**INFO6205 PROGRAM STRUCTURES AND ALGORITHMS**

**Assignment-2**

**Task:**

Part1: To implement three methods of a class called Timer and check implementation by running the unit tests in BenchmarkTest and TimerTest.

public class Timer {  
... // see below for methods to be implemented...  
}

public <T, U> double repeat(int n, Supplier<T> supplier, Function<T, U> function, UnaryOperator<T> preFunction, Consumer<U> postFunction) {  
// TO BE IMPLEMENTED  
}

private static long getClock() {  
 // TO BE IMPLEMENTED  
}

private static double toMillisecs(long ticks) {  
 // TO BE IMPLEMENTED  
}

Part2: To Implement InsertionSort (in the InsertionSort class) by simply looking up the insertion code used by Arrays.sort and run the unit tests in InsertionSortTest.

Part3: To Implement a main program (or you could do it via your own unit tests) to actually run the following benchmarks: measure the running times of this sort, using four different initial array ordering situations: random, ordered, partially-ordered and reverse-ordered, arrays to be sorted are of type Integer and Draw any conclusions from observations regarding the order of growth.

**Unit tests and Output:**

Conducted the timer test and the benchmark test for insertion sort by randomly generating an input array of type integers and size n and running the experiment 100 times for each n values doubling. The size of n was: 100,200,400,800, 1600, 3200. I have run the experiment for an ordered array, randomly ordered array, reverse ordered array and partially ordered array. Below are the test results for the experiments:

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Ordered Array of input size: 100 insertion sort takes meantime of : 0.02452

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Ordered Array of input size: 200 insertion sort takes meantime of : 0.01601

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Ordered Array of input size: 400 insertion sort takes meantime of : 0.032690000000000004

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Ordered Array of input size: 800 insertion sort takes meantime of : 0.058530000000000006

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Ordered Array of input size: 1600 insertion sort takes meantime of : 0.10612999999999999

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Ordered Array of input size: 3200 insertion sort takes meantime of : 0.1679

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Ordered Array of input size: 6400 insertion sort takes meantime of : 0.09666

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Ordered Array of input size: 12800 insertion sort takes meantime of : 0.30384

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Ordered Array of input size: 25600 insertion sort takes meantime of : 0.20813

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For PartiallyOrdered Array of input size: 100 insertion sort takes meantime of : 0.11334

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For PartiallyOrdered Array of input size: 200 insertion sort takes meantime of : 0.33304

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For PartiallyOrdered Array of input size: 400 insertion sort takes meantime of : 0.63213

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For PartiallyOrdered Array of input size: 800 insertion sort takes meantime of : 0.36543000000000003

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For PartiallyOrdered Array of input size: 1600 insertion sort takes meantime of : 1.08005

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For PartiallyOrdered Array of input size: 3200 insertion sort takes meantime of : 3.9644799999999996

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For PartiallyOrdered Array of input size: 6400 insertion sort takes meantime of : 18.21021

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:57 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For PartiallyOrdered Array of input size: 12800 insertion sort takes meantime of : 64.99396

---------------------------------------------------------------------------------------------------

2021-02-03 23:26:58 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For PartiallyOrdered Array of input size: 25600 insertion sort takes meantime of : 265.03508

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:01 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Random Array of input size: 100 insertion sort takes meantime of : 0.01462

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:01 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Random Array of input size: 200 insertion sort takes meantime of : 0.04858

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:01 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Random Array of input size: 400 insertion sort takes meantime of : 0.1784

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:01 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Random Array of input size: 800 insertion sort takes meantime of : 0.67229

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:01 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Random Array of input size: 1600 insertion sort takes meantime of : 2.58039

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:01 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Random Array of input size: 3200 insertion sort takes meantime of : 10.24742

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:01 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Random Array of input size: 6400 insertion sort takes meantime of : 41.7941

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:02 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Random Array of input size: 12800 insertion sort takes meantime of : 187.67301

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:04 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Random Array of input size: 25600 insertion sort takes meantime of : 837.8533299999999

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:14 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Reversed Array of input size: 100 insertion sort takes meantime of : 0.025879999999999997

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:14 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Reversed Array of input size: 200 insertion sort takes meantime of : 0.09218

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:14 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Reversed Array of input size: 400 insertion sort takes meantime of : 0.34345

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:14 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Reversed Array of input size: 800 insertion sort takes meantime of : 1.31698

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:14 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Reversed Array of input size: 1600 insertion sort takes meantime of : 5.16375

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:14 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Reversed Array of input size: 3200 insertion sort takes meantime of : 20.53088

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:14 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Reversed Array of input size: 6400 insertion sort takes meantime of : 81.89444999999999

