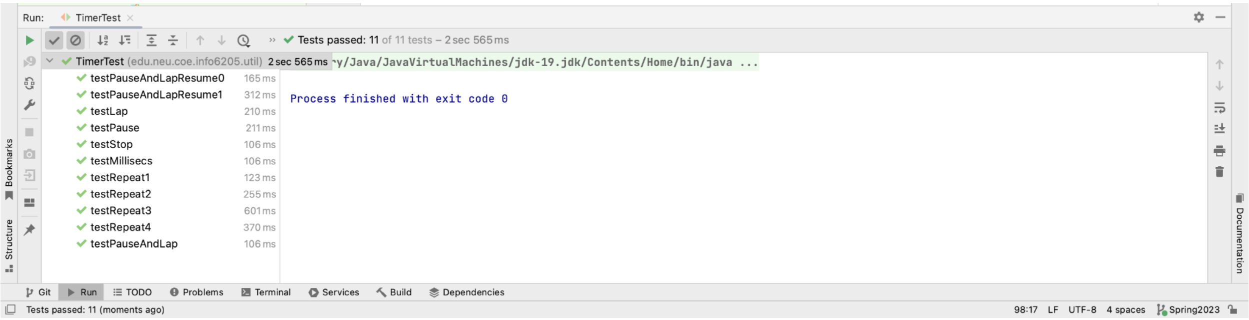
Assignment - 3 Benchmark

Link to programs: https://github.com/sachshan/INFO6205_Assignment3

1. Unit Tests for Timer Class



```
Tests passed: 11 of 11 tests – 2 sec 565 ms
TimerTest (edu.neu.coe.info6205.util) 2 sec 565 ms
    testPauseAndLapResume0   165 ms
    testPauseAndLapResume1   312 ms
    testLap                  210 ms
    testPause                211 ms
    testStop                 106 ms
    testMillisecs            106 ms
    testRepeat1              123 ms
    testRepeat2              255 ms
    testRepeat3              601 ms
    testRepeat4              370 ms
    testPauseAndLap          106 ms
```

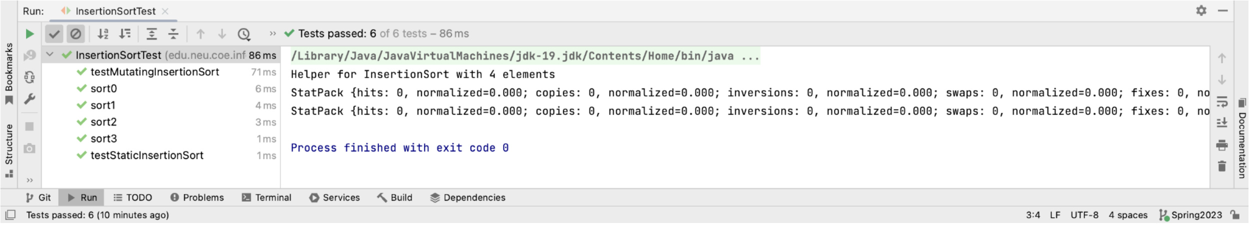
```
ry/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```

```
Tests passed: 11 (moments ago)
```

Figure 1. The image above shows all test cases pass for the timer class.

2. Unit Tests for Insertion Sort Class



```
Tests passed: 6 of 6 tests – 86 ms
InsertionSortTest (edu.neu.coe.inf 86 ms
    testMutatingInsertionSort   71 ms
    sort0                        6 ms
    sort1                        4 ms
    sort2                        3 ms
    sort3                        1 ms
    testStaticInsertionSort      1 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java ...
Helper for InsertionSort with 4 elements
StatPack {hits: 0, normalized=0.000; copies: 0, normalized=0.000; inversions: 0, normalized=0.000; swaps: 0, normalized=0.000; fixes: 0, no
StatPack {hits: 0, normalized=0.000; copies: 0, normalized=0.000; inversions: 0, normalized=0.000; swaps: 0, normalized=0.000; fixes: 0, no

