Andrew Katsanevas

CS 2420

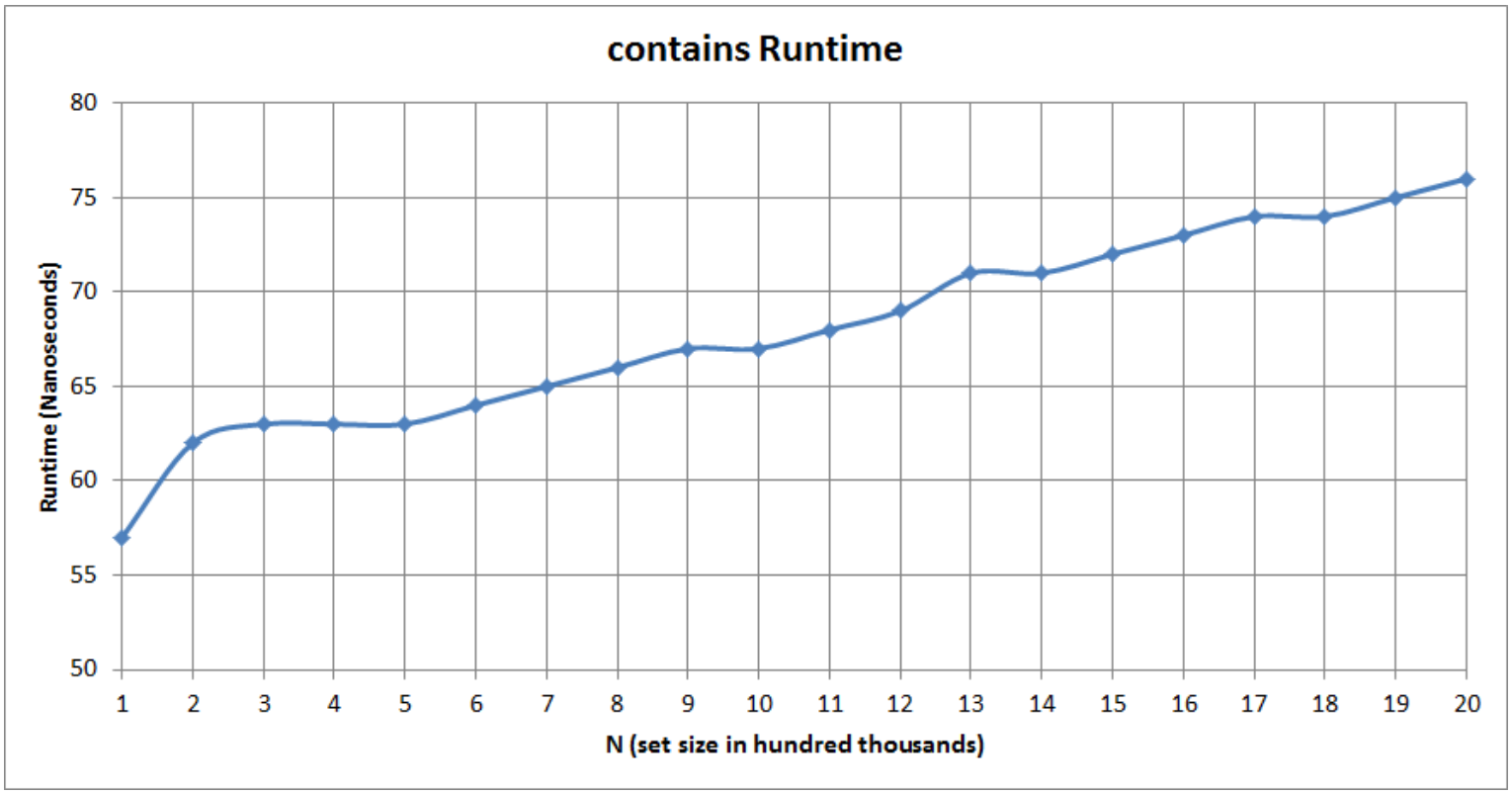
2/5/2015

Assignment #3 Analysis Document

My programming partner was Dyllon Gagnier again. I submitted the code this time. We switched roles probably about once every 30-45 minutes. I think we switched at a good rate. Each person had time to get plenty done before switching. Dyllon was a good partner once again, and we will most likely work on the next assignment.

If I had backed MySortedSet with a Java list instead of an array, some things would be different. The grow function would not be necessary, since Java lists can grow dynamically. I wouldn't have to keep track of a length variable, since the list's actual length would be correct. Using a list, I could also insert elements with the given add function. Similarly, remove and contains would also already have functions I could use. This would be easier to program since much of it is already done, and would take a significantly shorter amount of time to write. I suspect the running time of either option would be quite similar, assuming that I programmed methods like add and contains as efficiently as they are coded in Java's lists.

