Qiaofeng Wang
U0861228

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?

Guanlin Chen. I submitted code.

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

Twice, prefer more. Because in this way I can practice more.

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

He is good. I'd like to.

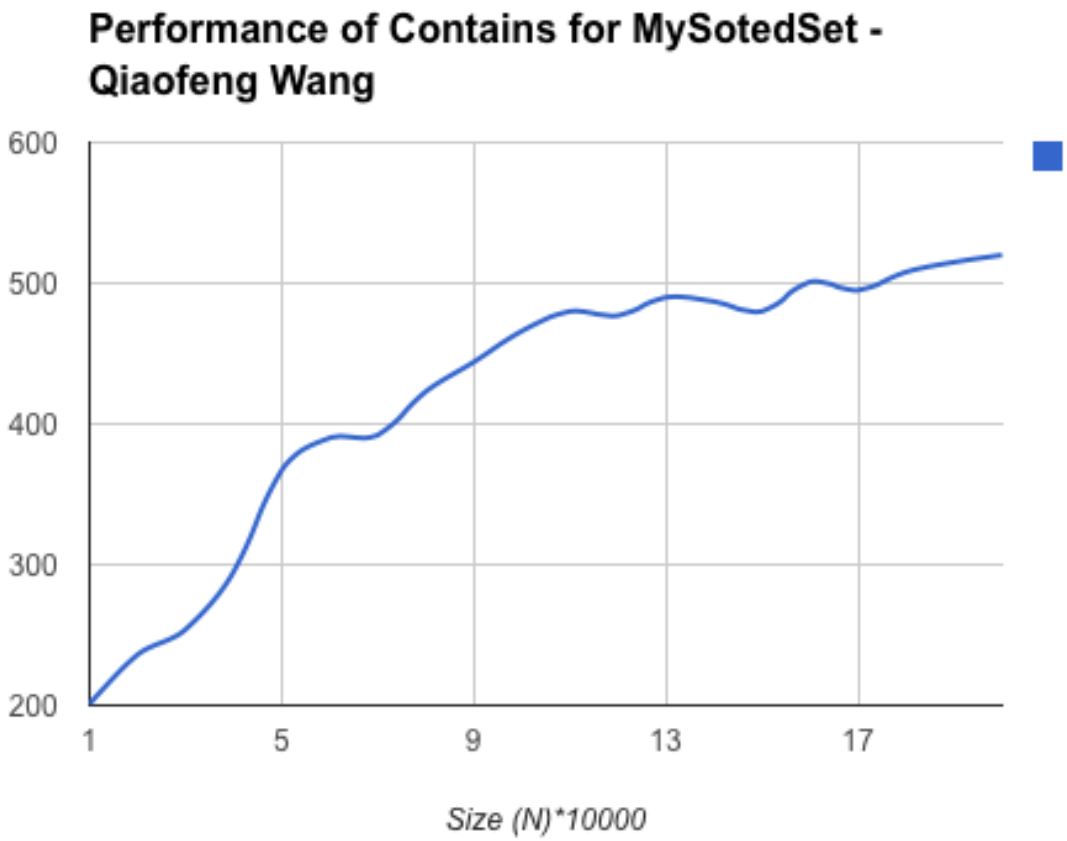
4. If you had backed the sorted set with a Java List instead of a basic array, summarize the main points in which your implementation would have differed. Do you expect that using a Java List would have more or less efficient and why? (Consider efficiency both in running time and in program development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

1. Java list do not need to grow the size. For instance, when basic array is full, it has to grow the array for a new item inserted.
2. Java list do not have index. So contains, add and remove methods have to search from the beginning of the list to the end to check the item. Not like basic array, it can use index to save time and operations.
For Java list, it will make program development time better. But for contains, add, remove methods the Java list will perform O(N).
For basic array, it may be worse in program development time. But it performs a O(logN) in contains method. It is better n running time.

