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 :

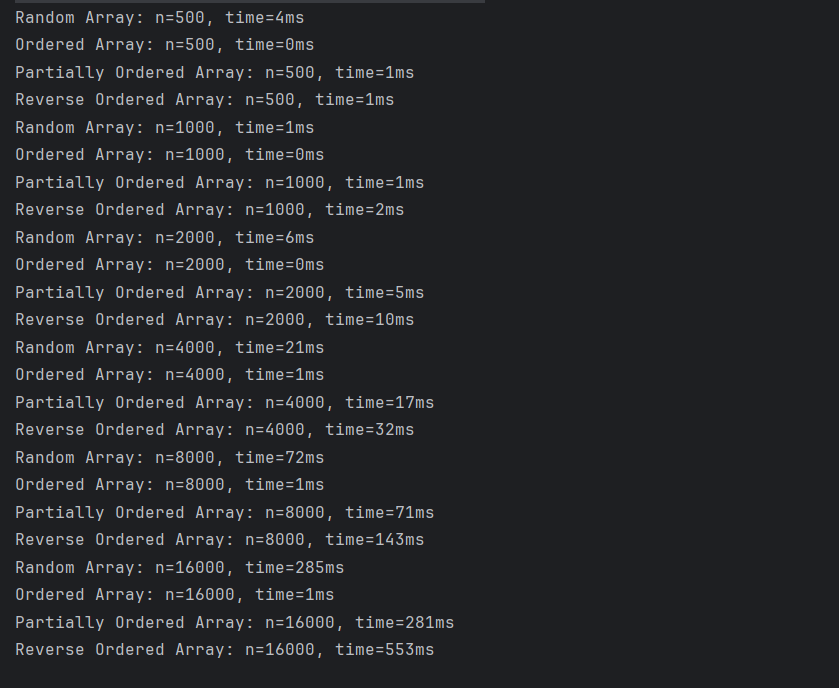
The sorting time for reverse-ordered arrays grows the fastest, which is expected since this represents the worst-case scenario for InsertionSort. Each element needs to be compared to each of the sorted elements, resulting in a time complexity of O(n^2).

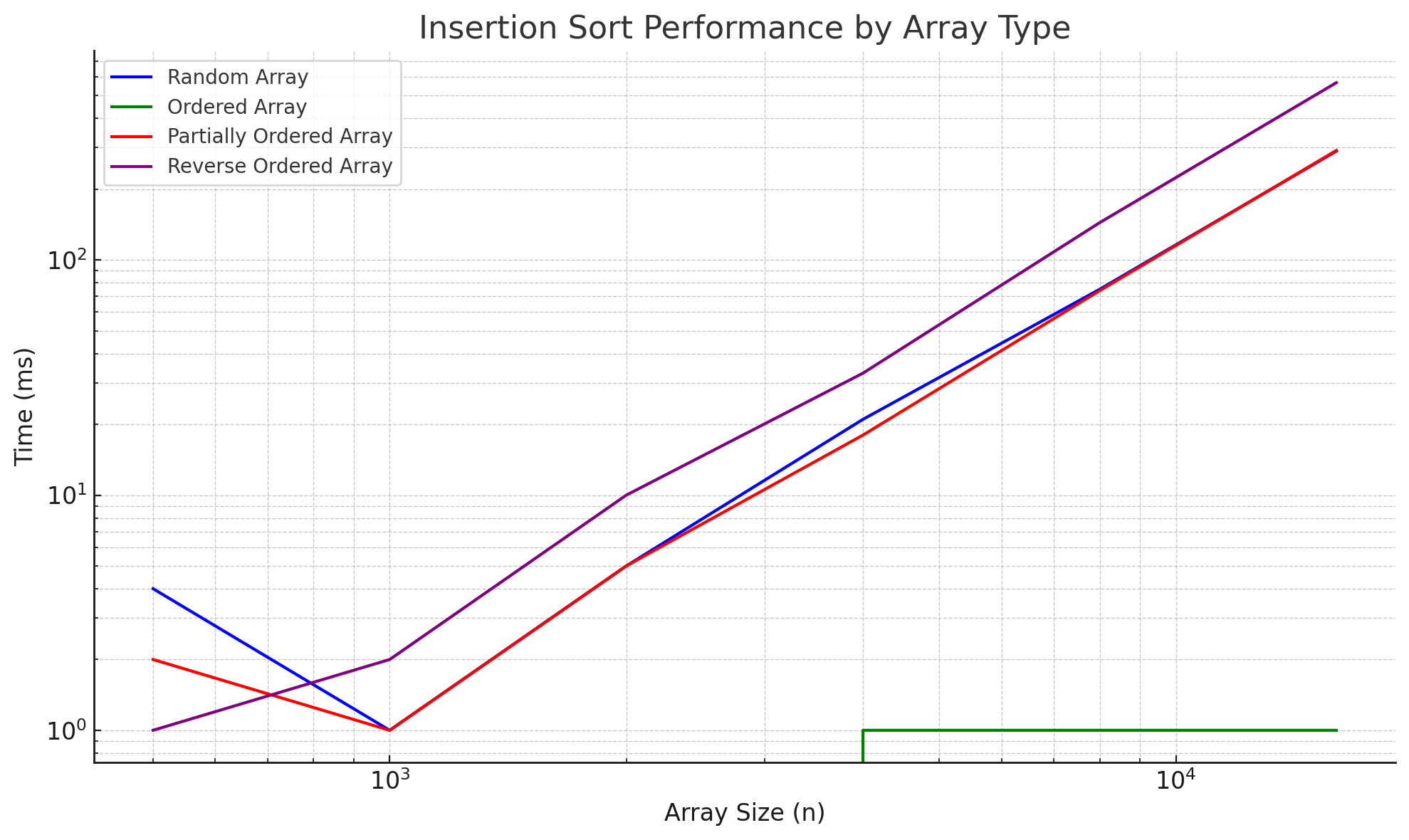
**Evidence to support that conclusion:**

