INFO 6205 - Program Structure & Algorithms – Assignment 2

Algorithmic Benchmarking

Submitted By,
Aakash Shukla
shukla.aa@northeastern.edu

1. Introduction

Benchmarking is the method of comparing performance of algorithms with respect to certain predefined set of tests against it.

In our experiments we will be running the benchmark tests on an implementation of insertion sort and determine its performance in 4 major scenarios when the algorithm is fed – sorted input, input that is sorted in descending order, random input and semi sorted input. We will be comparing the runtime of the algorithm on an initial set of 100 values and will go on doubling the input size. We shall be running each cycle at least 100 times and calculate the mean runtime in case of every run. The system clock time will be recorded using the System.nanoTime() directive in java.

2. Insertion Sort

Insertion works like one would sort cards in hand. Smallest valued cards are repeatedly pushed to one end repeatedly till they are in desired order.

Algorithm:

- Iterate from inputArray[1] -> inputArray[n].
- Compare current element to the previous element in array.
- Switch if it is smaller than the previous and then compare with the previous items of the array (if any).

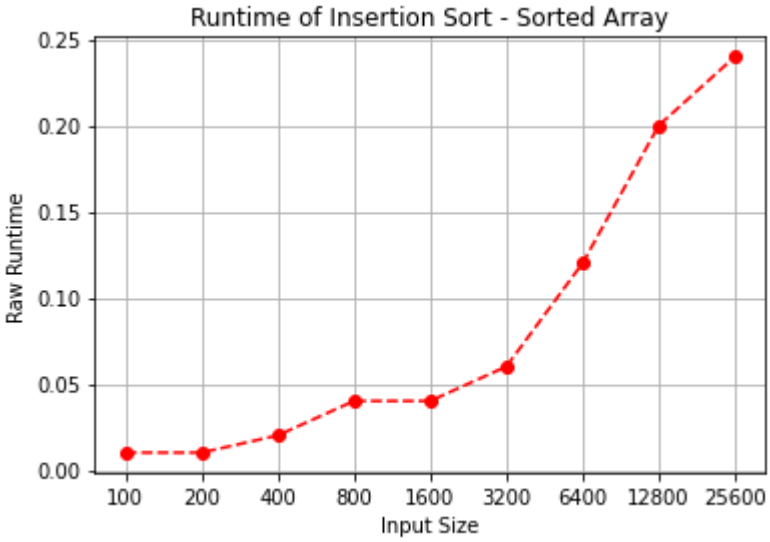
3. Benchmarks

- **Sorted Array**

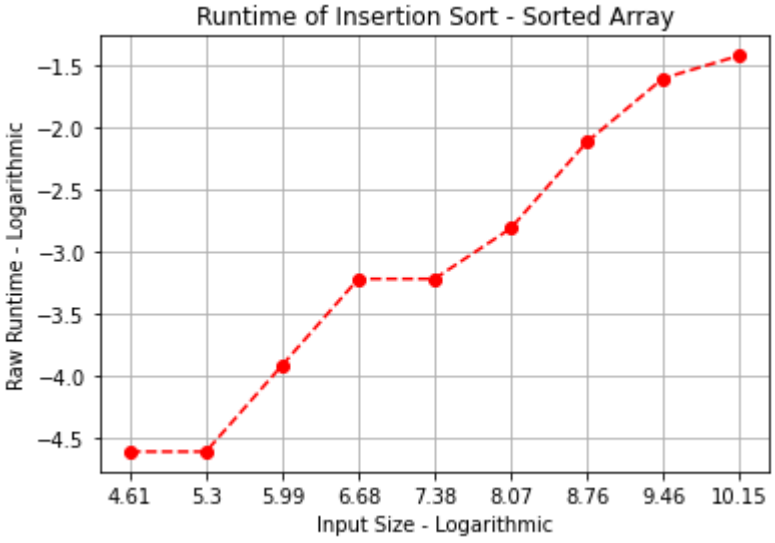
  By convention this should be the fastest as the input array is already sorted.

  The following table documents the run time of the algorithm on various input sizes and its normalized runtime.

