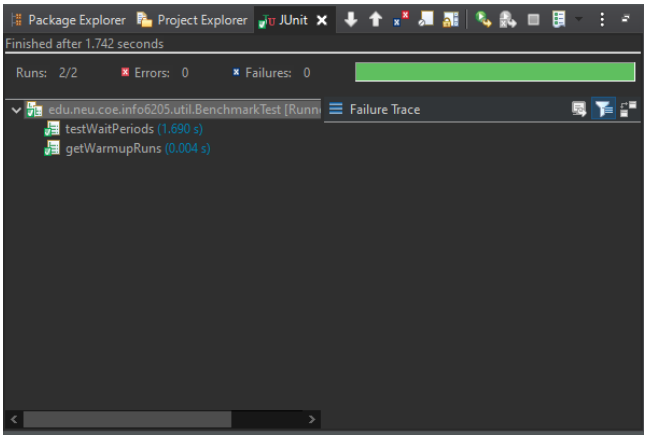
Assignment – 2

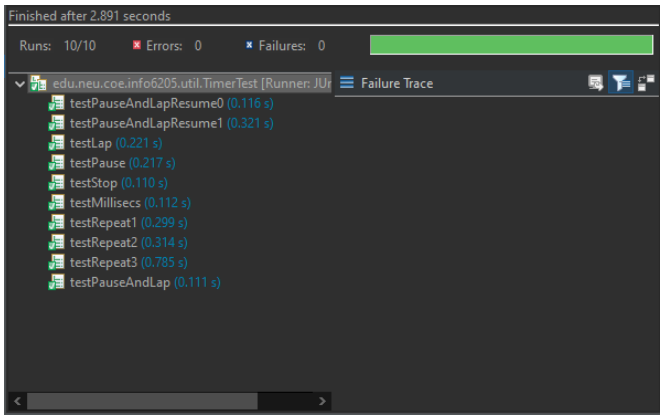
Program Structure and Algorithms

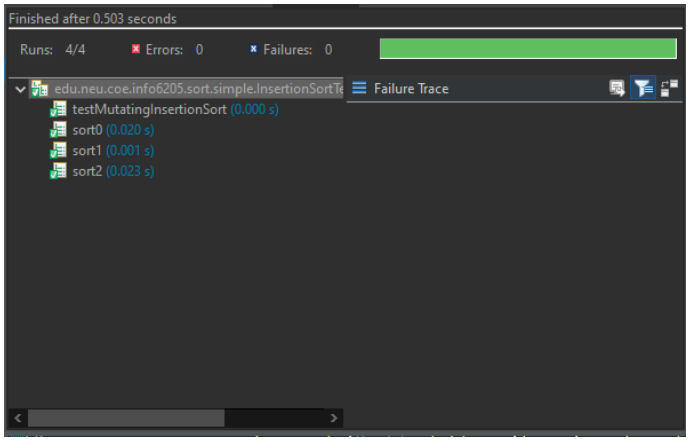
Pranav Agarwal

NUID – 001099801

1. Unit Tests –



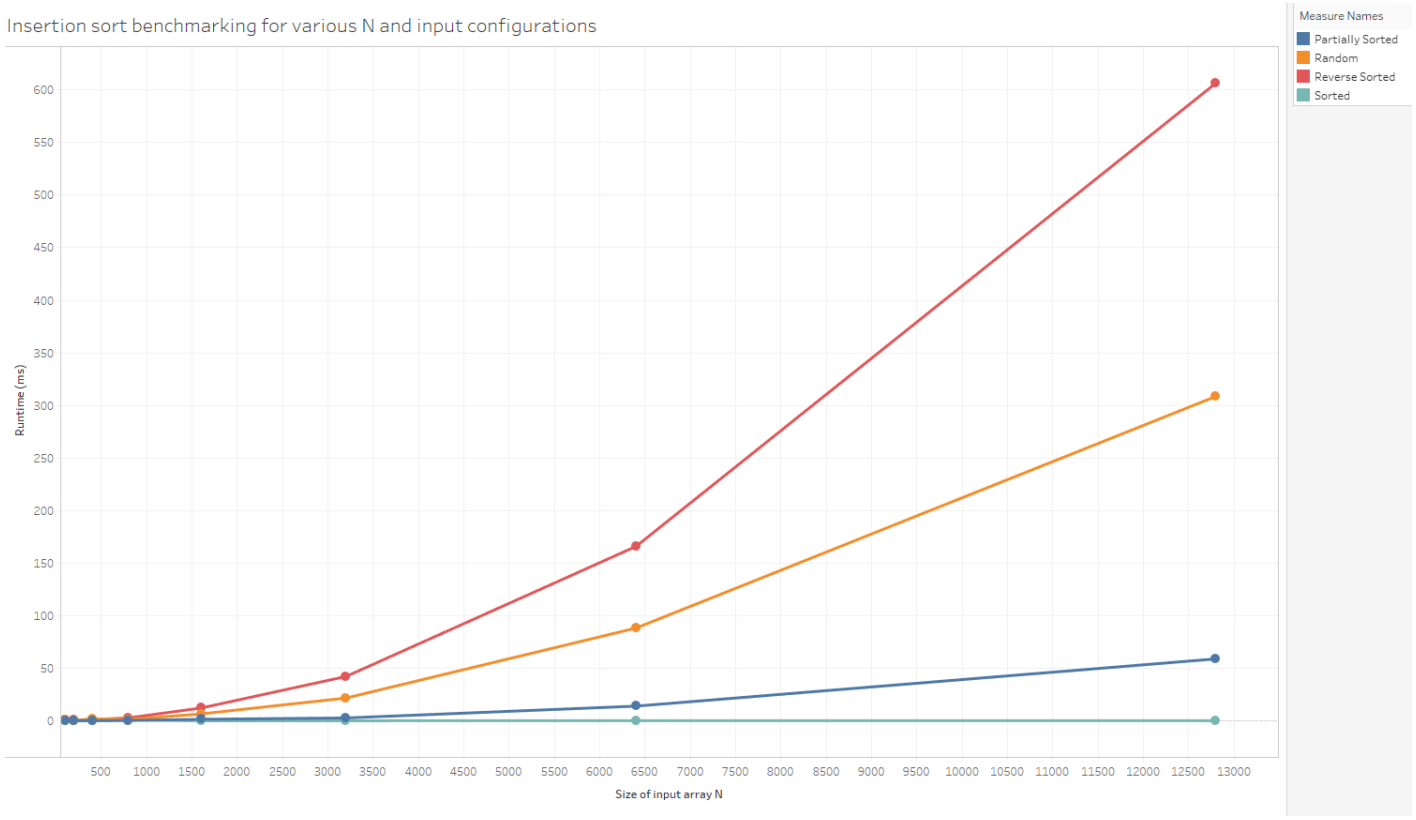




2. Benchmark Results –

# data.csv N	# data.csv Random	# data.csv Sorted	# data.csv Reverse Sorted	# data.csv Partially Sorted
100	0.820	0.009780	0.728	0.0998
200	0.405	0.012270	0.650	0.0989
400	1.962	0.043630	1.162	0.0568
800	1.287	0.621520	2.853	0.2793
1,600	6.577	0.032930	12.178	1.4696
3,200	21.697	0.061360	42.209	2.7336
6,400	88.229	0.087080	165.919	13.9690
12,800	308.597	0.131200	606.528	58.8911

(time in ms)



3. Observations from table and graphs–

- Random and reverse sorted grow as approximately N^2
- Time taken by random (average case) is approximately half of reverse sorted (worst case), as expected.
- Time taken by partially sorted is approximately 10-15% of that of reverse sorted.
- Time taken for sorted array (best case) is **significantly** better than average and worst cases

4. Notes regarding assignment –

- Partially sorted array was generated by sorting first 90% of random array
- Each cell in the table is the average time of 10 runs.
- The input for each run is identical i.e. the random array is generated only once per N. This is to ensure better benchmarking by removing variable factors.
- The partially sorted, reverse sorted and reverse sorted arrays are permutations of the same random array that is used for the random benchmark. Again, this is to ensure the numbers stay same and the difference in times is purely due to different inversion count.
- You can run the code by running the main class in Assignment2.java in the utils folder. A csv file containing results will be automatically generated.
- Output of the benchmark is stored in results/insertion_sort_benchmark/data.csv
- You can edit the parameters of the run by changing the constants in the assignment file.