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 :



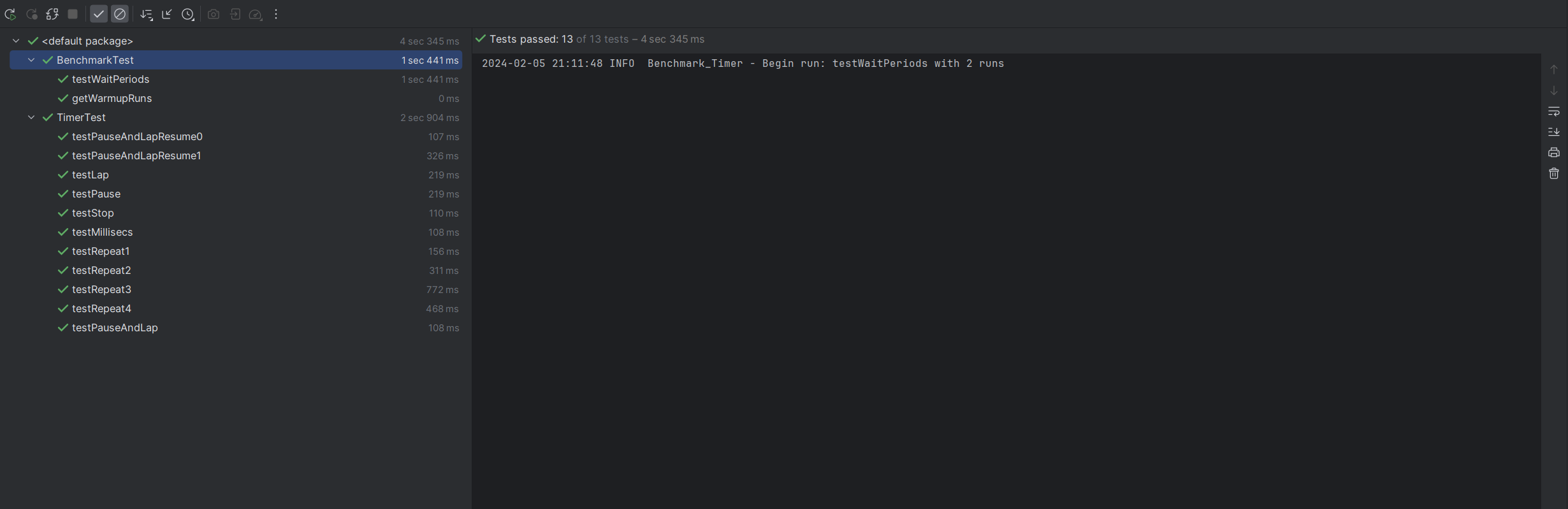


The drastic increase in sorting times for the **Random**, **Partially Ordered**, and **Reverse Ordered** arrays as n increases confirms the non-linear growth, characteristic of O(n^2) complexity.

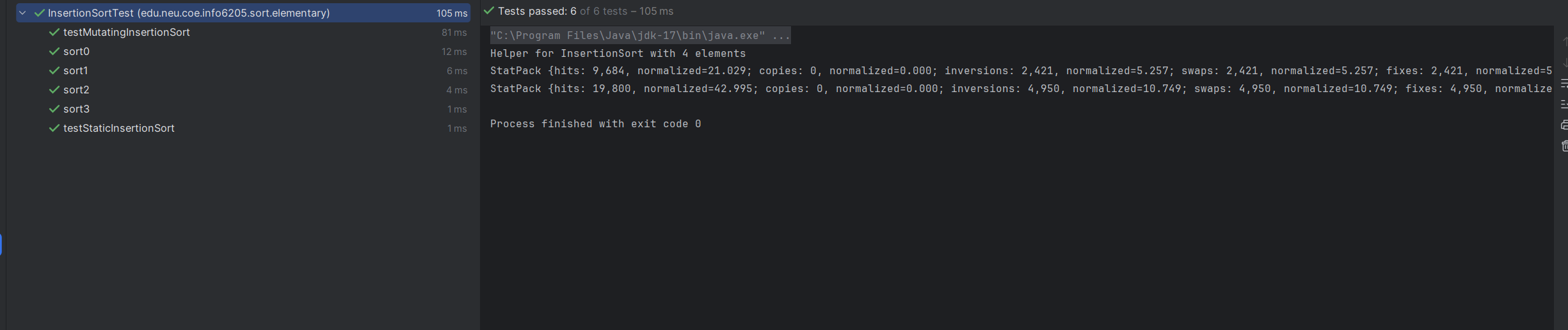
The low and mostly constant sorting times for **Ordered Arrays** across different sizes of n demonstrate the efficiency of Insertion Sort when the input is already sorted.

**Unit Test Screenshots:**

* **Timer :**

****

* **Insertion Sort :**

****