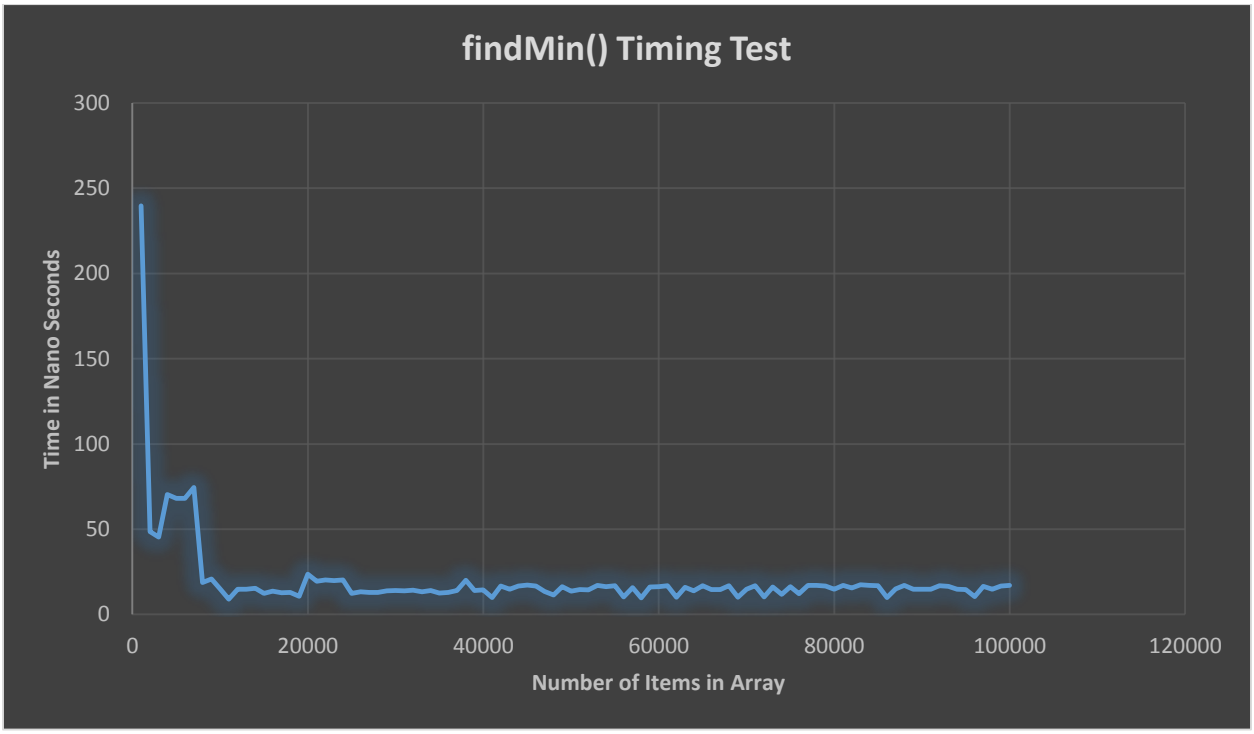
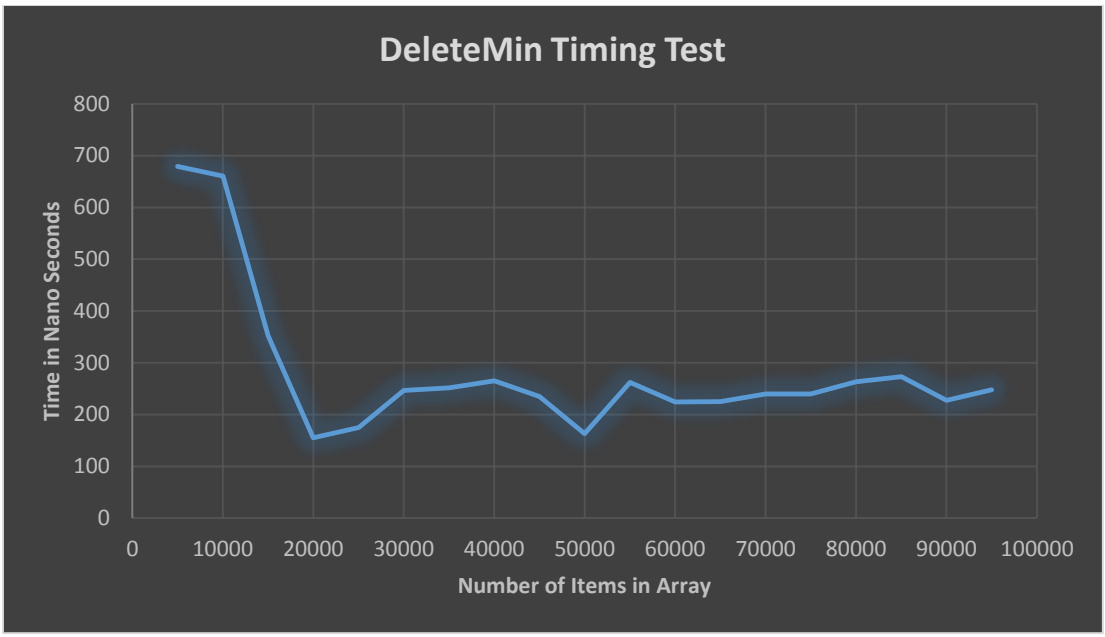
Ryan Bolander u0016911

When you are satisfied that your program is correct, write a brief analysis document. The analysis document is 20% of your Assignment 11 grade. Ensure that your analysis document addresses the following.

