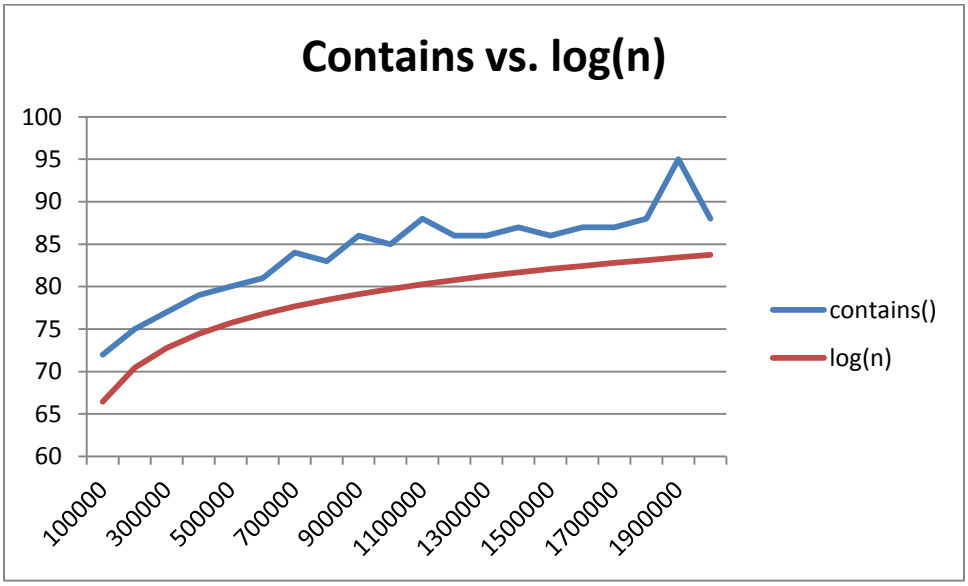
Lucas Fivas
2015-02-05
CS2420-008
Assignment 3

# Assignment 3 Analysis Document

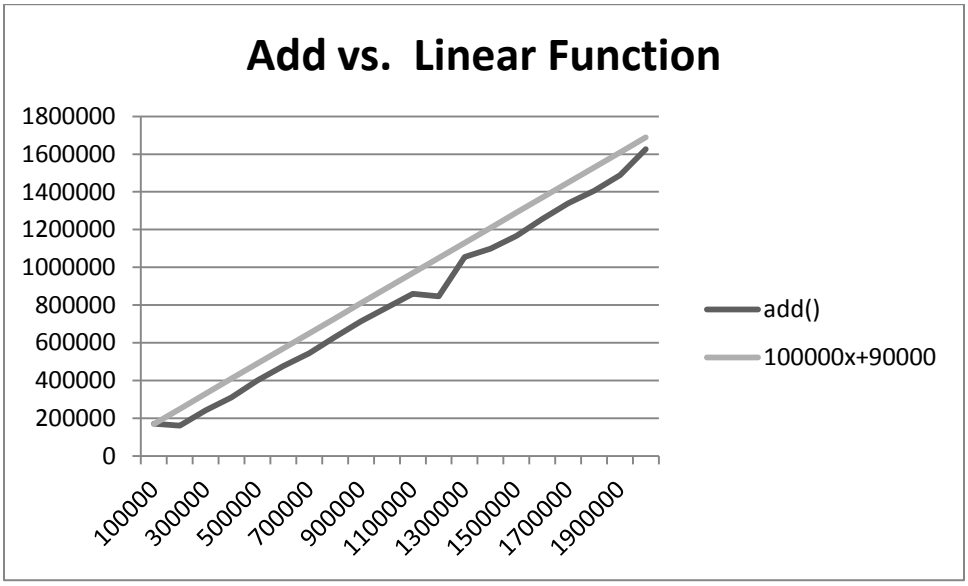
My programming partner for this assignment was Maxwell Wright. Generally, we tended to switch roles every 30-60 minutes, though we occasionally switched roles spontaneously when one of us had an idea for how to solve a problem. Overall, Maxwell is an acceptable partner. I feel like his slow typing and tendency to get distracted and switch tasks before finishing what he was working on slowed us down a bit, but as with the previous assignment, he had numerous insights on how to solve a problem that I wouldn't have thought of. I do plan to work with him again.

Using a Java list would have certainly made the process of programming this assignment much more efficient, and the code much simpler. For one, since we sorted the list as items were added, we could simply use the list's iterator instead of rolling our own. We wouldn't have had to keep track of the list's capacity and current count on our own, since the list would do that for us, and all we'd need to do for add and remove would be to use the list's `add(int index, E element)` and `remove(int index, E element)` methods. On the other hand, searching would be a bit slower, since the `add()` and `remove()` methods would have the extra baggage from the list's implementation. I think overall our implementation would have been less efficient. There's a reason that we still have access to basic arrays when ArrayList essentially provides all of their functionality and more, and there's a reason why other data structures which rely on arrays don't wrap them in a list.

The contains () method should follow O(log(n)) behavior. It does use a binary search, after all, which should be O(log(n)). I took deliberate care to ensure that we didn't run into any surprises with the code's complexity.



As you can see, the complexity here does, in fact, seem to follow a logarithmic function. Unfortunately for testing purposes, the algorithm is just too fast to get a reliable measurement - each data point above was the result of calling contains() 1,000,000 times in 1,000 call bursts and aggregated. The graph still ended up quite bumpy, but it still clearly follows the log(n) graph below.




These statistics were generated by creating 50 identical sets and adding the same item to them all in one go. As you can see, the complexity of this is linear, just as I'd suspect. While Add relies on the O(log(n)) worst-case behavior of binary search, it still has to shift any elements greater than the one it is adding, which is an O(n) operation. It's still considerably faster than adding without a binary search would be on average, but it's nowhere near as fast as contains().

In the end, I'd say I spent about ten to twelve hours on this assignment.