Input Size	Raw time per run (mSec)	Normalized time per run
100	0.01	3.18
200	0.01	1.73
400	0.02	1.34
800	0.04	0.51
1600	0.04	0.34
3200	0.06	0.31
6400	0.12	0.28
12800	0.20	0.26
25600	0.24	0.10

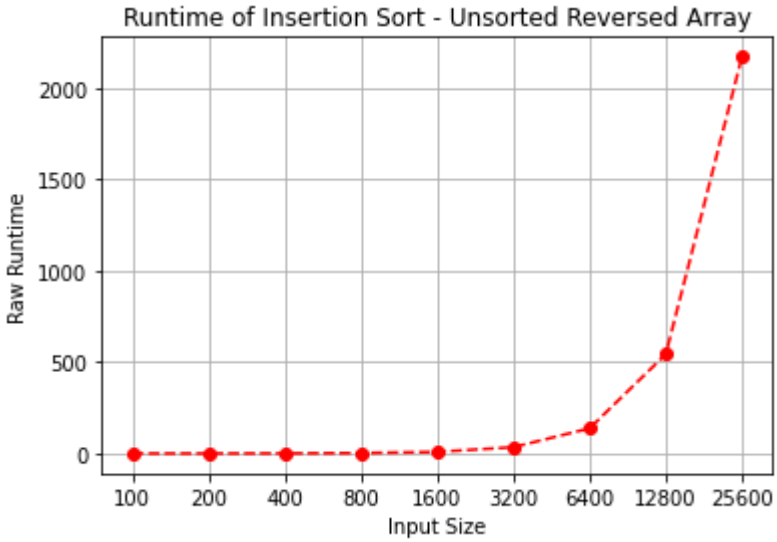

Runtime of Insertion Sort - Sorted Array
Raw Runtime
0.25
0.20
0.15
0.10
0.05
0.00
100
200
400
800
1600
3200
6400
12800
25600
Input Size

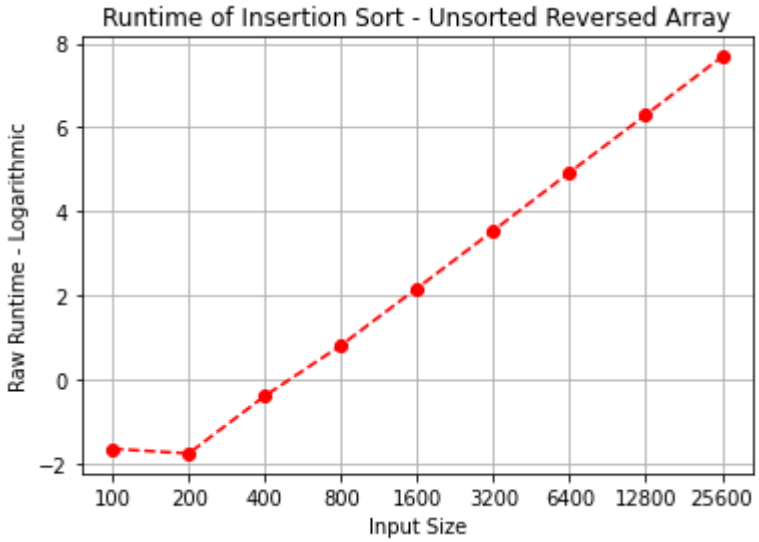

Runtime of Insertion Sort - Sorted Array
Raw Runtime - Logarithmic
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
−4.5
100
200
400
800
1600
3200
6400
12800
25600
Input Size

