Program Structures and Algorithms
Spring 2024

NAME: Gautham Venkata Krishna Prasad
NUID: 002249901
GITHUB LINK: https://github.com/gauthamkris7neu/INFO6205Assignment

Task:
Assignment 3 (Benchmark)

Relationship Conclusion:

- Random Array :
  The sorting time for this type of array increases significantly as the size of the array (n) doubles. This behavior is consistent with the expected time complexity of Insertion Sort, which is O(n^2) in the average and worst case.
- Ordered Array :
  The sorting time remains very low (mostly 0ms, increasing slightly for larger n), indicating that InsertionSort is highly efficient for arrays that are already sorted. This is because the inner loop of the algorithm hardly runs in this case, making the time complexity approximately O(n).
- Partially Ordered Array :
  The time taken for partially ordered arrays seems to follow a trend like that of random arrays, but usually slightly less. This indicates that while Insertion Sort benefits from some degree of order within the array, the time complexity remains close to O(n^2) for large n, albeit with a lower constant factor.
- Reverse Ordered Array :
  The sorting time for reverse-ordered arrays grows the fastest, which is expected since this represents the worst-case scenario for InsertionSort. Each element needs to be compared to each of the sorted elements, resulting in a time complexity of O(n^2).

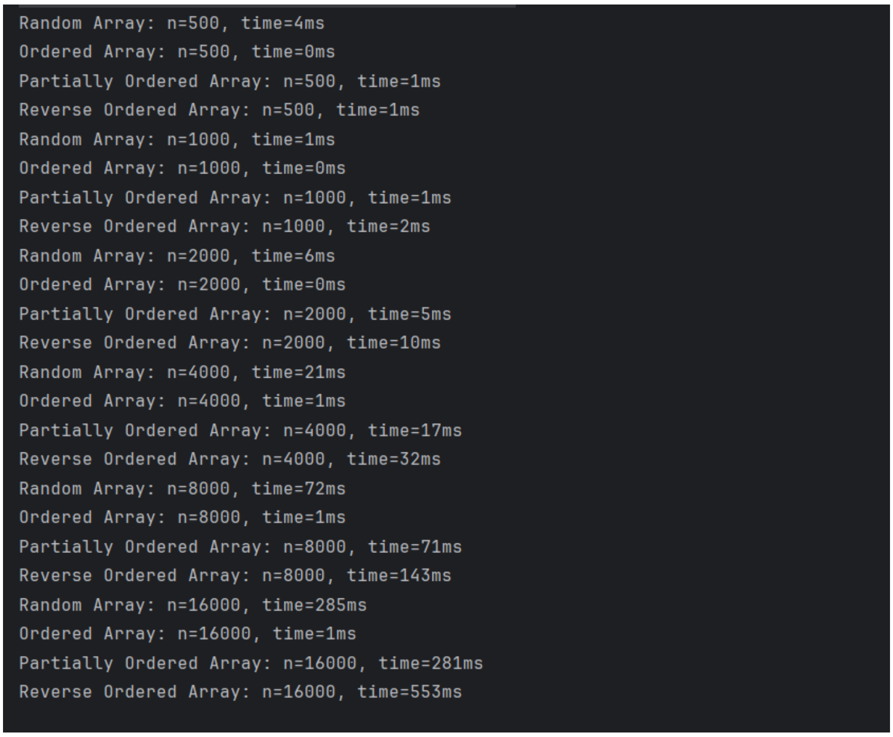
Evidence to support that conclusion:

Experiment Result for the 4 Different types of Arrays:

