Problem Set #3

5 questions

1  
point

1.

Suppose you implement the functionality of a priority queue using a *sorted* array (e.g., from biggest to smallest). What is the worst-case running time of Insert and Extract-Min, respectively? (Assume that you have a large enough array to accommodate the Insertions that you face.)

* Θ(n) and Θ(n)
* Θ(logn) and Θ(1)
* Θ(1) and Θ(n)
* Θ(n) and Θ(1)

Because insertion in an array would mean creating a copy of an array, hence O(n). Extract min can be done in O(1) if we keep track of where the “good” data starts, like “mbuf”

1  
point

2.

Suppose you implement the functionality of a priority queue using an *unsorted* array. What is the worst-case running time of Insert and Extract-Min, respectively? (Assume that you have a large enough array to accommodate the Insertions that you face.)

* Θ(n) and Θ(n)
* Θ(1) and Θ(n)
* Θ(1) and Θ(logn)
* Θ(n) and Θ(1)

1  
point

3.

You are given a heap with n elements that supports Insert and Extract-Min. Which of the following tasks can you achieve in O(logn) time?

* Find the largest element stored in the heap.
* Find the fifth-smallest element stored in the heap.
* Find the median of the elements stored in the heap.
* None of these.

1  
point

4.

You are given a binary tree (via a pointer to its root) with n nodes. As in lecture, let size(x) denote the number of nodes in the subtree rooted at the node x. How much time is necessary and sufficient to compute size(x) for every node x of the tree?

* Θ(height)
* Θ(n2)
* Θ(nlogn)
* Θ(n)

**Correct**

For the lower bound, note that a linear number of quantities need to be computed. For the upper bound, recursively compute the sizes of the left and right subtrees, and use the formula size(x) = 1 + size(y) + size(z) from lecture

1  
point

5.

Suppose we relax the third invariant of red-black trees to the property that there are no *three* reds in a row. That is, if a node and its parent are both red, then both of its children must be black. Call these *relaxed* red-black trees. Which of the following statements is *not* true?

* The height of every relaxed red-black tree with n nodes is O(logn).
* There is a relaxed red-black tree that is not also a red-black tree.
* Every red-black tree is also a relaxed red-black tree.
* Every binary search tree can be turned into a relaxed red-black tree (via some coloring of the nodes as black or red).

**Correct**

A chain with four nodes is a counterexample.