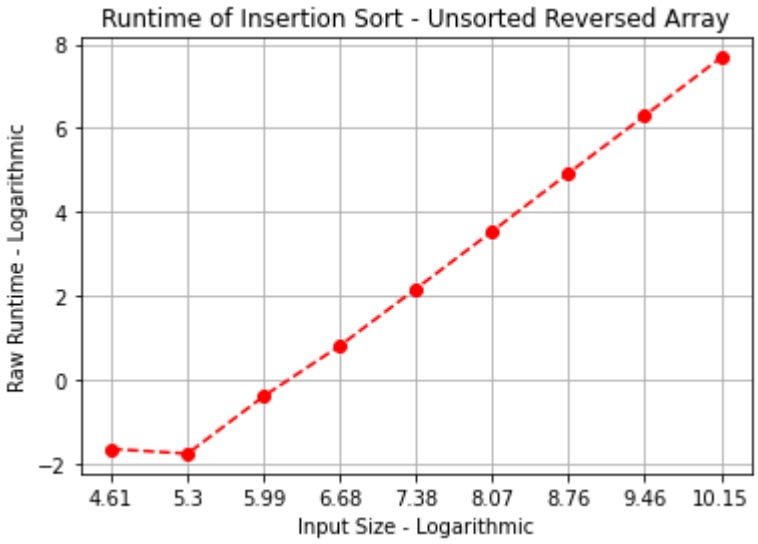

Runtime of Insertion Sort - Sorted Array
Raw Runtime - Logarithmic
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
−4.5
4.61
5.3
5.99
6.68
7.38
8.07
8.76
9.46
10.15
Input Size - Logarithmic

- **Unsorted Array**
  We will be considering an array that as been reversed for this case. The run times have been documented below.

Input Size	Raw time per run (mSec)	Normalized time per run
100	0.19	60.34
200	0.17	22.75
400	0.67	38.69
800	2.24	56.85
1600	8.75	99.01
3200	34.46	174.78
6400	137.70	320.11
12800	543.47	579.95
25600	2171.90	1071.09



