Matthew Canova

Analysis Document

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

Aaron Bellis

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 usually switch each session, which I think is fine.

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

I would prefer to work with Aaron for the rest of the semester, if possible. :)

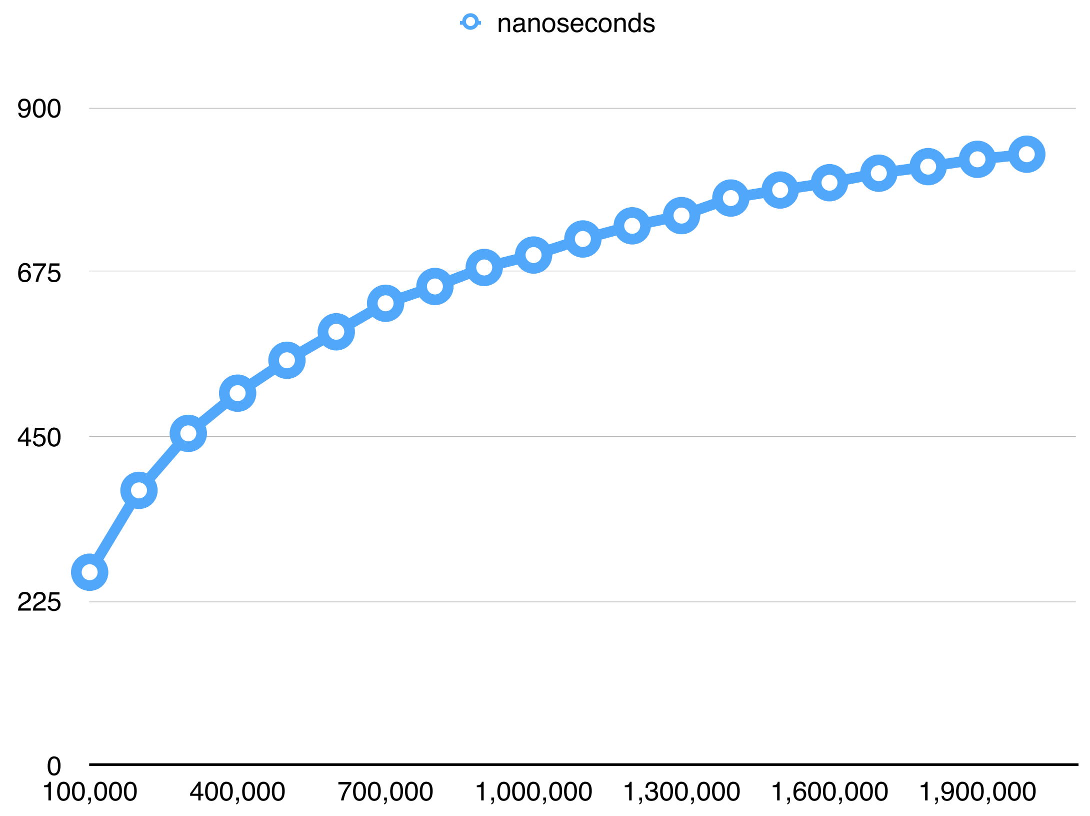
4. If you had backed the sorted set with a Java List instead of a basic array, summarize the main points in which your implementation would have differed. Do you expect that using a Java List would have more or less efficient and why? (Consider efficiency both in running time and in program development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

Using a Java List would have been more efficient because most of the functionality would have been implemented. Many of the methods we had to implement could have called the same method from the List. I expect it would be more efficient, since typically these implementations are exhaustively efficient, and I’m just an undergrad. :) It would obviously have been FAR more efficient from a development time perspective,

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

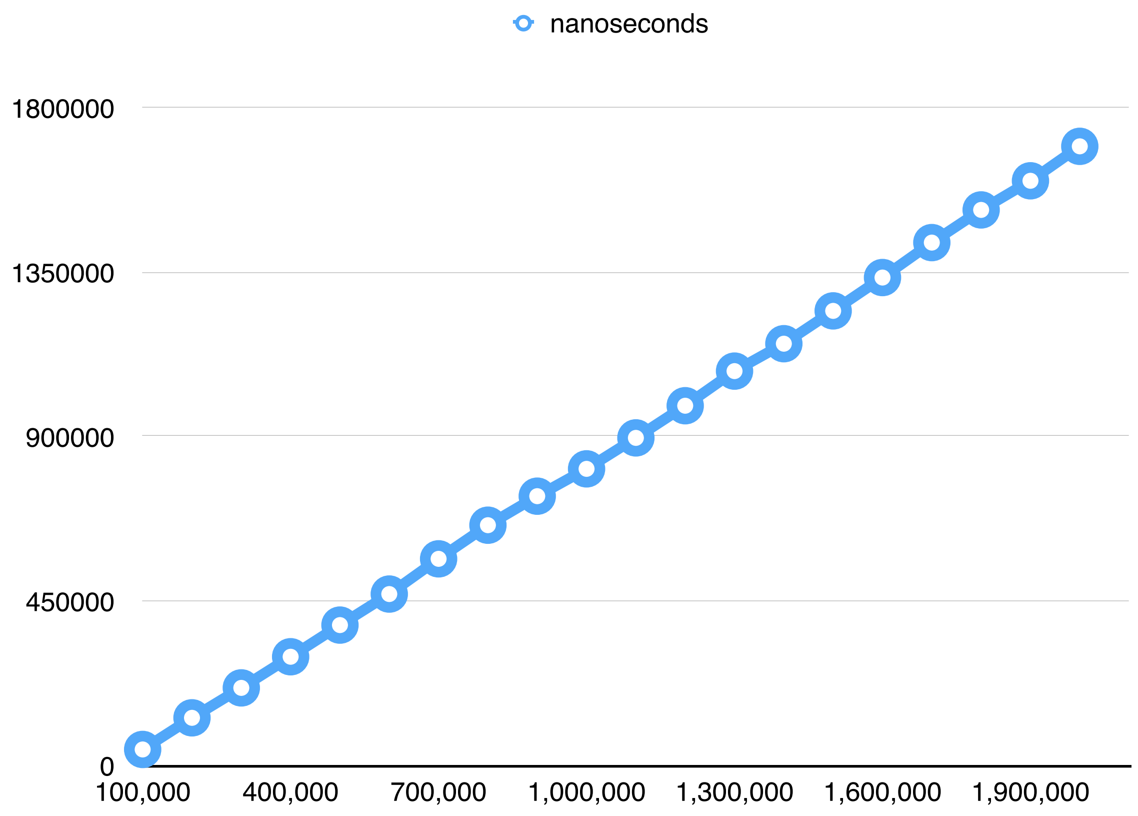
Since it primarily utilizes the binarySearch method that we wrote, I expect it to be O(logN) (which our graphs confirm). It is that order of operation because we are using a binary search and so half the set is disqualified in each step.

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?



The graph matches our expected results: 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)?



The amount of time it takes to find where the element would be added would be O(logN), since we would be using binary search, but since we have to move some modified N number of items to complete the add, the dominant factor is N and we are operating and O(N). I believe that our worst case would actually be O(NlogN), but I think average case is just O(N)

8. How many hours did you spend on this assignment?
10ish hours.