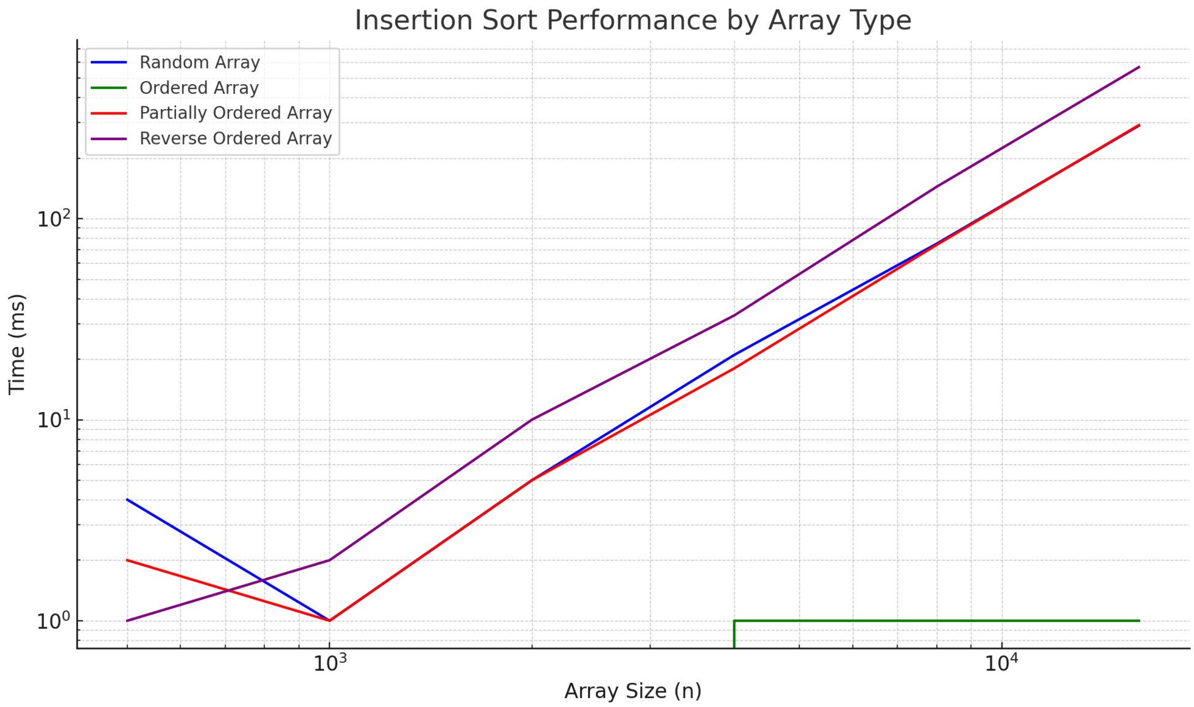
Array Type	Array Size (n)	Time (ms)
Random Array	500	4
Ordered Array	500	0
Partially Ordered Array	500	2
Reverse Ordered Array	500	1
Random Array	1000	1
Ordered Array	1000	0
Partially Ordered Array	1000	1
Reverse Ordered Array	1000	2

Random Array	2000	5
Ordered Array	2000	0
Partially Ordered Array	2000	5
Reverse Ordered Array	2000	10
Random Array	4000	21
Ordered Array	4000	1
Partially Ordered Array	4000	18
Reverse Ordered Array	4000	33
Random Array	8000	75
Ordered Array	8000	1
Partially Ordered Array	8000	74
Reverse Ordered Array	8000	144
Random Array	16000	290
Ordered Array	16000	1
Partially Ordered Array	16000	292
Reverse Ordered Array	16000	567

The above data was collected by running the method in Insertion Sort class for 5 values using doubling method :



```
Random Array: n=500, time=4ms
Ordered Array: n=500, time=0ms
Partially Ordered Array: n=500, time=1ms
Reverse Ordered Array: n=500, time=1ms
Random Array: n=1000, time=1ms
Ordered Array: n=1000, time=0ms
Partially Ordered Array: n=1000, time=1ms
Reverse Ordered Array: n=1000, time=2ms
Random Array: n=2000, time=6ms
Ordered Array: n=2000, time=0ms
Partially Ordered Array: n=2000, time=5ms
Reverse Ordered Array: n=2000, time=10ms
Random Array: n=4000, time=21ms
Ordered Array: n=4000, time=1ms
Partially Ordered Array: n=4000, time=17ms
Reverse Ordered Array: n=4000, time=32ms
Random Array: n=8000, time=72ms
Ordered Array: n=8000, time=1ms
Partially Ordered Array: n=8000, time=71ms
Reverse Ordered Array: n=8000, time=143ms
Random Array: n=16000, time=285ms
Ordered Array: n=16000, time=1ms
Partially Ordered Array: n=16000, time=281ms
Reverse Ordered Array: n=16000, time=553ms
```