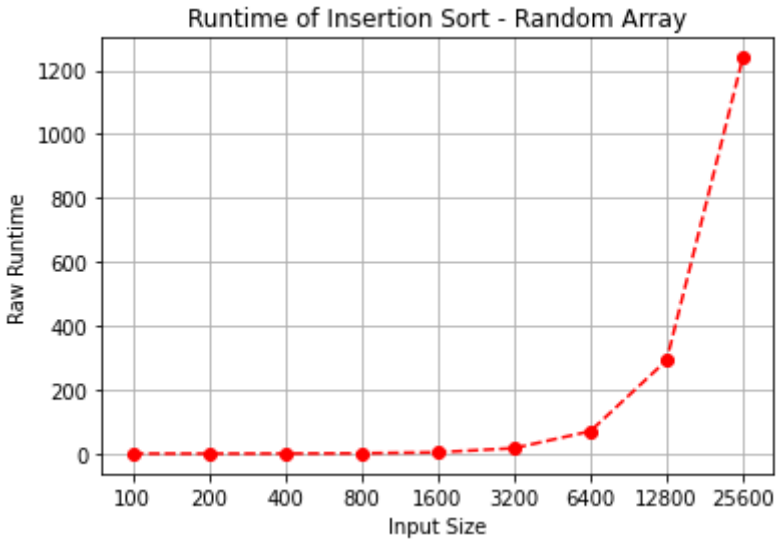


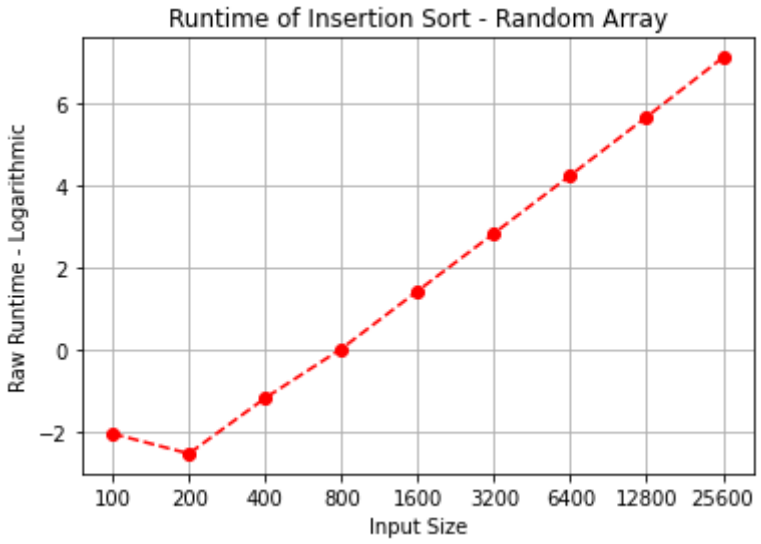


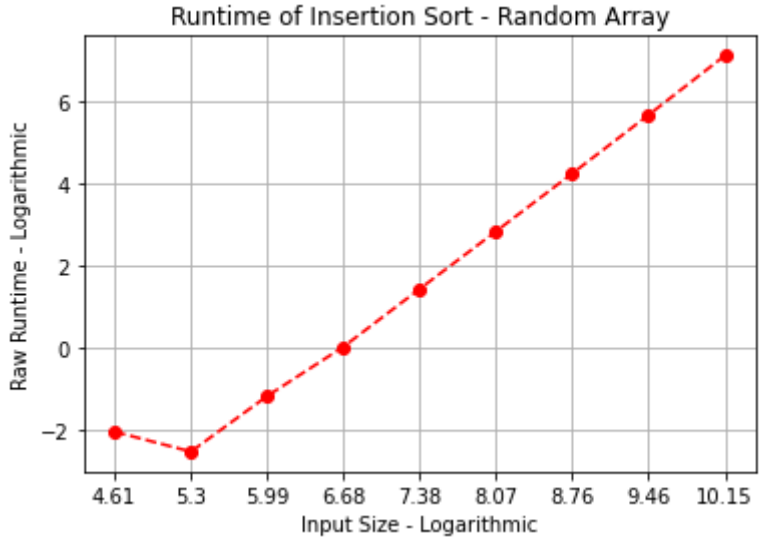
- **Radom Array**

  We will be considering a random array for this case. We will be using the java's in-built random number generator for this test case.

Input Size	Raw time per run (mSec)	Normalized time per run
100	0.13	41.29
200	0.08	10.71
400	0.31	17.90
800	1.03	26.14
1600	4.22	47.75
3200	17.16	87.59
6400	70.26	163.33
12800	292.20	311.81
25600	1239.82	611.43



