Brayden Wright
u0895942

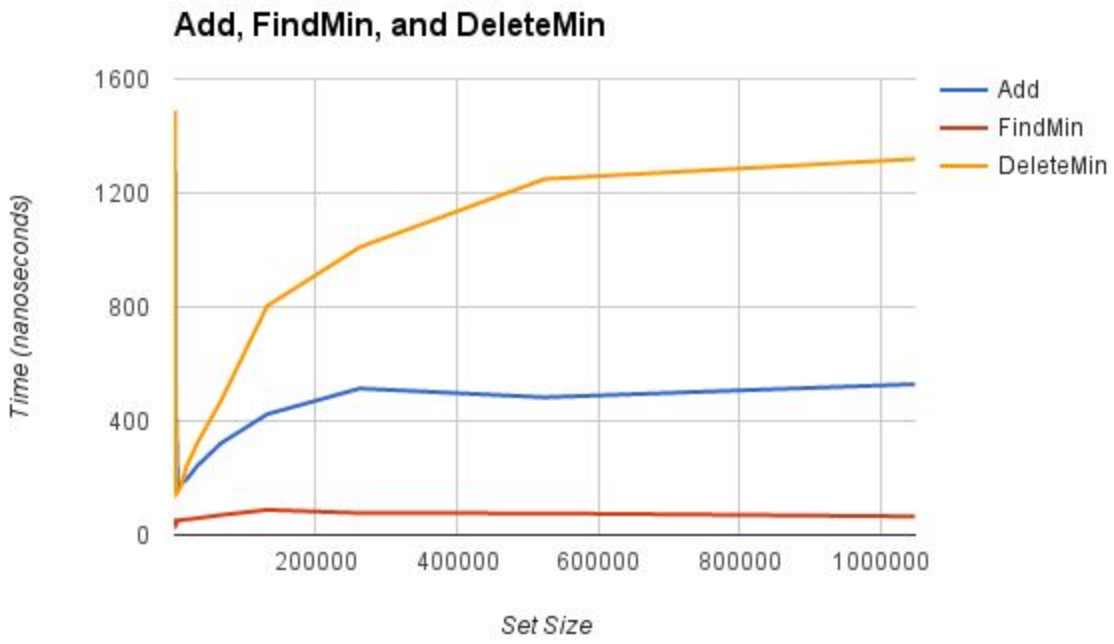
1. **Design and conduct an experiment to assess the running time efficiency of your priority queue. Plot the and analyze the results of your experiment.**

My experiment tests the performance of each of the three main methods we had to create for a given set size: add, findMin, and deleteMin. To do so, each method is individually called 1,000 times per set. The sets tested against are ordered integers and of set sizes 2^8 (256) up to 2^{20} (1,048,576).

The add method did worse than I expected. I expected it to run in constant time, O(C), but it ended up running close to O(log n), but still better than deleteMin. I believe the add method runs this gave this result due to all the array resizing it had to do since it would not be percolating up due to everything being added in order.

The deleteMin method ran about as expected and appears to be O(log n). It should run in O(log n) since it should only have to percolate down one branch at most, e.g. halving the data every time the new root node is moved down.

The findMin method also ran about as expected and appears to run in constant time, O(C). This should always be constant time because all this method does is grab the first item in the underlying array, which should always be the minimum item in a min-heap binary heap.



Brayden Wright
u0895942

2. **What is the cost of each priority queue operation? Does your implementation perform as you expected? Explain how you came to your conclusions.**

I sort of touched upon this in the previous question, so there's a little bit of insight there too. The add operation should run in O(C) time on average; I think my implementation runs in constant time as well if we were to assume that the array never had to be resized, otherwise my implementation did a bit worse. The findMin operation should also run in O(C) in all cases, since all one has to do is check the first item, e.g. array[0]. My implementation does perform as expected in that instance. The deleteMin operation runs in about O(log n) time on average, which mine also appears to do.

3. **Briefly describe at least one important application for a priority queue.**

In an IT environment, a priority queue can be used to track issues that need to be looked at. The higher the severity/priority, the sooner it *should* get pulled out of the issue tracker and the lower the severity/priority, the later it should get pulled out. Using a priority queue in this way would allow those monitoring the issues to tackle problems in a reasonable order instead of picking and choosing just the easy problems to solve. Now, that isn't how it always works, but it would probably be beneficial.

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

I think I spend around six hours on this assignment. That includes both the programming part and completing the analysis.