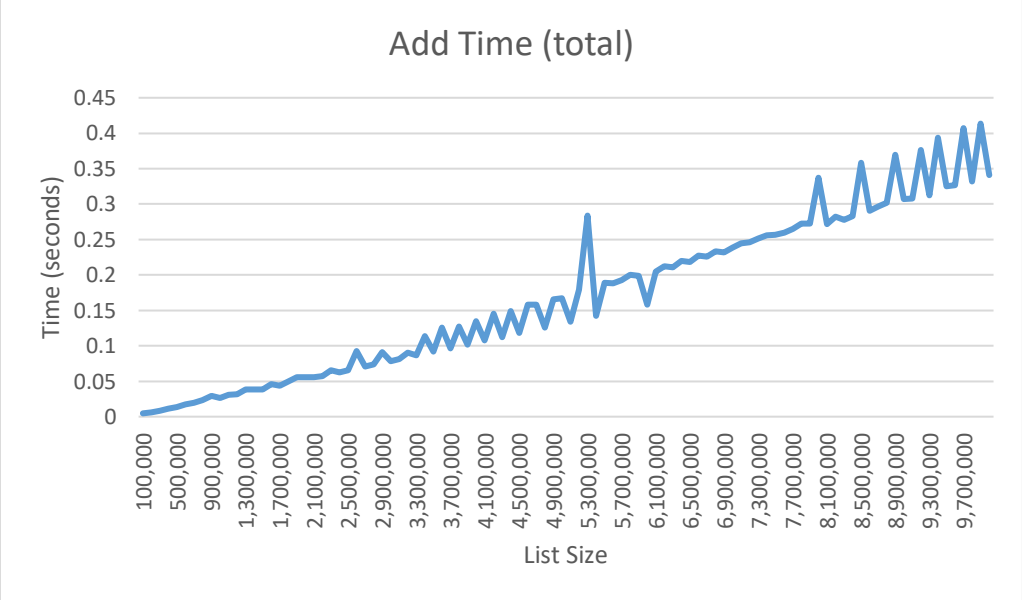
Nathan Page
CS 2420 – 001
Dr. Miriah Meyer
November 16, 2016

Assignment 11 – Analysis

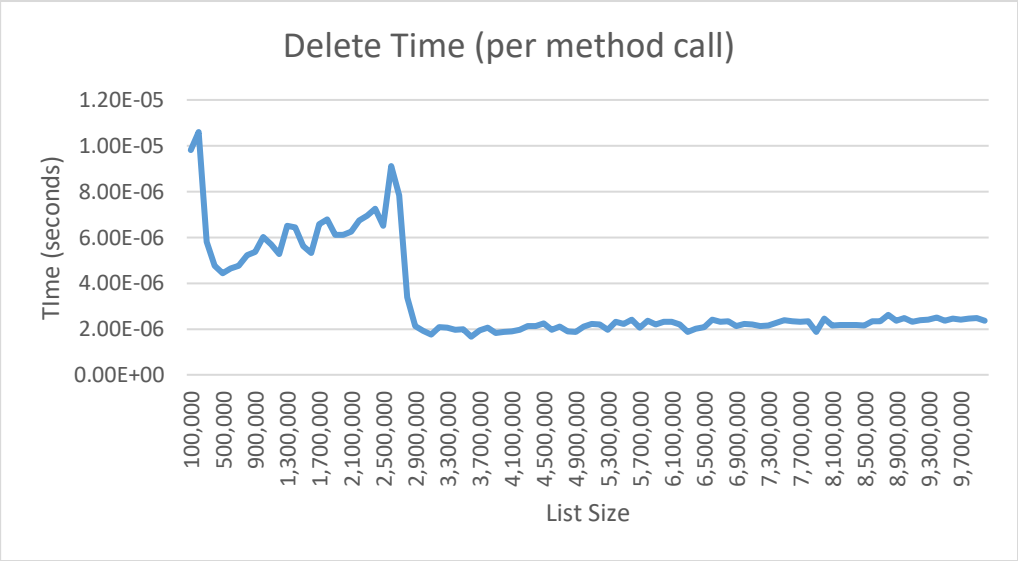
Design and conduct an experiment to assess the running-time efficiency of your priority queue. Carefully describe your experiment so that anyone reading this document could replicate your results. Plot the result of your experiment.