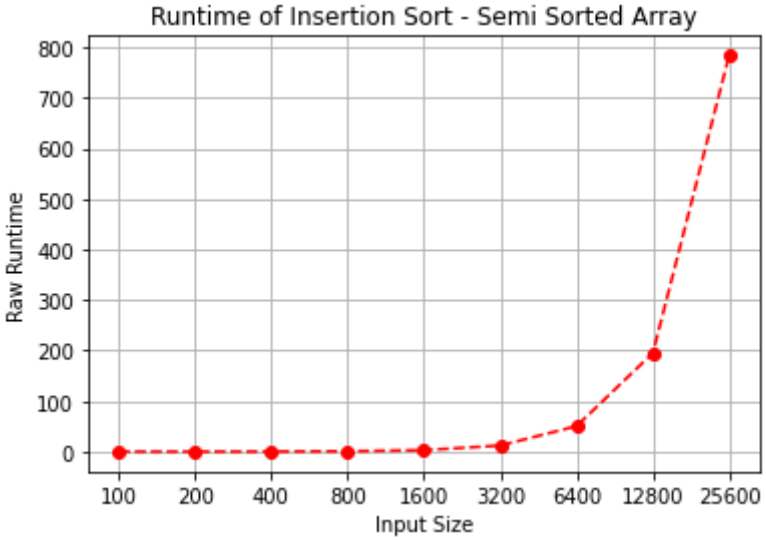


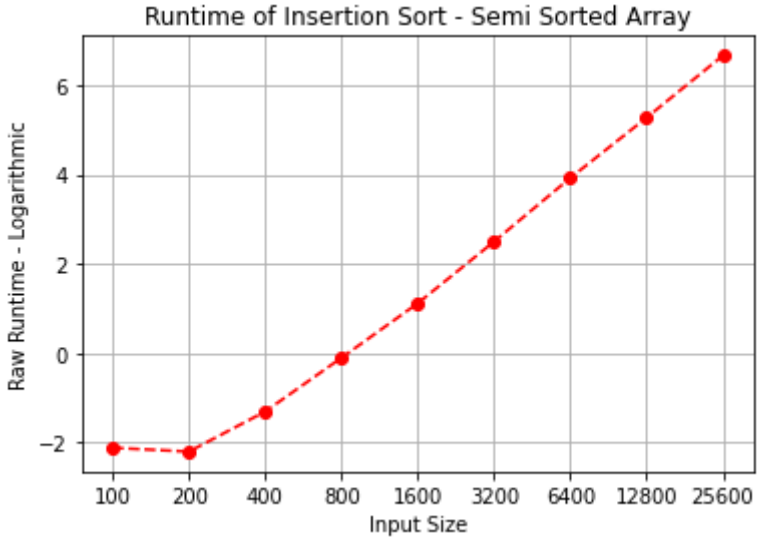


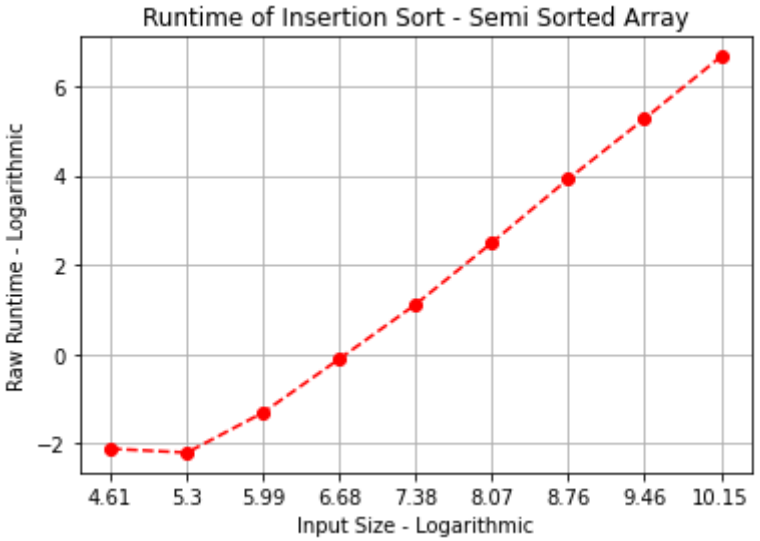
- **Semi sorted array**

  For this case we have taken an array where first n/2 numbers will be sorted, and the remaining n/2 numbers will be randomly generated. Here n in the size of the input.

Input Size	Raw time per run (mSec)	Normalized time per run
100	0.12	38.11
200	0.11	14.72
400	0.27	15.59
800	0.90	22.84
1600	3.05	34.51
3200	12.15	62.02
6400	50.56	117.53
12800	195.04	208.13
25600	783.40	386.34


Runtime of Insertion Sort - Semi Sorted Array
Raw Runtime
Input Size
0
100
200
300
400
500
600
700
800
100
200
400
800
1600
3200
6400
12800
25600

