Dylan Maloy

CS150 Lab

Lab #4 write-up

09/22/19

Introduction:

The goal of this lab was to implement some of the sorting algorithms that we learned (insertion, selection) along with some search methods that we learned (linear, binary). It also introduced the use of CSV files, and how to correctly make one of them. I did not make many assumptions in terms of the code layout / the code itself because the directions were pretty straight forward.

Approach:

My implementation contains 3 classes, one being the experimentController, another being the randomStringGenerator which generates a random string given the seed and amount of items, and the third being the StringContainer which controls all of the sort and search methods for the list. I ended up doing all of my searches 5000x in order to produce a number worth recording (not 0ms).

Methods:

For my experiments I ran each method 3 times (averaged after all runs), using 4 integer strings, and used a plethora of data sizes. The one pictured below being (500, 5000, 15000, 30000). I used these sizes because they gave me good numbers for my sorting algorithms (wouldn't take too long), while still getting valid results when searching 5000x.

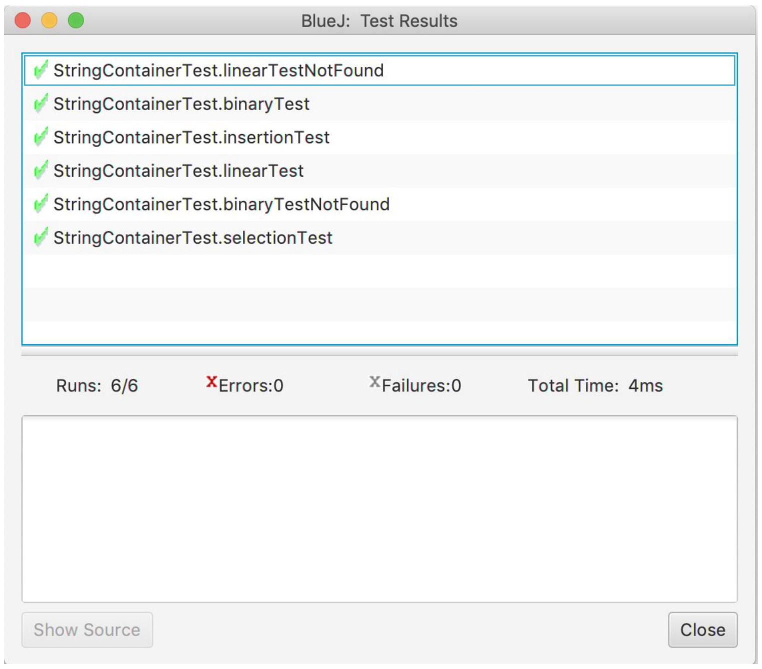
Data and Analysis:

From this data I found that the selection sort is a lot more efficient for mostly sorted / already sorted lists, however, it lacks / is similar in larger lists that are unsorted compared to the insertion sort. From this data, it shows that it is also hard to get good data from small data sets for search methods because they are too fast. However, when running the search methods, the binary search tended to be much more efficient for large data sets compared to the linear search which is only more efficient in very small sets of data. All of these conclusions were made over the course of many runs using many different data sizes / sets.

Conclusion:

