Queued

CSC 252 – Sections A and B

**Assigned:** 14 August 2014 **Due**: 21 August 2014

**Overview**

We’ve already seen a couple of good applications for a Priority Queue: triage at a Hospital, Huffman tree building. Now, it’s high time we built one!

**Outline**

For this lab, you will implement two versions of PriorityQueue, one heap-based and the other AVL-based. You will prove that they work by switching out Java’s priority queue for your own in your Huffman compressor. The interface for both solutions will be as follows:

**public** **boolean** offer(T data);

**public boolean** poll(T data);

**public boolean** peek(T data);

*HeapBasedPriorityQueue.java*

The first class will be called **HeapBasedPriorityQueue** and will use an array-based min-heap as the internal structure.

This priority queue will have a constructor:

**public** HeapBasedPriorityQueue(**int** initialSize);

The actual initial size should be the ideal size for an essentially complete heap that may contain initialSize values (think about powers of 2).

*AvlBasedPriorityQueue.java*

The second class will be called **AvlBasedPriorityQueue** and will use an AVL tree as the internal structure. The tree should rebalance as necessary.

**Thinking Questions**

When should your tree rebalance? Is there a way to get it to rebalance on its way back up from insertion or removal? If not, how does that affect performance?

What makes AVL a poor choice for a priority queue?

Will your implementation improve or degrade the performance of your Huffman tree? By how much?

*After* you have completed your priority queues, find the source code for Java’s priority queue. What do they do? Do you notice any interesting optimizations?

**What to Study**

Read Sections 6.3 and 6.4 in the Levitin text.

**What to Hand In**

Pass-offs will be done via one or more unit tests; however, please still check in your source code to your git repo along with a jar of your compiled files. Anything that doesn’t compile after the due date will receive **1** point off their total lab grade.

You should be checking in at least **HeapBasedPriorityQueue.java** and **AvlBasedPriorityQueue.java**.

**How You Will Be Graded**

Of the points possible, **30%** is for a verbal defense. **30%** is for a working heap-based priority queue, **30%** is for a working AVL-based priority queue, and **10%** for correctly balancing the AVL tree.

**Stretch Goals**

* Take an image compression from Huffman w/ Java Priority Queue and correctly decompress it successfully with Huffman w/ Your Heap Priority Queue. **(+5%)**
* If you didn’t do it already, now is your chance to complete the Hospital/Priority Queue simulation. **(+5%)**
* Implement a 2-3 tree as a priority queue. **(+10%)**
* Implement the heap-based priority queue as a min-max heap. **(+10%)** See <http://cg.scs.carleton.ca/~morin/teaching/5408/refs/minmax.pdf> for an outline.