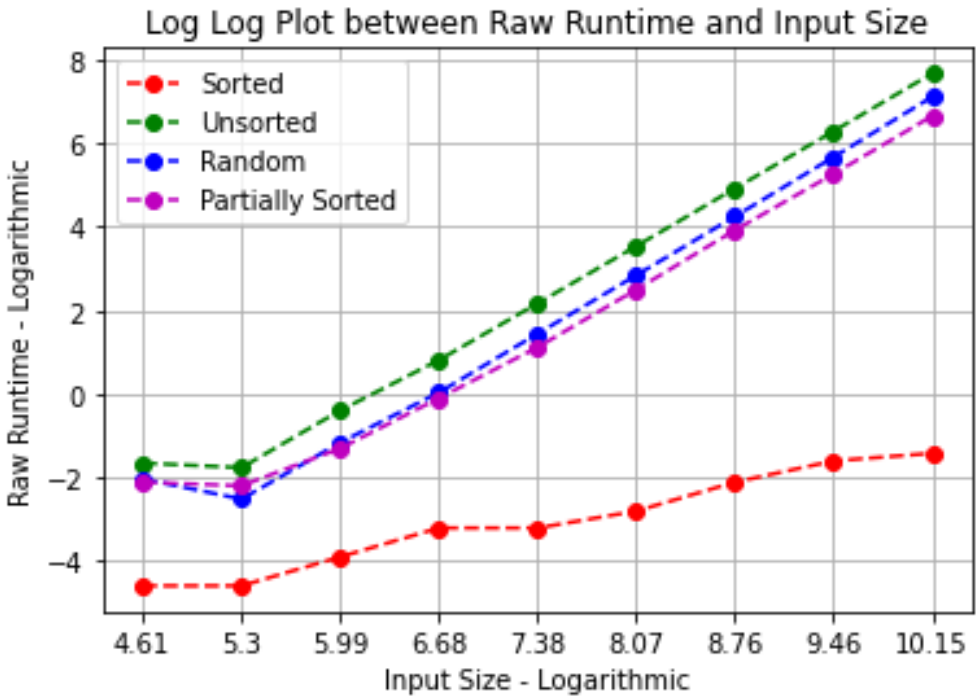

Runtime of Insertion Sort - Semi Sorted Array
Raw Runtime - Logarithmic
Input Size
−2
0
2
4
6
100
200
400
800
1600
3200
6400
12800
25600

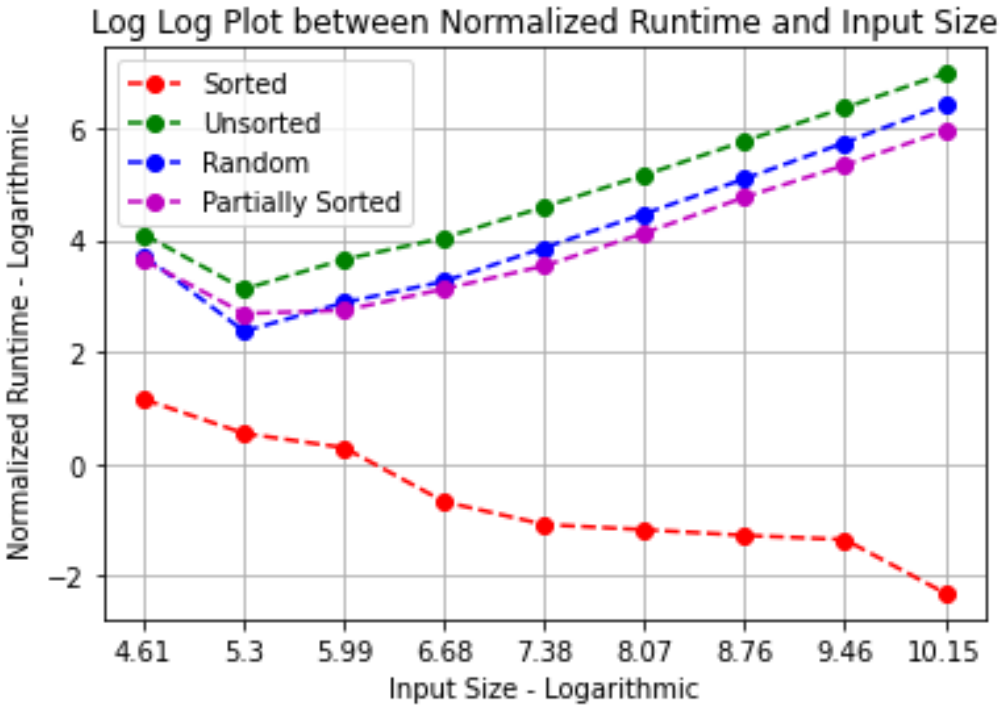

Runtime of Insertion Sort - Semi Sorted Array
Raw Runtime - Logarithmic
Input Size - Logarithmic
−2
0
2
4
6
4.61
5.3
5.99
6.68
7.38
8.07
8.76
9.46
10.15

