Assignment 4 Report

Exercise 3

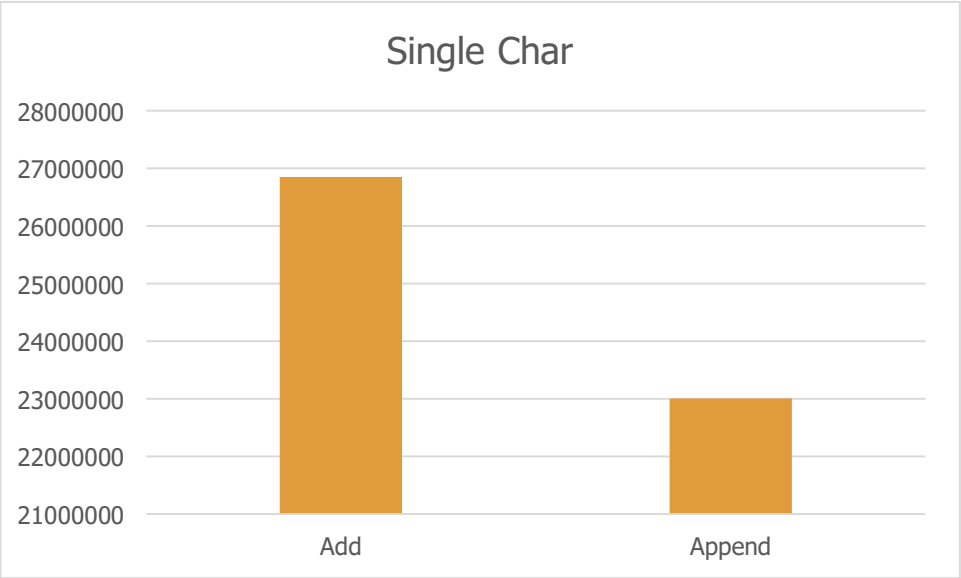
Setup

For the test the baseline was number of computes in 1 second. To get a more secure result the tests ran 10 times and the mean value was calculated.
I created a for loop for the number of tests. Inside I created another do-while loop that was ran as long the time was under 1 second. i.e. the number of computes in 1 second.

For the stringbuilder using append the toString() was included in each iteration inside the do-while.

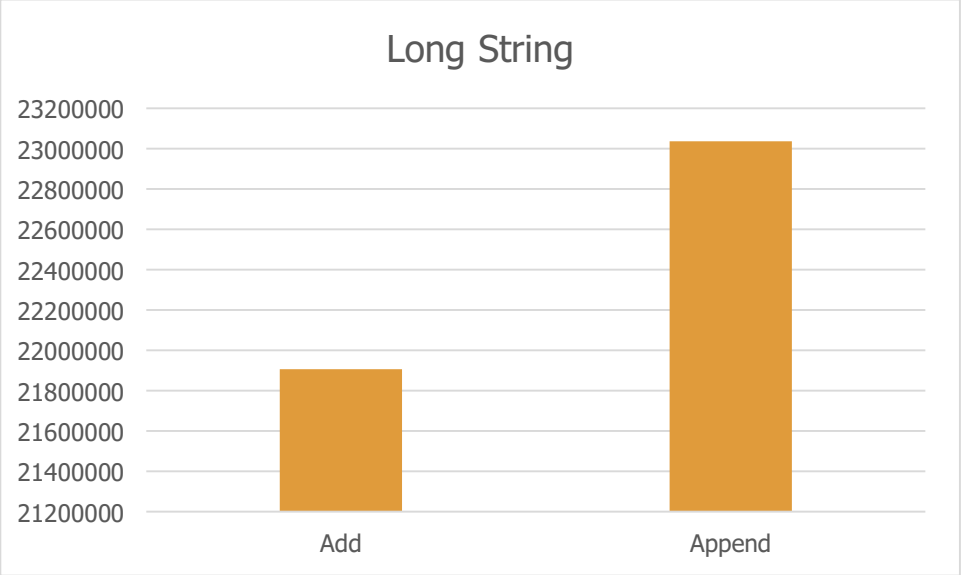
Tests was conducted on an Mac Pro 5.1 with 3,33 GHz 6-Core Intel Xeon and 24GB RAM running OS X El Capitan v10.11.2.

Test results



The table above shoes how many concatenations is done within 1 second.

The length of each string is the same as the concatenations counted.



The table above shoes how many concatenations is done within 1 second.

To the string length you multiply the calculated concatenations by 80.

2

Discussion

This result did baffle me a bit since using the + operator made more computes than using the append in stringbuilder when concatenating single chars. Not sure it might be that using the toString() affects the stringbuilder when only appending single chars, i.e. append is using two computes and using the + operator only uses 1 compute adding the char.

