Cody Cortello
u0781604
CS 2420 – Summer 2014

Assignment 3 Analysis

My programming partner is Casey Nordgran, and he will be submitting the source code for our program. We switched roles perhaps a bit less often than we should, but when one of us got really into typing a certain implementation we wouldn't switch after only twenty minutes. Sometimes it's hard to stay the 'driver' and simply type code when you think of an awesome way to implement the current method.

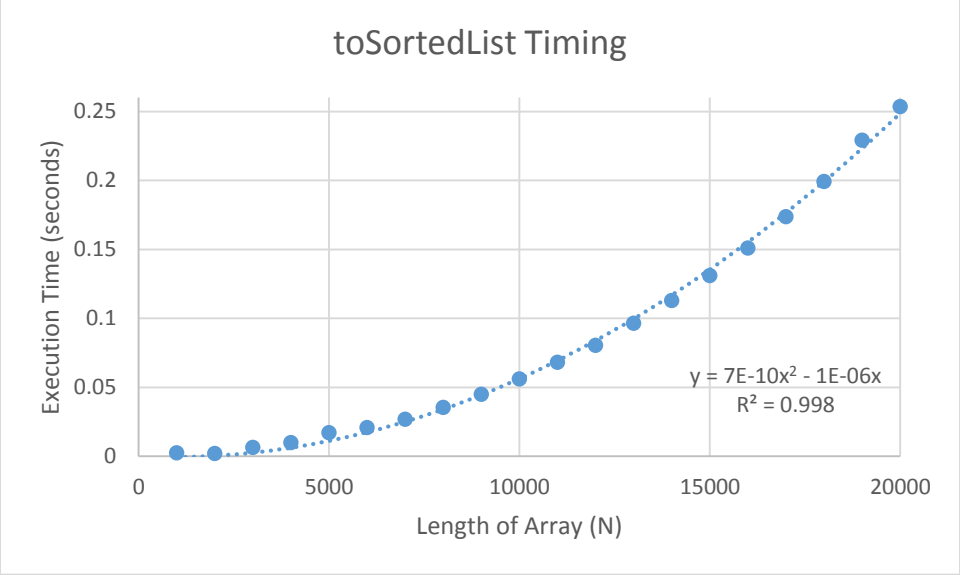
Honestly, I would've preferred to switch even less often, simply because I'm really good at quickly implementing code in IntelliJ, and I love driving way more than navigating. However, that would be delegation of roles and not equal programming between Casey and myself, so that's not an option.

I really like working with Casey and find him agreeable and clever. Despite learning Java through this class (he had taken the C++ version of 1410) and using a new programming environment he is quite up to speed. I do plan on working with Casey again, and will probably work with him until we have to switch partners later in the semester.

There wasn't really a part of the assignment which I didn't understand, as I took the class last semester, so it's hard to choose a most difficult part of the assignment. We didn't run into any major issues when debugging our code, and though I've had programming classes in the past where I just could not figure out why something wasn't working that hasn't happened in this class thus far. If I had to pick a most difficult part of the assignment I'd say it was implementing the toSortedArray() class simply because I incorrectly diagnosed a problem we were having while debugging it and spend perhaps ten minutes trying to find out why the insertion sort implementation wasn't working when it was the Comparator which was returning values we hadn't accounted for (I had forgotten that Comparable can return any integer, not just -1, 0, or 1).

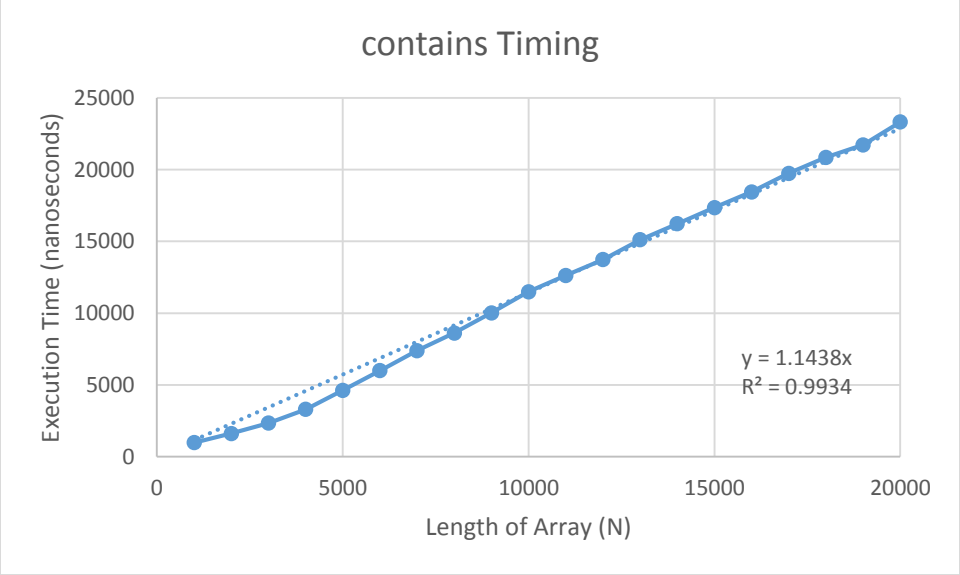
Cody Cortello
u0781604
CS 2420 – Summer 2014

