Eli Dalton
2-5-2015
CS 2420
Assignment 3 Analysis

When you are satisfied that your program is correct, write a brief analysis document. The analysis document is 30% of your Assignment 3 grade. Ensure that your analysis document addresses the following.

1. Who is your programming partner? Which of you submitted the source code of your program?

    Braden Davis was my programming partner for assignment 3.

2. How often did you and your programming partner switch roles? Would you have preferred to switch less/more often? Why or why not?

    We didn't switch roles very often because the work was done on Braden's computer since mine does not have java 7 working on it yet. I like typing because I feel likeI learn better when I do, but I'm okay with not doing most of the typing. We'll switch roles more often when I can actually use my computer to write the assignments.

3. Evaluate your programming partner. Do you plan to work with this person again?

    Probably, but our work schedules do not coincide as much as we would like and finding time to work together is somewhat limited.

4. If you had backed the sorted set with a Java List instead of a basic array, summarize the main points in which your implementation would have differed. Do you expect that using a Java List would have more or less efficient and why? (Consider efficiency both in running time and in program development time.)

    First off, if we had used java's built in ArrayList, we would have ensured that everything worked correctly. Our program still needs some tweaks, but we couldn't get the help we needed. Second, Java's ArrayList is the optimal version of our own SortedSet, so everything would have run smoother, operations would have run quicker as well, and sorting would be more efficient than a binary search because those who programmed java would use something less basic and more efficient. There is also an EnsureCapacity function built in to Java's ArrayList that allows the list to grow appropriately before adding any items so the list could grow to the appropriate size with one growth rather than multiple steps.

5. What do you expect the Big-O behavior of MySortedSet's contains method to be and why?

    Contains() should have complexity of O(logN) because it uses a binary search. The list is always being divided into two parts and in the best case scenario, contains could find the item right at the middle entry making it O(1). However, this is not likely and on average, we will still have O(logN) complexity.

6. Plot the running time of MySortedSet's contains method for sets of sizes 100000 to 2000000 by steps of 100000. Use the timing techniques demonstrated in Lab 1. Be sure to choose a large enough value of timesToLoop to get a reasonable average of running times. Include your plot in your analysis

document. Does the growth rate of these running times match the Big-oh behavior you predicted in question 5?

We are still working on this part and will update our submission once it is completed if still possible. Theoretically, the growth rate should be as stated in 5, but can't say until we have the data complete.

7. Consider your add method. For an element not already contained in the set, how long does it take to locate the correct position at which to insert the element? Create a plot of running times. Pay close attention to the problem size for which you are collecting running times. Beware that if you simply add N items, the size of the sorted set is always changing. A good strategy is to fill a sorted set with N items and time how long it takes to add one additional item. To do this repeatedly (i.e., timesToLoop), remove the item and add it again, being careful not to include the time required to call remove() in your total. In the worst-case, how much time does it take to locate the position to add an element (give your answer using Big-oh)?

The correct position can always be found with at most LogN steps. Each time we look at the list, we look at the middle item and see if our item to be added goes in the half of the list above or below the middle. We then check the item one away from the middle in that direction to see if the item fits between the two. Then the upper or lower half of the list is viewed in the same method as the entire list was in the previous step. At most, we have 2 * LogN comparisons to make before we find the right location for the item, and can insert it into the list, but 2 * LogN is still O(LogN) complexity because we ignore integer coefficients.

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

We spent at least 14 hours on this assignment. Every day we both had off of work we worked on it from when Braden got off of work until it was time to retire for the evening.

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

Upload your document (.pdf only!) to the Assignment 3 page by 11:59pm on February 5.