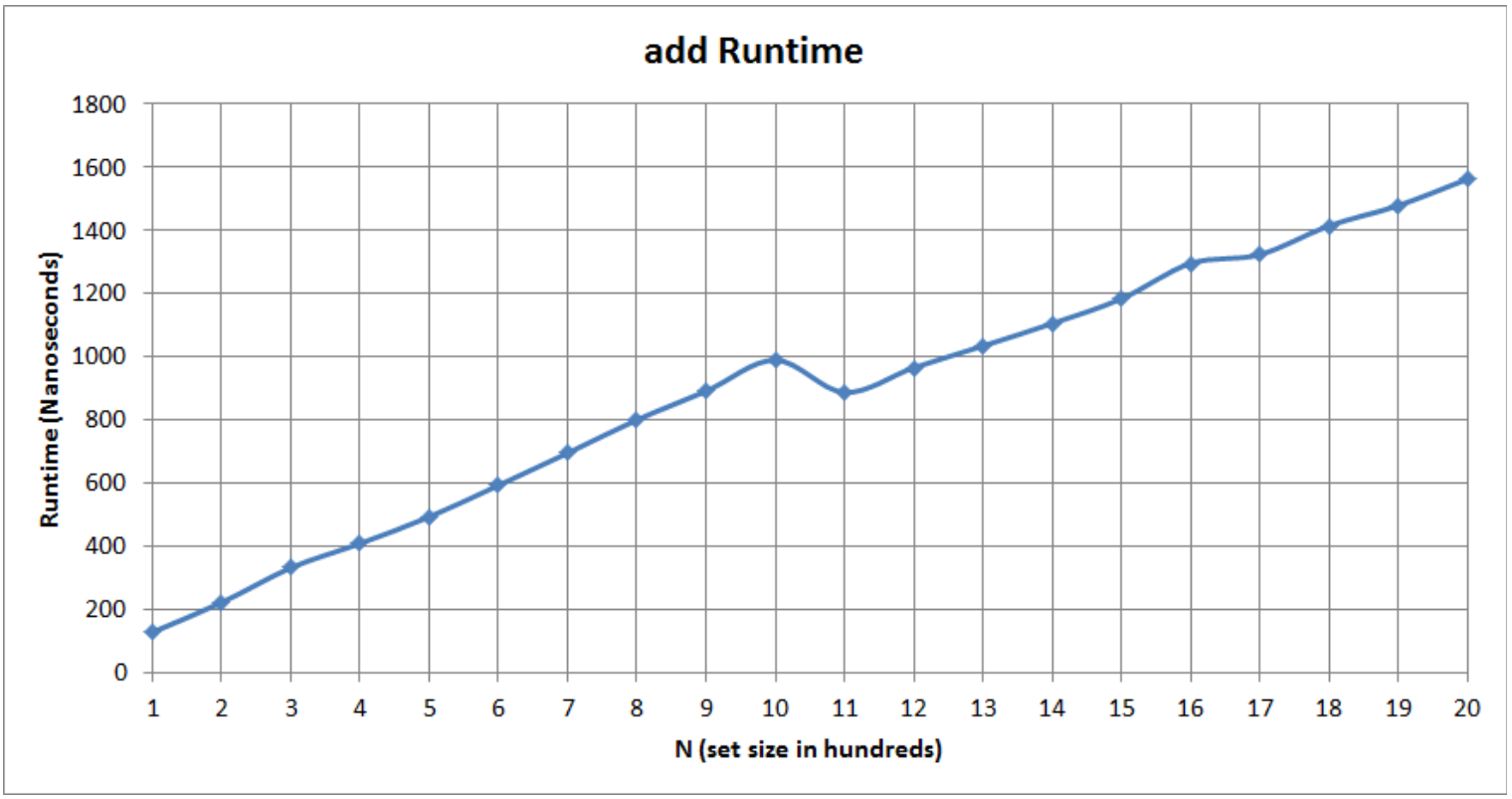
I would expect the Big-O behavior of my contains method to be O(logN), since it's essentially a binary search. The runtime of contains at many different set sizes is graphed on the next page. As expected, the runtime increases approximately logarithmically.



The add method uses the contains method and a binary search, then adds the given element at the correct index, shifting any elements above it. The worst case for this method is adding an element that comes before the rest in the set. After finding the index with binary searching, it would have to shift over every element in the list to insert a new one at the beginning. The worst case complexity would be O(N + logN), since we must add the time to shift all elements to the time of binary search.

The plot on the next page shows the runtime of the add method on different-sized sets. For the sake of consistency, an integer 10% of the way into each set is being repeatedly removed and added. Only the adding counts towards the time,

not the removing. Since this is a fairly bad case scenario, the plot appears somewhat linear. The best case scenario would be logarithmic, O(logN + c), since it would be a binary search then adding an element onto the end.



We spent around 8-10 hours on this assignment.