removal from the same end. A *quack*, as you might guess from its amusing name, is a combination of the two: it maintains a sequence of elements and supports the ability to insert or delete at *either* end. Accordingly, a quack must support the operations *insert-left*, *insert-right*, *delete-left*, *delete-right* which insert and delete elements on the left and right sides of the sequence. For example, if the quack holds the sequence "1 2 3 4 5" and we call *insert-right*(6), the quack would then hold "1 2 3 4 5 6". If we then called *delete-left*, the quack would hold "2 3 4 5 6".

We want to build a *min-quack*, supporting also a *find-min* operation. Please show how to use a pair of min-stacks to implement a *min-quack* where *insert-left*, *insert-right*, *delete-left*, *delete-right* and *find-min* all run in $O(1)$ amortized time. Use potential functions in your amortized analysis for full credit.

4. Min-Quack, Part II (4 points). Please design a min-quack data structure (defined in the previous problem) where all five fundamental operations (*insert-left*, *insert-right*, *delete-left*, *delete-right* and *find-min*) take $O(\log n)$ worst-case time.

5. Housing Prices (6 points). There are n houses located at various locations along a one-dimensional street (think of this as a number line). For each house $i = 1 \dots n$, you are told its location x_i, as well as its price p_i. The numbers $x_1 \dots x_n$ are integers in the range $1 \dots n^2$, and the prices $p_1 \dots p_n$ are integers. A house i is *overvalued* if there exists another house of at most half its price within some distance D – that is, if there exists another house j such that $p_j \leq p_i/2$ and $|x_i - x_j| \leq D$. Given D, Please give a fast algorithm for counting the number of overvalued houses.

This page can be used for scratch work.