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

O(logN). Because it uses binary search. Every time it check a item in the array, it does not check the next one. Since the array is sorted, it goes 1/2 of the array to check another value. In this way, the running time will be O(logN), which is better than O(N).

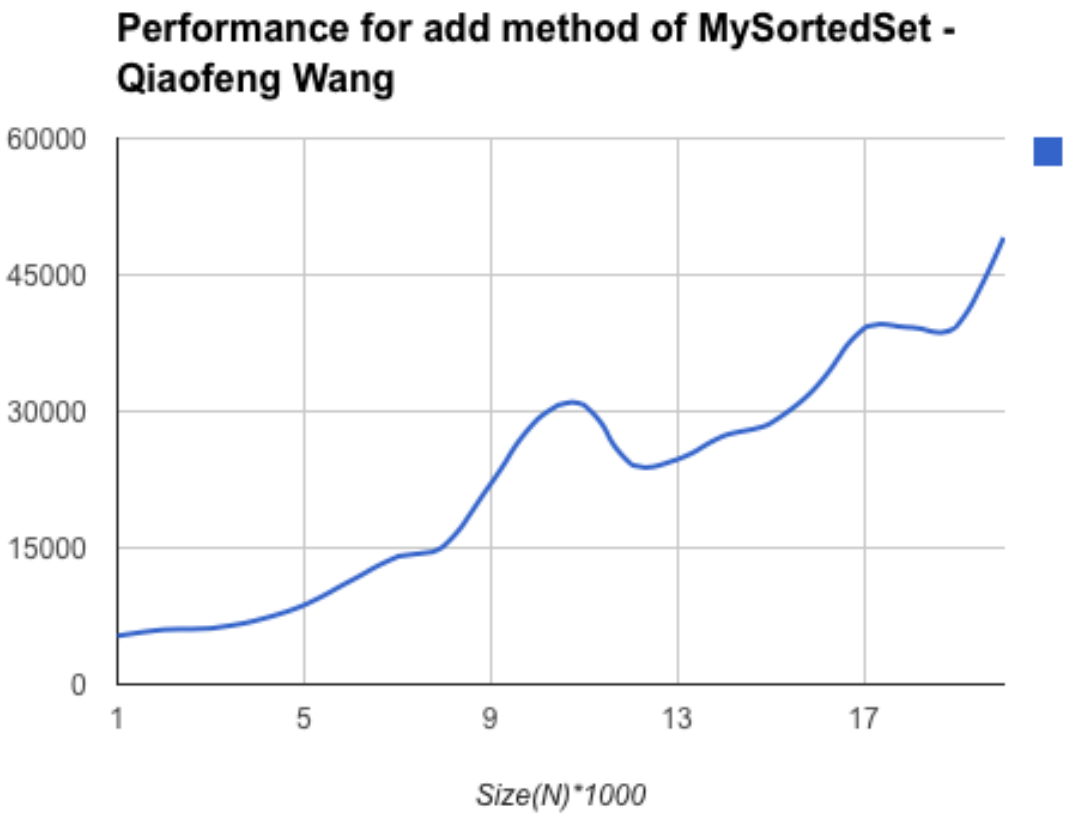
6. Plot the running time of MySortedSet's contains method for sets of sizes 100000 to 2000000 by steps of 100000. Use the timing techniques demonstrated in Lab 1. Be sure to choose a large enough value of timesToLoop to get a reasonable average of running times. Include your plot in your analysis document. Does the growth rate of these running times match the Big-oh behavior you predicted in question 5?



Yes, it is O(logN).

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)?

It should take O(logN) to locate the correct position at which to insert the element.
But when a new item is inserted to the array, the whole array has to be moved. So the plot should be O(N). Every item after the inserted item will be moved to the next index.
In worst case, it will take O(logN) to locate the position to add an element. Then inserted it will take O(N).



The plot is O(N), proved my answer.

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

15+ hours

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.