Concatenating longer string (80 chars) is where the results is more expected since stringbuilder using append computes a lot more then using the + operator.

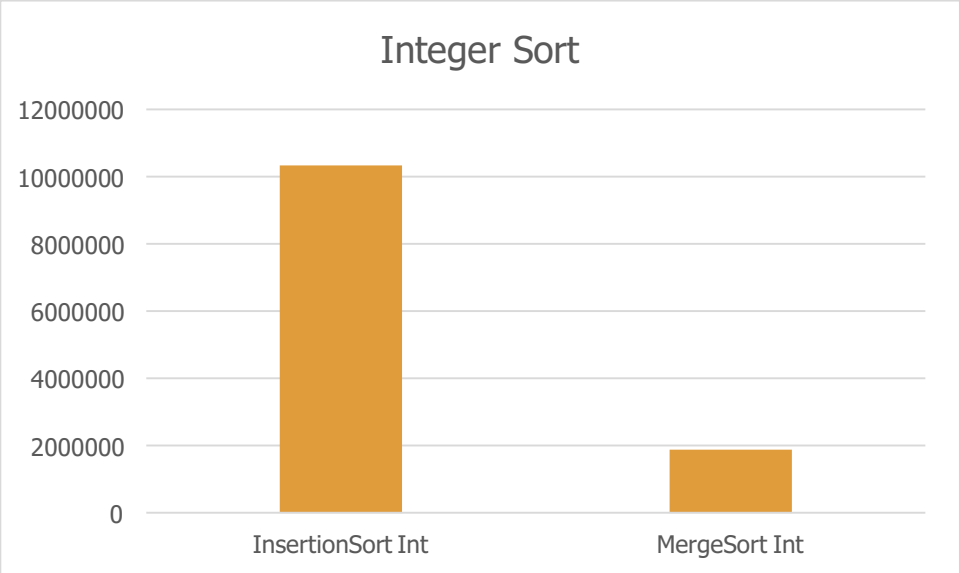
Exercise 4

Setup

I created a main method that called others inside that did the actual tests. The tests ran 10 times and the mean value was calculated before returning the time elapsed. The methods all hade the same size of the array tested. Size of the array = 8000.

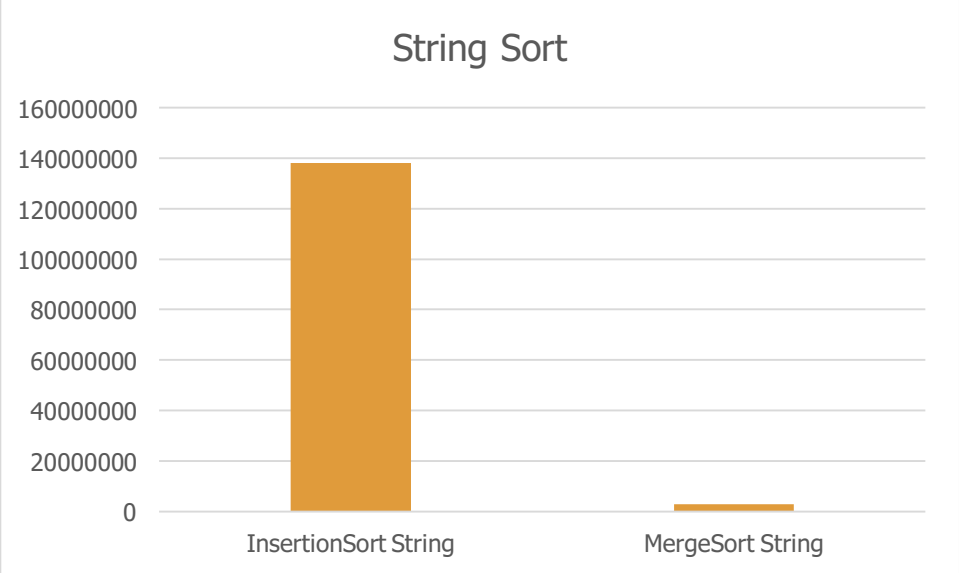
Tests was conducted on an Mac Pro 5.1 with 3,33 GHz 6-Core Intel Xeon and 24GB RAM running OS X El Capitan v10.11.2.

Test Results



3

The table above shows how many nanoseconds it took for the algorithms to sort and array with a length of 800.



The table above shows how many nanoseconds it took for the algorithms to sort and array with a length of 800. The strings used had a size of 10 characters each.

Discussion

Merge sort is fastest on both sorting integers and strings.
To my understanding that is what's expected. Merge sort has a faster algorithm for sorting larger arrays.

I could have increased the size of the arrays to see how large array could be that merge sort could have sorted within 1 second. These result does how ever show that given an equal array the merge sort algorithm is a lot faster then insertion sort, especially when it comes to strings.