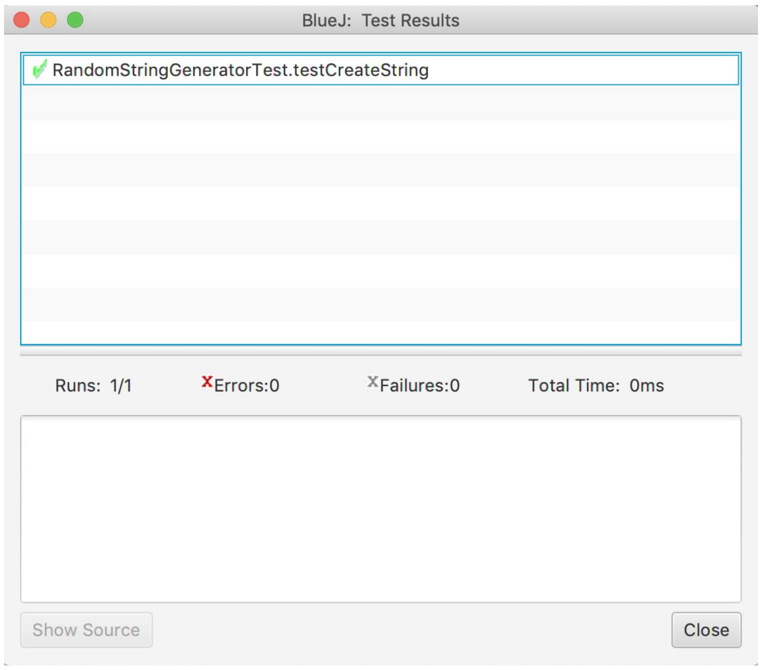
In conclusion I created a program which would run the sort and search methods given the parameters required (amount of runs, amount of integers per string, different data sizes). However, even though my program worked, I found that it was hard to get good sort AND search data on the same run (either the data was too big for sorting, or too small for searching). I solved this by doing the searches 5000x in order to produce a time worth recording.

Figures / Tables:



```
✓ StringContainerTest.linearTestNotFound
✓ StringContainerTest.binaryTest
✓ StringContainerTest.insertionTest
✓ StringContainerTest.linearTest
✓ StringContainerTest.binaryTestNotFound
✓ StringContainerTest.selectionTest

Runs: 6/6     Errors:0     Failures:0     Total Time: 4ms
```

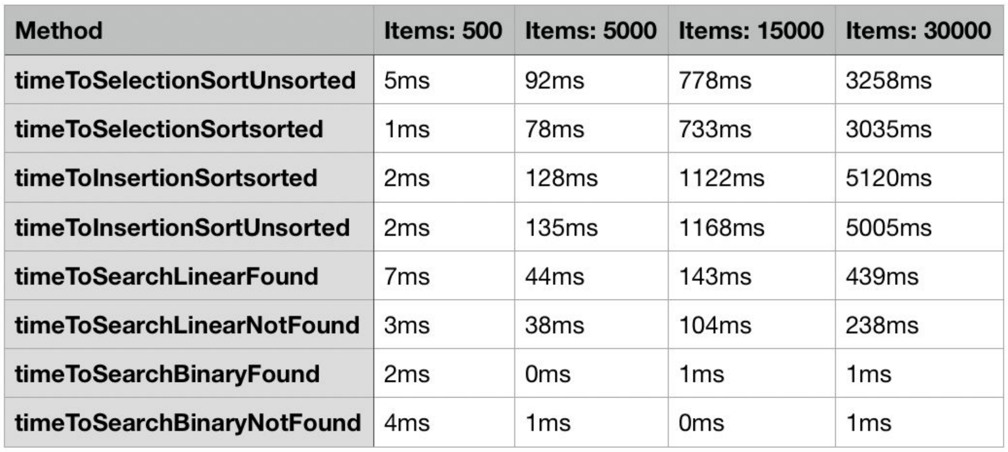
- String container unit tests



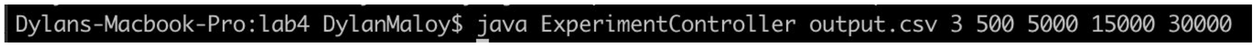
```
✓ RandomStringGeneratorTest.testCreateString

Runs: 1/1     Errors:0     Failures:0     Total Time: 0ms
```

