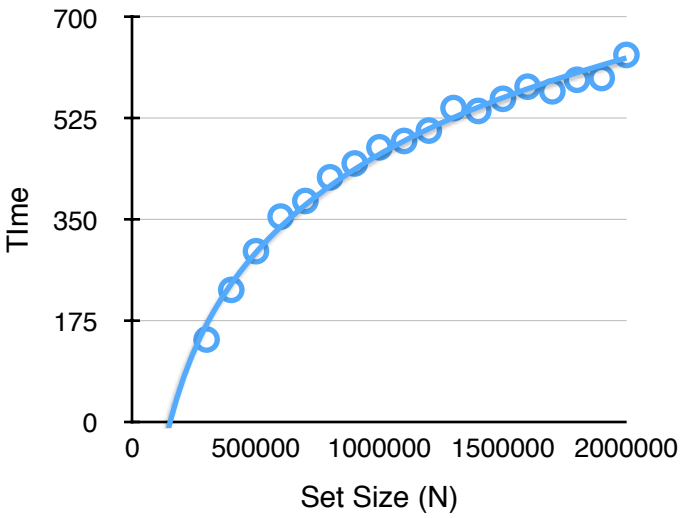
For this assignment I worked with Shibo Tang, the same partner I worked with last time, and I once again submitted the source code. We switched roles more this time than last however still not as often as we likely should have. Once again I felt like I had a much better idea of how to approach a problem or fix a problem, frequently I would come up with an implementation or solution faster than him and he never fought me on any decisions, so even when I was not driving it would usually be me telling him how to implement it instead. I think we worked much better when I was the one coding and talking about the ideas with him, and him offering suggestions or tweaks if my logic was not entirely correct on there was a better way to do it. When he's driving I think he does worse due to feeling pressure to come up with the solution and how to implement it quickly so we're not just staring at the screen, causing him to freeze up. Overall though I think we worked well together and I'm planning on working with him on the next assignment. While there were some issues we still completed the assignment in about 9-10 hours; 4-5 to finish coding the assignment, another 2-3 hours of testing and fixing bugs, and 1-2 hours typing up the analysis and running tests.

If we were using a Java list instead of just a regular array, it would have made things much more complicated. If we weren't required to use the binary search or have custom comparators I'd simply use a TreeSet since that deals with most of the requirements of the assignment (no duplicates, sorted and efficient along with having add/remove/contains methods). However because we needed those things it would have been difficult to the point I'm not entirely certain how I'd go about doing it, just having a differing comparator would complicate things as I'm not certain TreeSet has a way to take in a different one. If someone could figure it out I imagine it would be more efficient performance-wise but actually coding it would be far more difficult, and I'm not certain it the amount of effort would justify the increased performance.

Assignment 3 Analysis

Trevor Chapman

Timing for Contains Method



For the contain's method performance, I'd guess it would be log(N). For ours all we do is a binary search which is a log(N) search, and return true or false from that. After performing the timing tests, that seems fairly accurate. The difference between the time it takes to perform contains() on a set of 100,000 elements is not substantially different than the time it takes to perform contains() on a set of 2,000,000 elements, as shown in the above figure

The add's method performance was much worse. Because it combines the binary search in finding where to add the element, in addition to moving everything over, it performs at O(Nlog(N)) at it's worst. However, with the way we programmed it to add an element to the end of the list is O(1).

Timing for Add Method