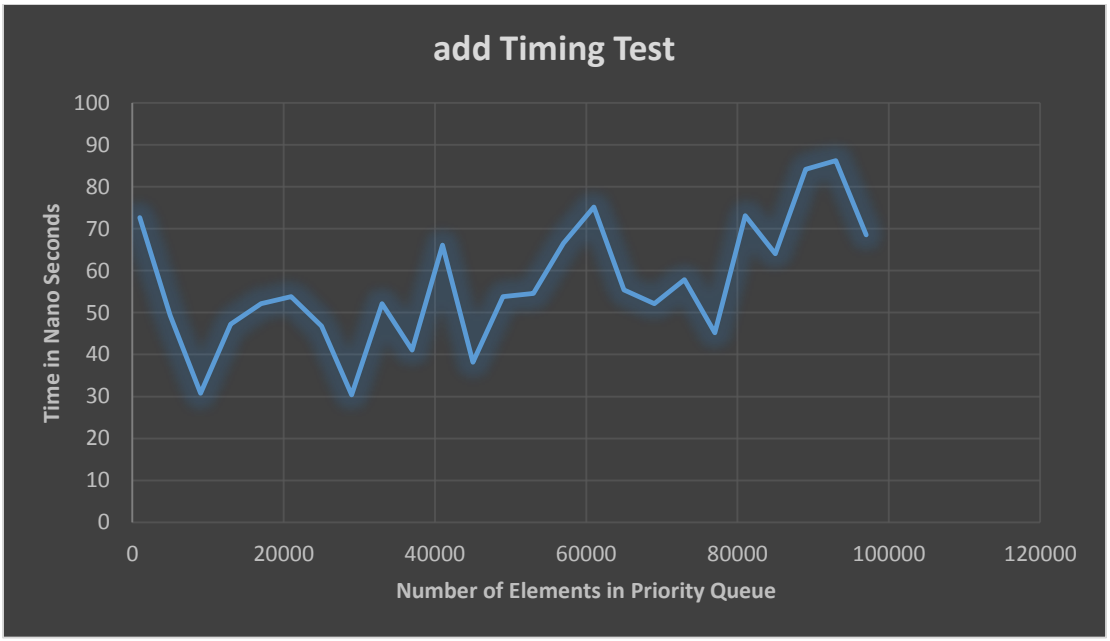
1. Design and conduct an experiment to assess the running-time efficiency of your priority queue. Carefully describe your experiment, so that anyone reading this document could replicate your results. Plot the results of your experiment. Since the organization of your plot(s) is not specified here, the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



Above is a timing test of the FindMin() for the priority queue. I tested find min on various size arrays and timing in nano seconds. I created a priority queue graph using a for loop to fill the priority queue with numbers. I then timed that priority queue by finding the min over an iteration count of 5000 and average out the results of each iteration. The above graph appears to be in constant time so O(c).



I created a random list and then added all those elements into my priority queue and then timed the deletion of the element. At the start of each run of the for loop I would clear the priority queue and then re-add all the items from the shuffled list and then time the remove of the same item again. I did this a number of times in order to get a good average of the time it would take to remove elements. I increased the size of the list in an outer for loop each time to time various size lists. I made sure to clear the list of elements before adding another set and shuffling it to be used with adding into the priority queue. My timing isn't a clear cut O(log n) but it does appear that is what it is. It jumps up to around 200 nanoseconds fairly quickly and then slowly moves up from there. The various could be due to the fact that it has the ability to terminate early so some deletions of the min may be a lot quicker than others.



For the add timing test I again created a random list that I then used to add to the priority queue by using a loop. Once that priority queue was created I then added an element that fell in the middle of the number of elements in the priority queue. So just (number of elements / 2) was added. I timed this add 1000 times to get a good average and the above plot is what I got for a graph. Again, it is hard to see the O(log n) nature of the graph but it does appear to be there. However, due to the ability of the add to terminate early you can get variations in that timing.

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.)

The cost of findMin() should be constant time O(c). My implementation performs as expected.

The cost of deleteMin() should be O(log n). My Implementation performs as expected with the added benefit of terminating early if need be.

The cost of add() should be O(log n). My implementation does appear to perform O(log n) but it is hard to tell due to the ability of it to terminate early.

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)

A priority queue can be used for something like a triage in the hospital. By using a weight on the injury of the patient and their need to be seen as well as when they got there. You can prioritize who gets seen first by adding them to the priority queue. Then based on the injury and when they got there you can pull them off the priority queue in a manner that is beneficial to their health.

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

4 hours.

Programming partners are encouraged to collaborate on the answers to these questions. However, each partner must write and submit his/her own solutions.