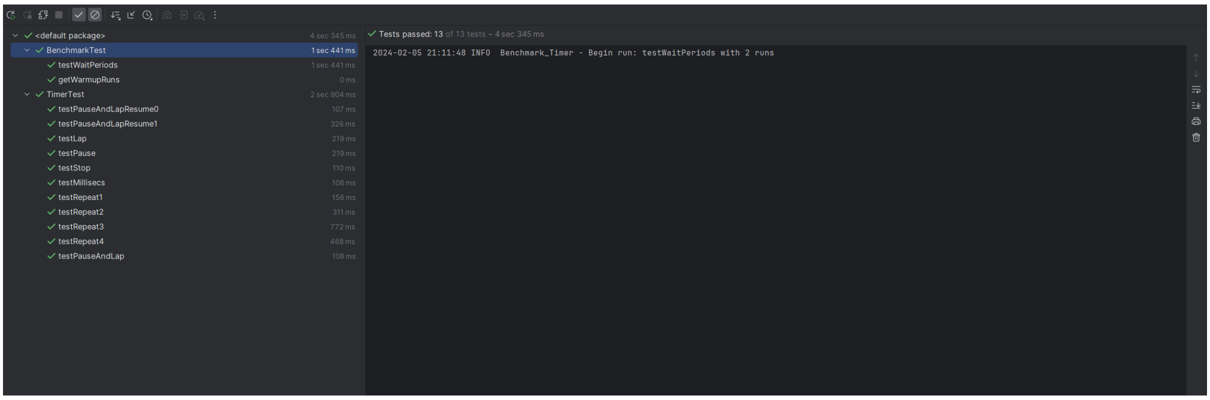


The drastic increase in sorting times for the Random, Partially Ordered, and Reverse Ordered arrays as n increases confirms the non-linear growth, characteristic of O(n^2) complexity.

The low and mostly constant sorting times for Ordered Arrays across different sizes of n demonstrate the efficiency of Insertion Sort when the input is already sorted.

Unit Test Screenshots:

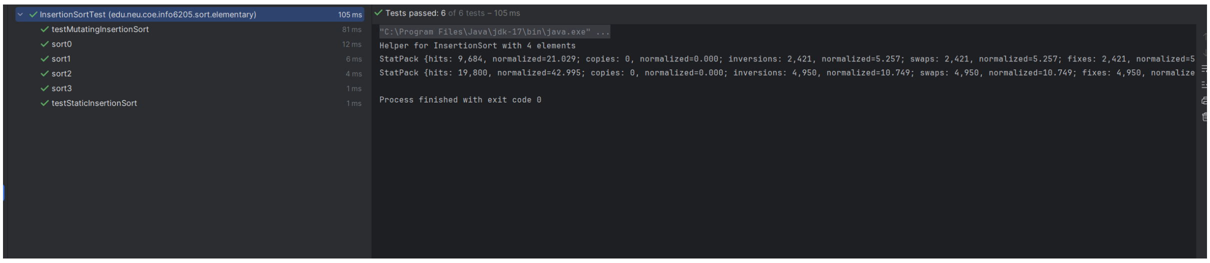
- **Timer :**



```
✓ <default package>                 4 sec 345 ms
  ✓ BenchmarkTest                   1 sec 441 ms
    ✓ testWaitPeriods               1 sec 441 ms
    ✓ getWarmupRuns                 0 ms
  ✓ TimerTest                       2 sec 904 ms
    ✓ testPauseAndLapResume0        107 ms
    ✓ testPauseAndLapResume1        326 ms
    ✓ testLap                       219 ms
    ✓ testPause                     219 ms
    ✓ testStop                      110 ms
    ✓ testMillisecs                 108 ms
    ✓ testRepeat1                   156 ms
    ✓ testRepeat2                   311 ms
    ✓ testRepeat3                   772 ms
    ✓ testRepeat4                   468 ms
    ✓ testPauseAndLap               108 ms
```

```
✓ Tests passed: 13 of 13 tests – 4 sec 345 ms
2024-02-05 21:11:48 INFO  Benchmark_Timer - Begin run: testWaitPeriods with 2 runs
```

- **Insertion Sort :**



```
✓ InsertionSortTest (edu.neu.coe.info6205.sort.elementary)   105 ms
  ✓ testMutatingInsertionSort                               81 ms
  ✓ sort0                                                   12 ms
  ✓ sort1                                                   6 ms
  ✓ sort2                                                   4 ms
  ✓ sort3                                                   1 ms
  ✓ testStaticInsertionSort                                 1 ms
```

```
✓ Tests passed: 6 of 6 tests – 105 ms
"C:\Program Files\Java\jdk-17\bin\java.exe" ...
Helper for InsertionSort with 4 elements
StatPack {hits: 9,684, normalized=21.029; copies: 0, normalized=0.000; inversions: 2,421, normalized=5.257; swaps: 2,421, normalized=5.257; fixes: 2,421, normalized=5
StatPack {hits: 19,800, normalized=42.995; copies: 0, normalized=0.000; inversions: 4,950, normalized=10.749; swaps: 4,950, normalized=10.749; fixes: 4,950, normalize

Process finished with exit code 0
```