Thuy Nguyen
MySortedSet Assignment Analysis
February 4, 2015

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

Partner: Fred Javalera
I submitted the source code: Thuy Nguyen

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

I was the driver 70% of the time. I like programming a lot so I don't mind driving most of the time.

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

Fred seems not to be that strong of a programmer, even on the simple algorithms of returning the next element in the next() method of the Iterator class. He contributes by writing lots of test cases though, but even his test cases for this assignment didn't catch critical bugs such as expansion of the MySortedSet backing array. When we were debugging binary search, he made a fix that wasn't correct even though it made most of the test cases pass. So I had to carefully debug binary search in order to fix it correctly, not just relying on Fred to make a fix that he couldn't support. I wrote the critical parts of the assignment: binary search, insertion sort, add, remove, clear, and iterator. He says that he needs to look over the assignment for 1 or 2 days before working together next time, and I hope to see good results from his prior preparation.

I do love to problem solve, write algorithms, and type my programs. So it's okay if I do most or all the work. But I'm a bit worried of how Fred will manage harder assignments. I don't think I'll work with Fred again because I told him not to write unnecessary comments like, "return true", "return false", "declare and initialize x variable", and he got offended claiming that last semester's TAs gave him really good feedbacks for having all those comments. I told him that it's good programming practice to have our code be self-documentary, and we should write comments when it's not clear or common on what we're doing. Iterating with a for each loop doesn't qualify as uncommon, yet Fred wants to comment each and every line even though it's clear from the code of what is happening. For example: if (!set.contains(element)), Fred would write, "if set doesn't contain element", which is a clutter of space. Or right before a return true/return false statement he would literally write, "return true"/ "return false". But he told me that I can't deem a comment as unnecessary. After arguing about how he shouldn't write unnecessary comments, I don't think it's productive for me to work with Fred any longer.

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.)

I would use a LinkedList to answer this question:

With a LinkedList I would not have to worry about growing the backing array to accommodate adding more and more elements. I would not need to worry about shifting the elements over after removal of an object – the List would do that for me automatically. I also wouldn't worry about checking and casting to see if I can remove an object, since the LinkedList would do that automatically too. With a LinkedList, removal would take constant time, compared to removal for MySortedSet which is O(N), removal for LinkedList O(1) would be more efficient. However, in order to remove an element, I first have to find the element, and finding an element would take O(N) time for a Linked List, while I can find the element in O(log N) time with binary search. Overall, I do not think using a LinkedList would be more efficient than using backup array.

With a LinkedList I have methods already provided (and already tested) for me such as first, last, remove, removeAll, contains, containsAll, isEmpty, clear, iterator. However, I would need to implement my own methods of add because I need to do insertion sort after adding each element. Using a LinkedList would help me use a data structure that others have already tested (bug free) and with methods that others have already created. Also if I want methods such as listIterator which will let me traverse the set backwards, then the List would have that provided for me. So I would save time debugging and re-implementing iterator (for example). But there is a memory cost in using a LinkedList because of all the methods (a lot of them I won't need) and class creation overhead that come with a creating a LinkedList. A simple backing array does not take up much space thus more efficient in space complexity.

A Linked List *could* make some of my job easier because I can use the methods mentioned above, the methods that are bug free and that others have made for me. However, I don't think a LinkedList would be appropriate for MySortedSet, because MySortedSet is a special container that doesn't need all the methods that a LinkedList has such as push, pop, pollLast, pollFirst, indexOf, removeFirstOccurence, etc. In terms of programming time, using a LinkedList could have saved me time programming methods like remove and iterator. But I still needed to write the specialized add method, and I still need to do binary search for the contain methods.

Overall, I don't think LinkedList would be more efficient than using a backing array for MySortedSet in terms of space and time complexity. Perhaps more efficient in terms of easier for programmer to use the methods already in Linked List such as remove, first, last, and iterator. And for future methods that MySortedSet could take on that LinkedList already provides (like listIterator()). However, I would choose a backing array to implement MySortedSet because it's more appropriate for the type of operations that I'm doing to the set – insertion sort, binary search – and I don't need all those methods provided by the LinkedList. Besides, searching is the most important operation for the set, so I have to write my own search method using binary search, which the LinkedList methods don't provide me anyways. Also, I need to implement the