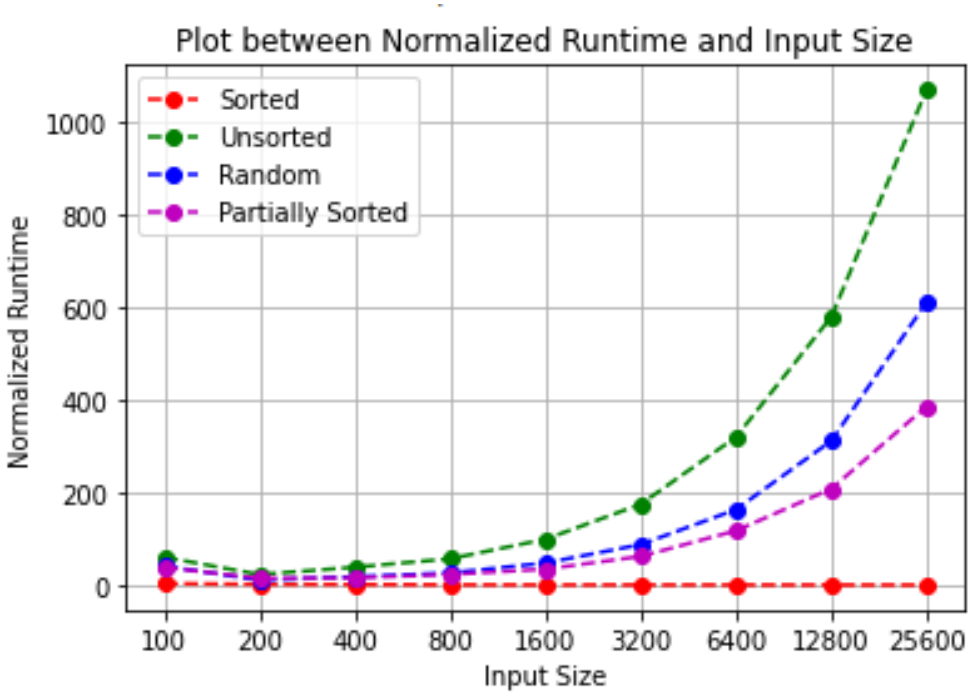
4. Comparisons Between Different Types of Arrays

As we see from the graphs below, sorted array takes minimum time followed by semi sorted array, random array and finally reversed array.

