SOFT 437 Assignment 3

Hai-Ling Rao, 20121206, 18HLR

What is Instrumentation?

Generally, instrumentation refers to measuring the performance of a product. In software performance analysis, software instrumentation is used to monitor the performance of our program in order to spot any performance issues.

Purpose of this Experiment

The purpose of this experiment is to:

- Test my implementation of instrumentation
- Using my instrumentation class, view the performance of some algorithms for the same task (sorting)
- Observe the performance impact of instrumentation on the overall software performance

Experiment Variables

There are a few variables in this instrumentation that could cause inconsistent results when testing sorting:

- Processing speed - CPU speed differs from system to system, and a CPU may be slower at some times due to handling other tasks.
- Random array ‘task’ - depending on the state of the randomized array, sorting the array may take longer or shorter, which can make it difficult to measure true sorting time or compare results between sorting algorithms
- Instrumentation overhead: the cost of the instrumentation affecting the performance of the program being instrumented
- The first run of the program is also usually slowest due to some initial processing

Test Case Design

In order to make the tests reproducible and representative:

- Run the program many times and take the average to get a more confident result
    - The loop size should be large enough to get a consistent result each time
- Arrays randomized since some arrays sort faster than others

- Each program being compared has to sort the same randomized arrays
- Calculate the instrumentation overhead

Instrumentation Log Analysis + Test Drive Questions

Overhead of Instrumentation (Q1)

- The overhead of the instrumentation calls are mainly from requesting the current time in nanoseconds/milliseconds, and storing the startTiming() calls in a stack.
- **Timing the overhead: Test.testInstrumentationOverhead()**
  - To test the overhead of a pair of start/stop timings, I ran start/stop thousands(10,000) of times to get the average time cost for a pair of instrumentation calls.

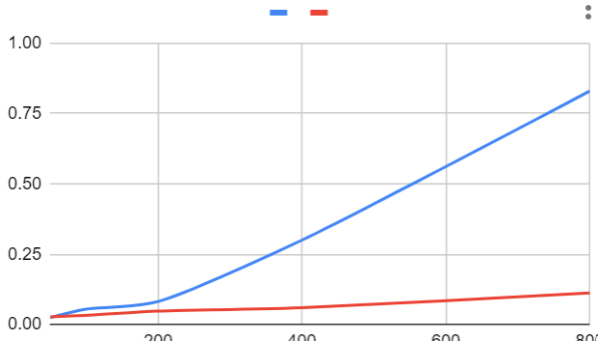
> 💡 Average run time per pair of start/stop calls was 221ms/10000 ~= 0.022ms

BubbleSort vs QuickSort (Q4)

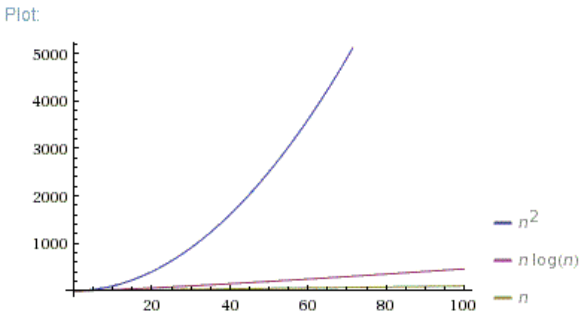
Each test is done 1000 times, and the average taken.

ArraySize(n)	Total Time BubbleSort(ms)	Total Time QuickSort(ms)	BubbleSort Avg.	QuickSort Avg
50	25	28	0.03 ms	0.03 ms
100	55	33	0.06 ms	0.03 ms
200	82	48	0.08 ms	0.05 ms
400	300	60	0.30 ms	0.06 ms
800	829	112	0.83 ms	0.11 ms
1600	2614	160	2.61 ms	0.16 ms
3200	13948	505	13.95 ms	0.51 ms
6400	80231	866	80.23 ms	0.87 ms

Which one is faster?



red: quicksort; blue: bubble sort



different time complexities

We can see that, as the array size scales, QuickSort is much faster than Bubble Sort? This is because the QuickSort algorithm is n(log n) time complexity while BubbleSort is n^2(does not scale as well with array size).

Main and populateArray() (Q5)

Running times for main

- Running time for my instrumentation is usually the sum of the Instrumentation initialization, all my sorting tests, as well as the cost of running populateArray and the dump() at the end of the test.

Timing populateArray():

- similar to the instrumentation overhead test, I run populateArray() many times(10,000) to get the average, and also measure time for different sizes:

Size	1000	2000	4000	8000	16000
Time(ms)*1000	180	322	619	1200	2412

We can see here as the size doubles, the time also roughly doubles, which makes sense for an O(n) operation.

Sorting with Instrumentation(Q6, Q7)

Prediction:

- As calculated in the previous questions, the average sorting time with instrumentation should increase by around 0.022ms
- as the size of the array scales, the O(nlogn) and O(n^2) times will dominate the O(1) instrumentation time cost.
- Bubblesort is slower, so the instrumentation cost becomes negligible sooner.

Results:

- As predicted, smaller array sizes such as 50, where the sorting time was around 0.03, had their average time doubled due to instrumentation overhead
    - the difference was not exactly 0.022ms, as inconsistencies with testing environment can cause small variance.
- however, larger arrays were dominated less by the instrumentation, where the instrumentation overhead made <5% difference to the overall sorting time.
- QuickSort is a significantly faster algorithm, so instrumentation will make up a larger portion of the overall run time compared to BubbleSort (for the same array size).