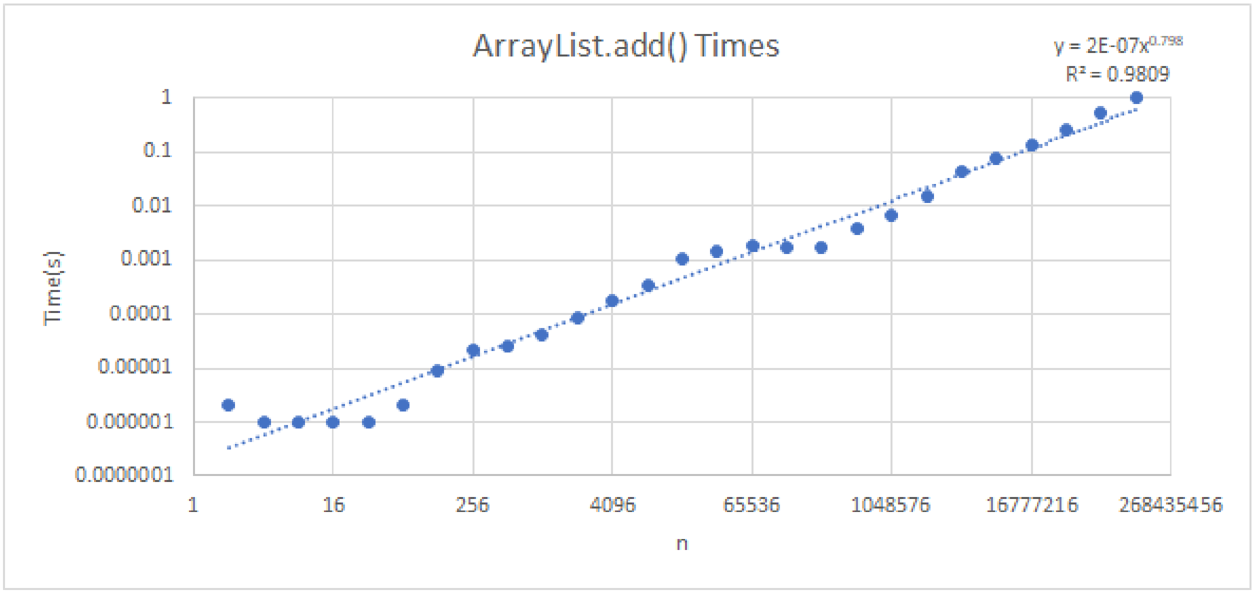
Joshua Stallings

Integrated Computer Science

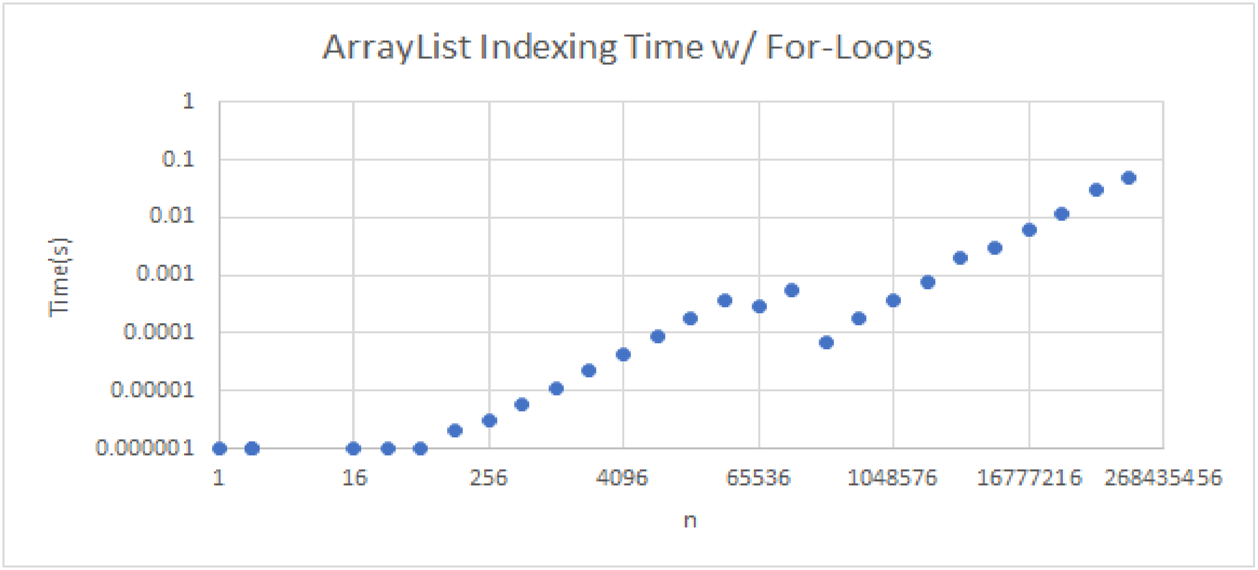
Teacher Squirrel Seward

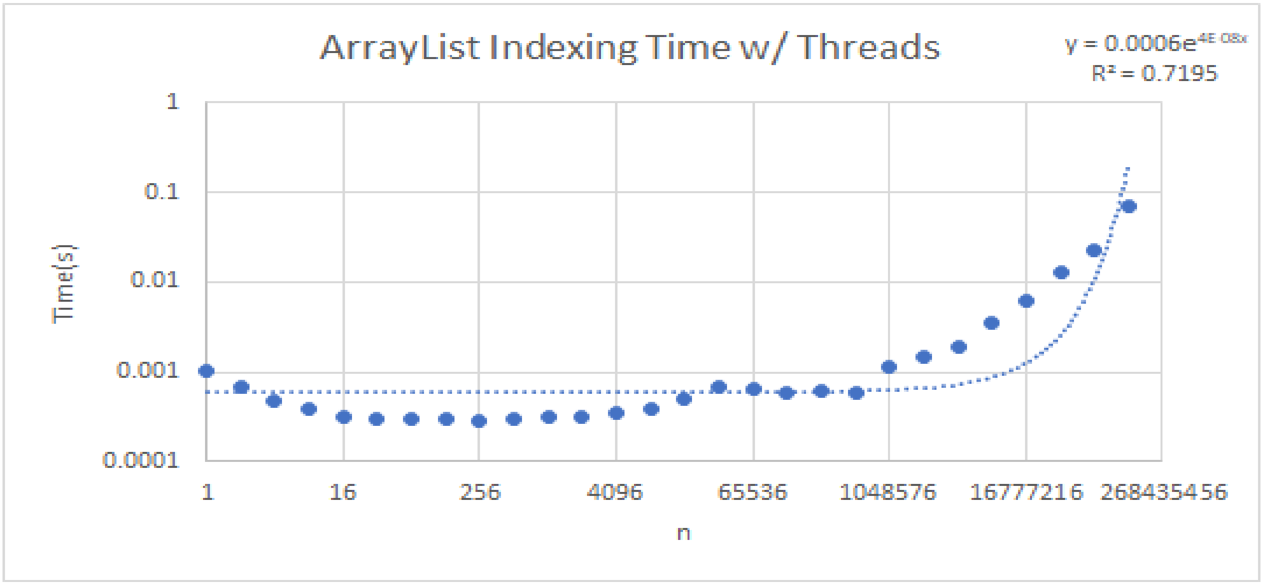
3 March 18

The assignment for the ICS class was to make an ArrayList that implemented ICSList and had most of the functions of the Java ArrayList. Then it was supposed to be benchmarked by timing how long it took to add 2^i integers until the computer could not handle any more. I also experimented slightly with two different ways of finding the index of a specified value; one was a for-loop through the entire array until the value was found, the other used threads to split the array into 3 pieces with one thread going through each piece. This was intended to be slightly faster than the for-loop through the entire array.



Here is the chart for the ArrayList.add() times. The largest number of items that the computer could handle was 1.34E+08, and that took about one second to do. Overall, nothing about this was surprising or horrible.





Another part of the ArrayList that was timed was the time for it find the index of a value with a for-loop versus three threads going through the list with 3 smaller for-loops. It was expected for the threads to go at least slightly faster or about the same speed as the for-loop, but as shown above this was not the case. The for-loop could index more items at the same speed.

When the threads started with a similar amount of items, they indexed noticeably slower before becoming a little bit faster and generally increasing speed. With extremely large amounts of items, they both completed the indexing in about the same amount of time.

Overall, the ArrayList performed about as expected which is to say not absolutely horrible. I had expected the thread idea to at least have similar speed compared to the for loop, but it did not. This was probably because the threads were more involved in setting themselves up and had to call more functions whereas the for-loop was just a for-loop.