To test the efficiency of this program three tests were performed, one each on add, delete and findMin. For the add method the program the time was taken for how long it takes to add all items in the list, this means that a linear graph as shown below actually represents a constant time operation that is being run for list sizes increasing at a linear rate.

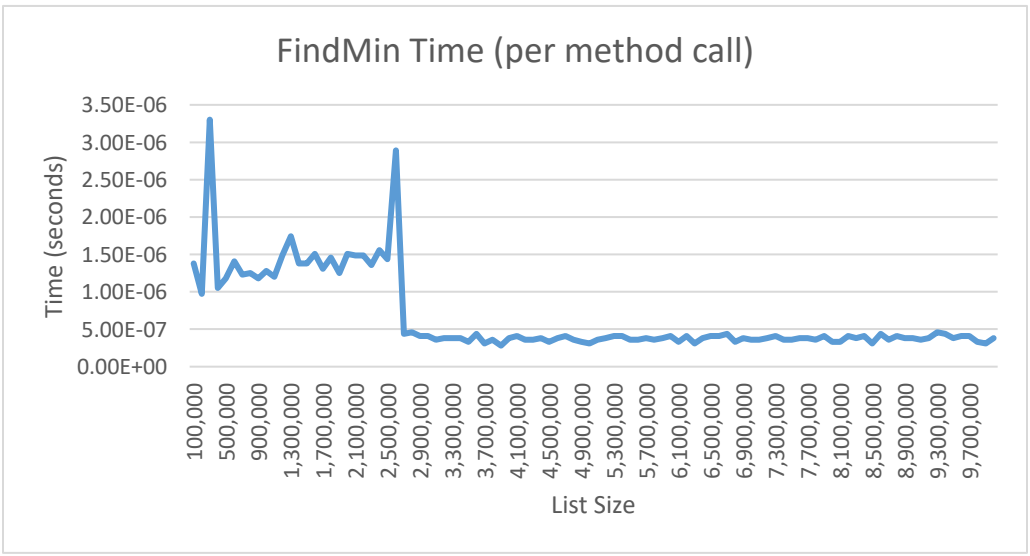


Nathan Page
CS 2420 – 001
Dr. Miriah Meyer
November 16, 2016

The delete and findMin methods show a constant time with a slightly longer time for smaller list sizes, likely due to junk collection occurring from the previous test run. Despite this, all tests ran for less than 12 microseconds for the delete method and less than 3.5 microseconds for the findMin method.



All tests used random numbers and used increasing list sizes, the results were averaged out over ten runs to reduce fluctuation.



Nathan Page
CS 2420 – 001
Dr. Miriah Meyer
November 16, 2016

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 add method has a complexity of O(c) since it only performs a swap 1.6 times on average which does not depend on the size of the list. The delete method has at worst a complexity of O(logN) since after the root node is deleted it is replaced with the last node which will usually end up in the last or second to last level, in this case the tree does not have many levels and therefore the graph looks closer to O(c). The find min should only return the root of the tree so the complexity will be O(c). In all cases the general complexity should be somewhere close to a constant.

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 area that a max queue could possibly be used would be in a multicore processor. Each thread will have a priority depending on how important the task is to the functioning of the computer and the cores need to be synchronized which means that when a program is broken up for parallel processing the remaining threads would need to be at the highest priority to prevent a delay caused by cores processing threads out of order. This may not be the way a system is implemented or this may be handled by hardware rather than software but this data structure could theoretically be a good option.

Another implementation this data structure would work well for would be a site that provides answers that are moved up in the list based on popularity. This would be a site such as Stack Exchange, Reddit or yahoo answers. This could also be a news website or app where the most popular stories are the ones that appear on the front page.

How many hours did you spend on this assignment?

This assignment required 4 hours for coding, 1 hour for testing, 3 hours for timing and debugging and 3 hours for this analysis for a total of 11 hours.