- String generator unit test

output

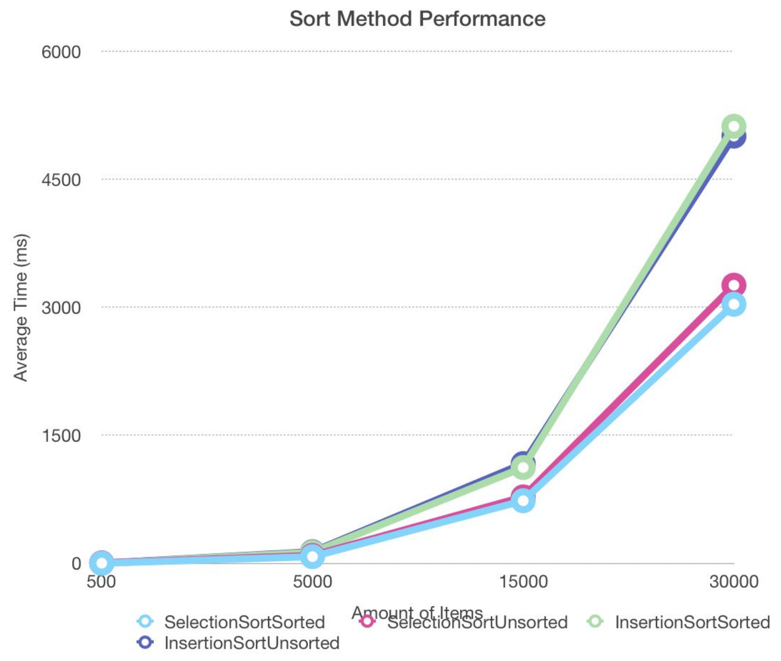


Method	Items: 500	Items: 5000	Items: 15000	Items: 30000
timeToSelectionSortUnsorted	5ms	92ms	778ms	3258ms
timeToSelectionSortsorted	1ms	78ms	733ms	3035ms
timeToInsertionSortsorted	2ms	128ms	1122ms	5120ms
timeToInsertionSortUnsorted	2ms	135ms	1168ms	5005ms
timeToSearchLinearFound	7ms	44ms	143ms	439ms
timeToSearchLinearNotFound	3ms	38ms	104ms	238ms
timeToSearchBinaryFound	2ms	0ms	1ms	1ms
timeToSearchBinaryNotFound	4ms	1ms	0ms	1ms

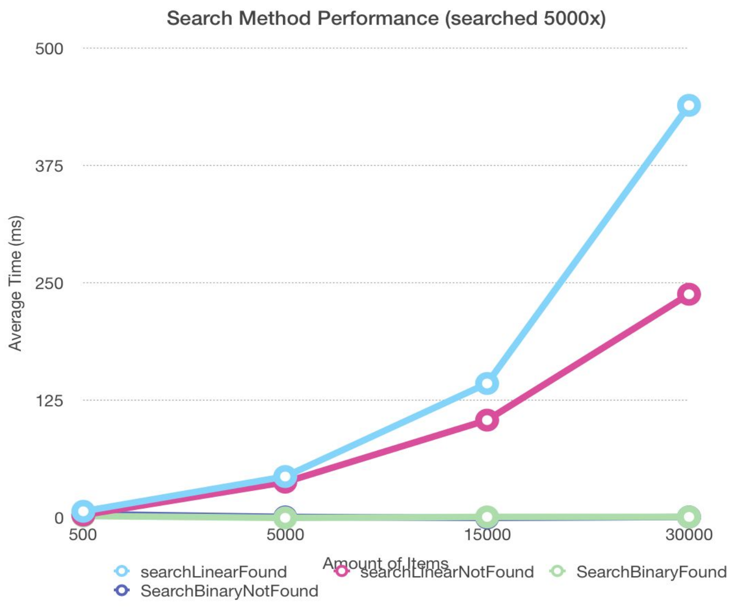


```
Dylans-Macbook-Pro:lab4 DylanMaloy$ java ExperimentController output.csv 3 500 5000 15000 30000
```

- CSV output averaged over 3 runs (4 different amounts of items) (4 integer strings) && console input for this run



- Sort method performance for different data sets



- Search method performance for different data sets (searched 5000 times)

References:

Sorting Algorithms:

- https://learn.zybooks.com/zybook/LAFAYETTECS150PfaffmannFall2019/chapter/4/section/1

Java API for arrayList:

- https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Java compareTo:

- https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html