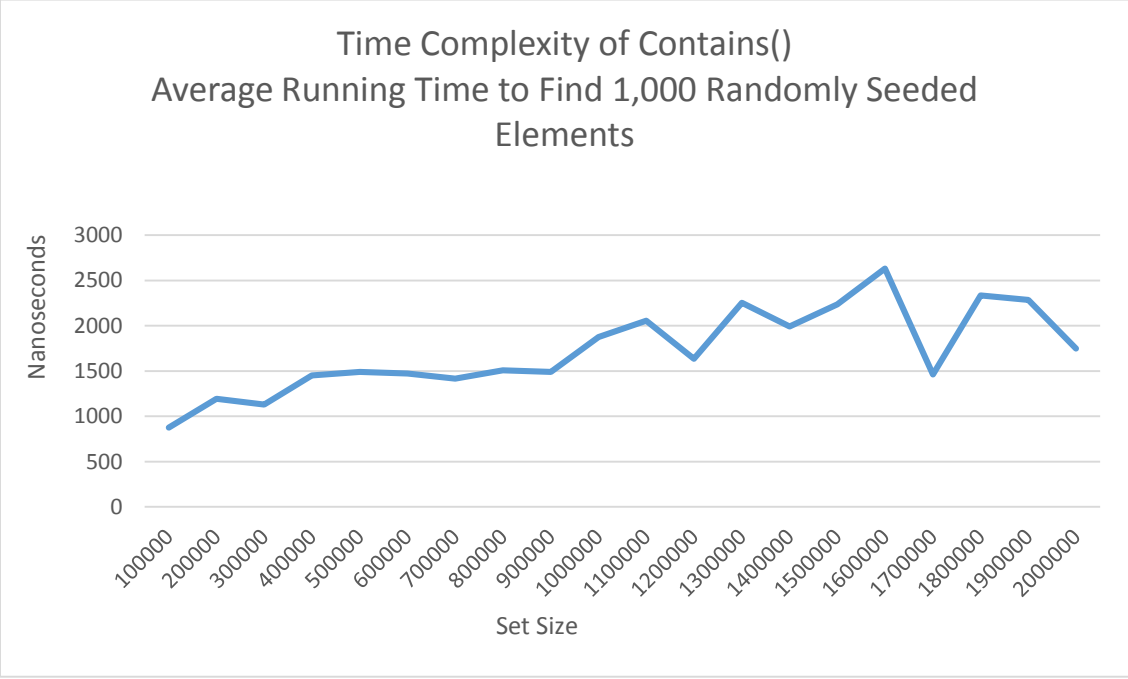
binary search in my remove, add, and contains methods, which LinkedList does not do, therefore, using a LinkedList for remove, add, and contains is not efficient. And using a LinkedList just to store my elements is not efficient because of the overhead memory cost of having a LinkedList. A backing array is better for implementing MySortedSet.

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

O(log n) because we're using binary search to see if the element is in there or not. And binary search is a divide-and-conquer algorithm which cuts the problem size in half after each look at the array.

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?

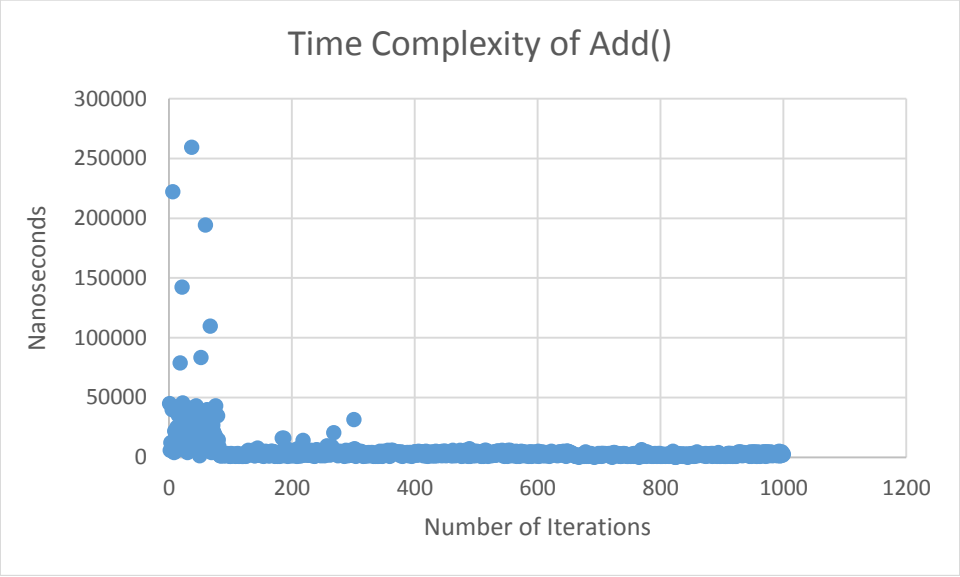
I made 20 sets, size 100,000 to 2,000,000 in increments of 100,000 elements. For each of those sets, I looped through 1,000 times to find 1,000 random numbers. Yes, the time complexity of the contains() method is O(log n) because the search time grows so little (or sometimes not at all) with each increment of 100,000 elements. There is fluctuation in time because the computer system is consistently changing, but overall, the running time is O(log n).



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)?

I filled the set with 1,000 random numbers. Then I looped through 1,000 times, only incrementing the iteration variable if I can successfully add a random number. I removed this random number after I calculated the time to takes to add the number. Adding an element would take O(n) running time because insertion sort could move the element all the way to the front of the array.

It takes about 5292.208 nanoseconds to add an element.



8. How many hours did you spend on this assignment?
About 16