Sorted Array < Partially Sorted Array < Random Array < Unsorted Array

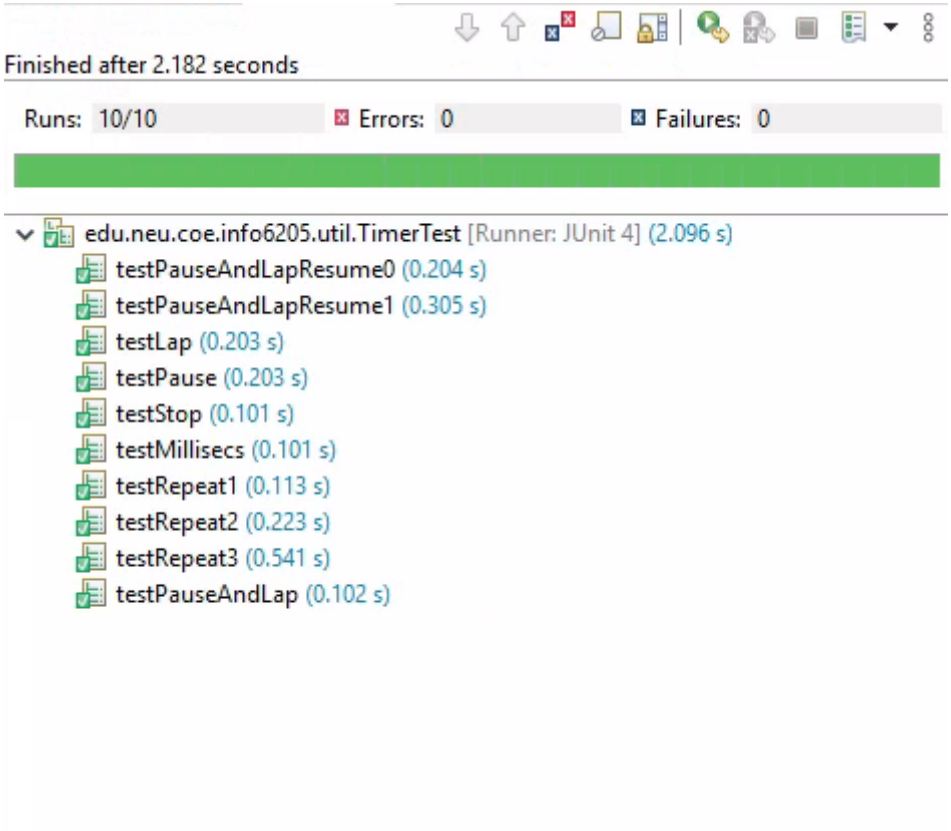




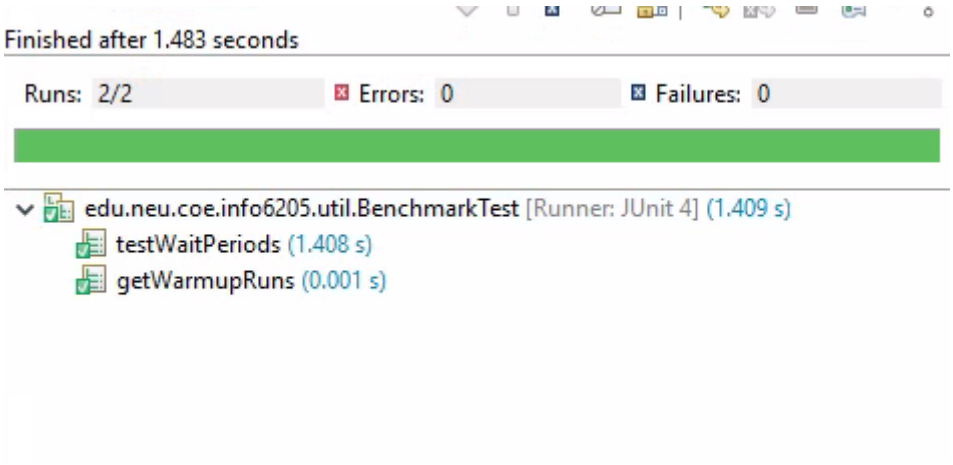


5. Unit Tests

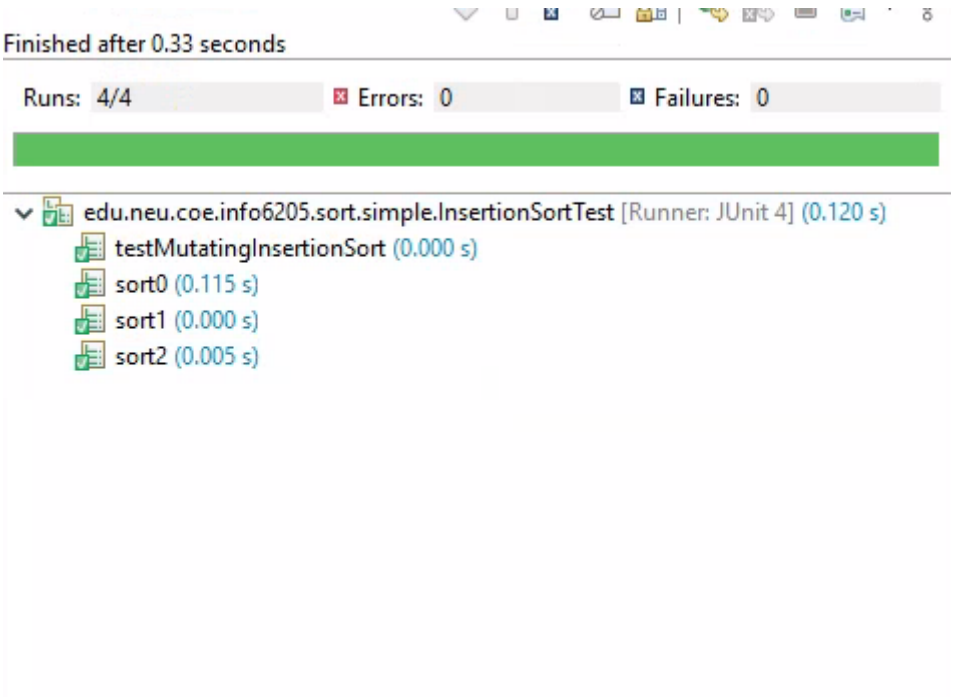
Screenshots from successful unit tests are given below



TimeTest.java



BenchmarkTest.java



InsertionSortTest.java