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:15 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Reversed Array of input size: 12800 insertion sort takes meantime of : 327.55728

---------------------------------------------------------------------------------------------------

2021-02-03 23:27:19 INFO Benchmark\_Timer - Begin run: Benchmark test for InsertionSort with 10 runs

For Reversed Array of input size: 25600 insertion sort takes meantime of : 1332.59077

---------------------------------------------------------------------------------------------------

Summary of time taken by insertion sort (in milliseconds) to run on array orderings of four types: ordered array, partially ordered array, randomly ordered array and reverse ordered array are given below in the table with its corresponding graphical representation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Size (n)** | **Ordered Array** | **Partially Ordered Array** | **Random Array** | **Reversed Array** |
| 100 | 0.02452 | 0.11334 | 0.01462 | 0.025879 |
| 200 | 0.01601 | 0.33304 | 0.04858 | 0.09218 |
| 400 | 0.03269 | 0.63213 | 0.1784 | 0.34345 |
| 800 | 0.05853 | 0.36543 | 0.67229 | 1.31698 |
| 1600 | 0.10612 | 1.08005 | 2.58039 | 5.16375 |
| 3200 | 0.1679 | 3.964479 | 10.24742 | 20.53088 |
| 6400 | 0.09666 | 18.21021 | 41.7941 | 81.89444 |
| 12800 | 0.30384 | 64.99396 | 187.67301 | 327.55728 |
| 25600 | 0.20813 | 265.03508 | 837.85332 | 1332.59077 |

**Conclusion:**

As we can know that Insertion sort scans the array, compares each pair of elements and swaps elements if they are not in order. Since each operation contributes to the runtime of the algorithm, insertion sort can run in O(n) or O(n2 ) time depending on how the array to be sorted is ordered.

We make the following observations and conclusions from our timer and benchmark tests:

1. In the best-case scenario, insertion sort will simply compare elements in O(n) time and perform 0 swaps. Hence, the best-case scenario happens when the array is sorted, and insertion sort runs in O(n) time.

2. In the worst-case scenario, to insert the last element alone we will need n-1 comparisons and perform n-1 swaps. Similarly, for second last element, we will need n-2 comparisons and n-2 swaps, and so on.

The number of operations for the worst case in insertion sort is 2x (1 + 2 + ⋯ + n − 2 + n − 1)

Using the summation, we get: 2(n − 1) (n − 1 + 1) 2 = n (n − 1)

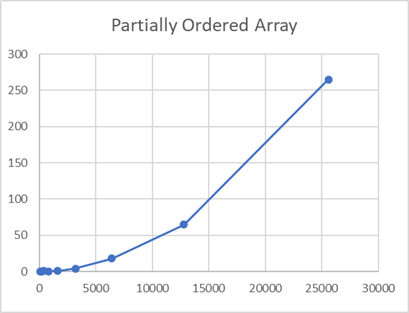
So, in the worst case, insertion sort takes O (n^2), that is, when the array is sorted in reverse order.

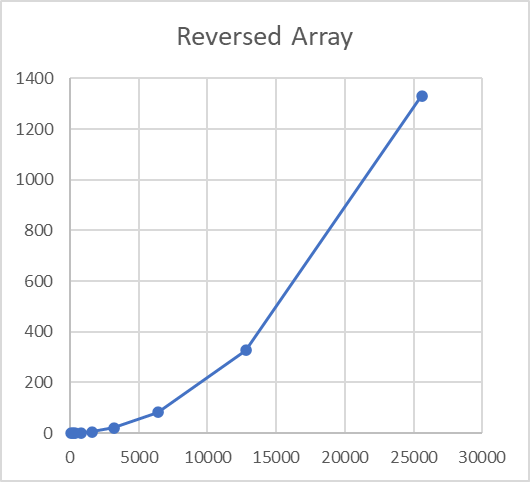
From the values of experiments, we observe that on an average (randomly ordered array, partially ordered array) insertion sort takes O (n^2) time approximately.

Analyzing the table values of the run time for each algorithm and its graph, we can conclude that the run time for insertion sort for different ordered arrays follows as

***Reverse Sorted Array > Randomly Ordered Array > Partially Ordered Array > Ordered Array***

3. Also from Observing the graphs if we find the slope of the line it would approx. be ≈ 2, giving the power of the relationship.





4. For each array types in the graphs above we can observe that the mean time increases with the increase of n. In addition, the relationship between them is approximate Meantime = k n where k is a constant.

Ordered array 

Reverse-Order array 

Random-Order array 

Partially-Order array 

**Screenshots of Tests:**