When timing the toSortedList method I used a HashSet when adding Integers to ensure no duplicates were added to the list (even though the ArrayBasedCollection's add method ensures that no duplicates are added). The timing was the same as Paymon implemented in class, and I simply used the TimingDemo as framework code to time the toSortedList method. As shown in the graph below, the execution time is very clearly $O(N^2)$. This makes sense, as Insertion Sort is $\frac{N(N-1)}{4}$ on average, and toSortedList implements Insertion Sort.

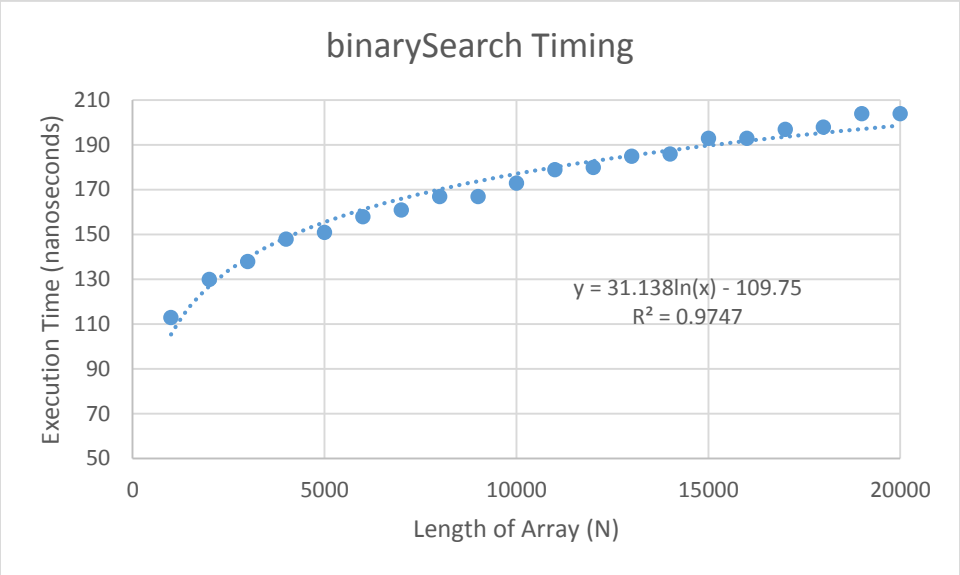


Similarly, I timed the contains method using the same timing code and the only real bit of trickery was substantiating a random array of integers which I'd added. Luckily Paymon's permuteInts() function was readily modified to permute an array as I needed, and since I'd already set up the timing for the toSortedList the rest of the timing had been implemented. The output, as graphed below, shows that the contains method is clearly $O(N)$, and interestingly it happened that almost exactly one comparison was made each nanosecond, on average. This shows correct contains implementation since contains should

iterate through half of the array on average (assuming that the item searched for is always in the array), giving the $O(N)$ complexity shown below. However, our implementation is closest to the worst case, being $O(N)$, as our line of best fit is $y = 1.1438x$ (the best case is if the object is always found on the first try – $O(1)$, and the average is iterating halfway through = $\frac{N}{2}$).



The binarySearch method was almost identical to the contains method for timing implementation, and I used the same array to search for items in an ArrayList I substantiated with toSortedArray before I began timing. As shown in the graph below, the complexity is very clearly $O(\log N)$, which confirms the expected complexity of a binary search.


binarySearch Timing
Execution Time (nanoseconds)
210
190
170
150
130
110
90
70
50
0
5000
10000
15000
20000
Length of Array (N)
y = 31.138ln(x) - 109.75
R² = 0.9747