Process finished with exit code 0
```

```
Tests passed: 6 (10 minutes ago)
```

Figure 2. The image above shows all test cases for the insertion sort class.

3. Observation regarding Order of Growth of Insertion Sort Function

Array Length	Random Array	Ordered Array	Partially Ordered Array	Reverse Array
500	76498	1313	64317	154882
1000	304694	2385	277750	594088
2000	1180312	5121	916372	2750225
4000	4807605	9073	3854889	11426376
8000	22866714	19685	16569321	46841514

Table 1. This table displays the Sort Runtimes (nanoseconds) for different array lengths and orderings.

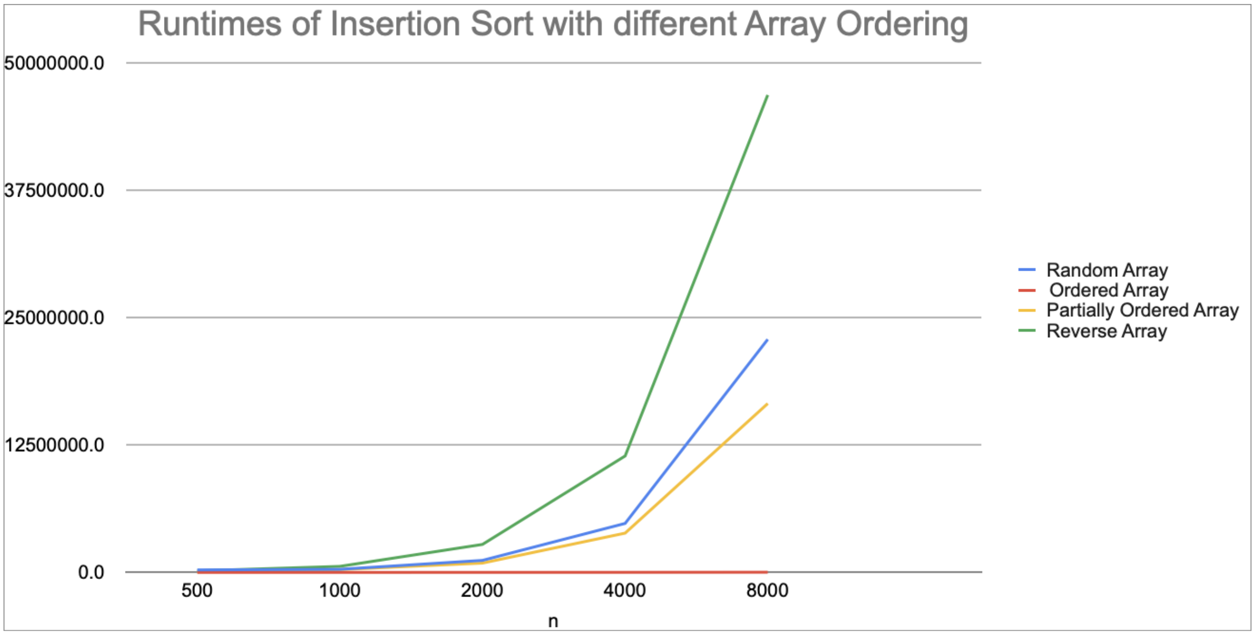


Figure 4. The Graph Above plots the relationship between the array size and runtime(nanoseconds) for different ordering of the arrays.

I. Ordered Array: As seen in the table and graph, the runtime of the ordered array is linear. The runtime is linear, as the array loops for the total length of the array and does not make any swaps.
II. Reverse Array: As seen in the graph, this ordering has the highest runtime. We can derive from the table that the runtime of insertion sort quadruples each time the array size doubles. Each element is swapped for the remaining array length in each iteration. Thus, the runtime of insertion sort for the reverse array ordering is quadratic.
III. Random Array: The slope of the graph is less in the reverse order, but the increase in runtime is quadratic
IV. Partially Ordered: The slope is more than ordered and less than random ordered. The runtime of the array is quadratic.