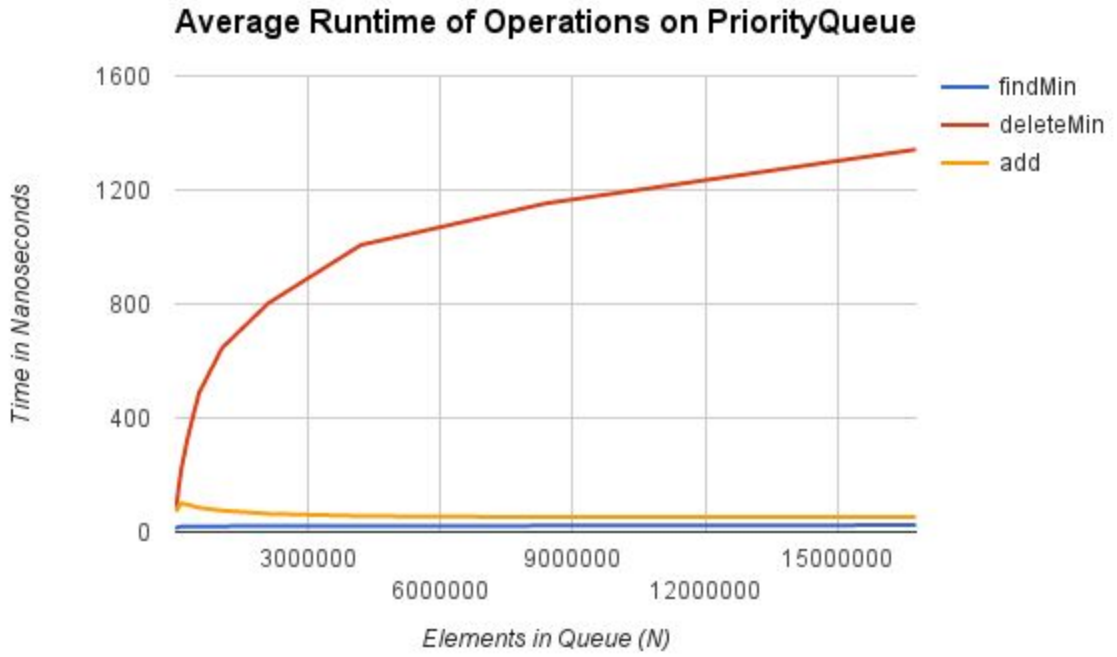
Savannah Simmons u1086770

**1. Design and conduct an experiment to assess the running-time efficiency of your priority queue. Carefully describe your experiment, so that anyone reading this document could replicate your results. Plot the results of your experiment.Since the organization of your plot(s) is not specified here, the labels and titles of your plots(s), as well as, your interpretation of the plots is critical.**

To assess the efficiency of my priority queue, I timed the three main operations to access and modify it: *findMin()*, *DeleteMin()*, and *add()*. On priority queues of size 1024 to 16777216 (increasing by powers of 2), I averaged how long it took to perform each operation once over 1,000,000 iterations.




The above plot shows the average time it took to perform each of the three operations once at varying sizes of N.

**2. What is the cost of each priority queue operation (in Big-O notation)? Does your implementation perform as you expected? (Be sure to explain how you made these determinations.)**

This is probably the most perfect graph I've had all semester. As expected, *findMin* and *add* clearly perform at O(C). Calling *findMin* simply returns the first item in the backing array, so this should certainly perform at constant time. As for *add*, while there is an amount of percolating happening which accounts for the slight increase in time over *findMin*, analysis shows that regardless of the size of the queue, *add* still performs an average of only 2.6 comparisons.

This is different from *deleteMin*, which carries a higher likelihood of having to iterate through the entire height of the tree, or logN. So it makes sense that the above plot of *deleteMin* demonstrates a clear logarithmic trend, indicating that *deleteMin* does indeed have time complexity O(logN).

**3. Briefly describe at least one important application for a priority queue.(You may consider a priority queue implemented using any of the three versions of a binary heap that we have studied: min, max, and min-max)**

One important application of a priority queue that we've touched on before is in implementing Dijkstra's algorithm. In order to find the path with the lowest cost between two nodes in a graph, the different possible paths are explored by prioritizing the in-progress paths by the lowest current cost.

**4. How many hours did you spend on this assignment?**

I spent around 6-7 hours on this assignment.