Urvi V. Aryamane (001040582)
Program Structures and Algorithms
Spring 2021(SEC 05)

Task:

Part 1) To implement three methods of a class called Timer. Timer is invoked from a class called Benchmark_Timer which implements the Benchmark interface.

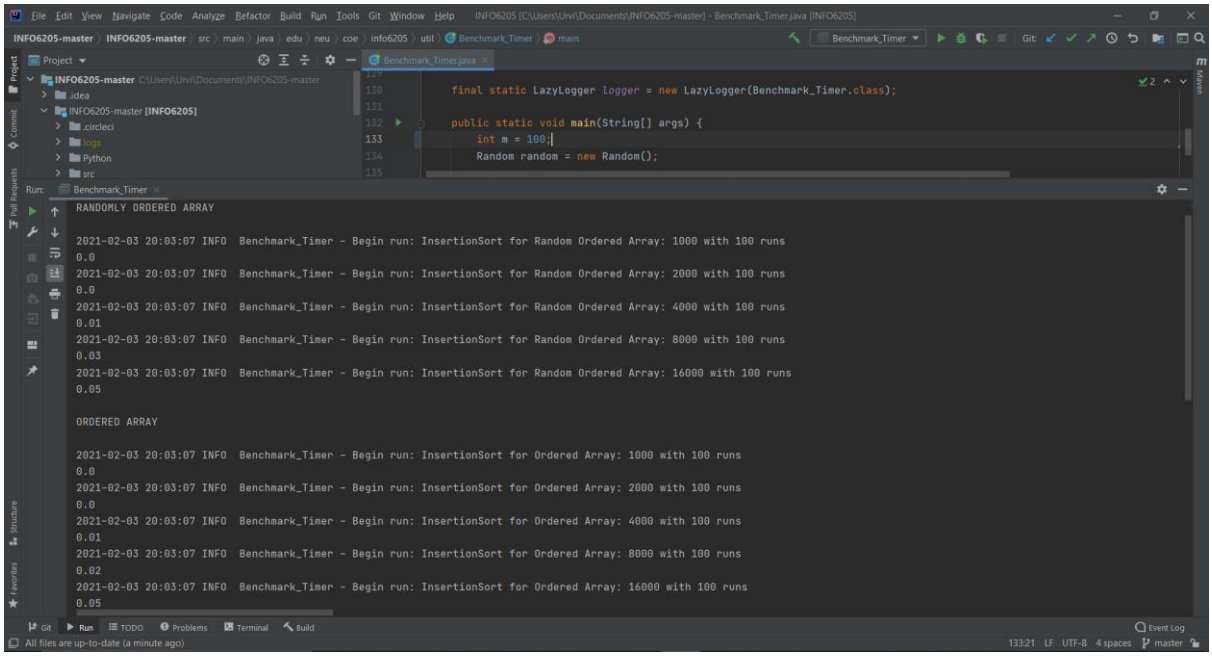
Part 2) Implement InsertionSort (in the InsertionSort class) by simply looking up the insertion code used by Arrays.sort by using the helper.swap

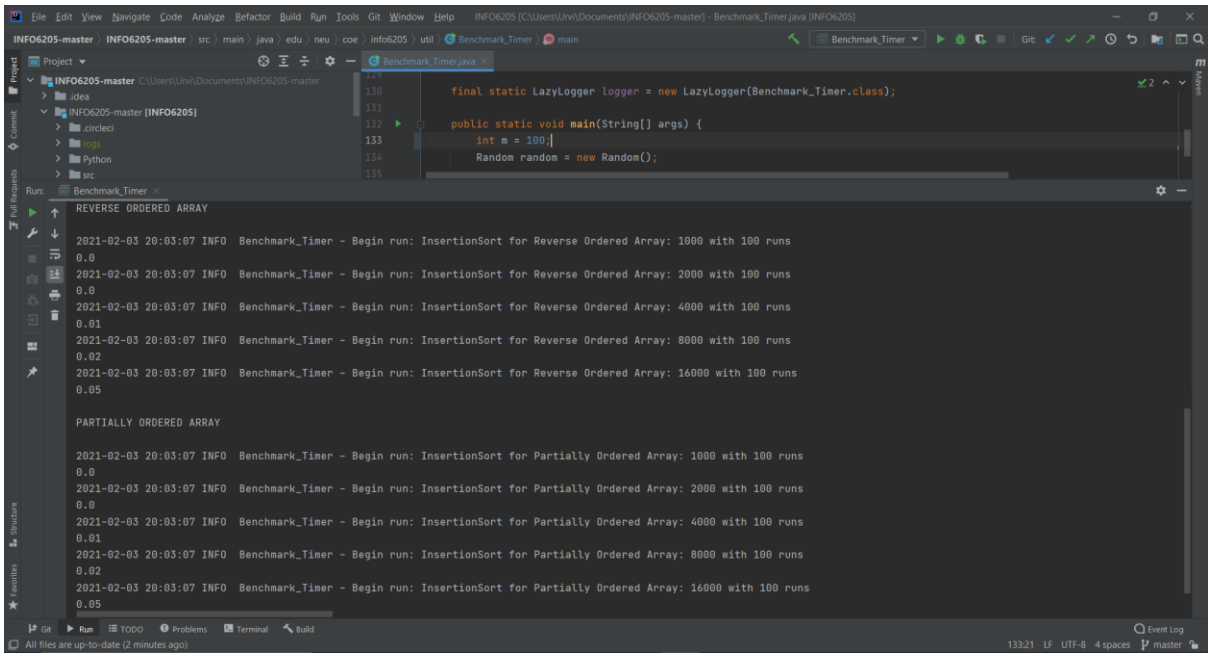
Part 3) Implement a main program to actually run the following benchmarks: measure the running times of this sort, using four different initial array ordering situations: random, ordered, partially-ordered and reverse-ordered.

Relationship Conclusion:

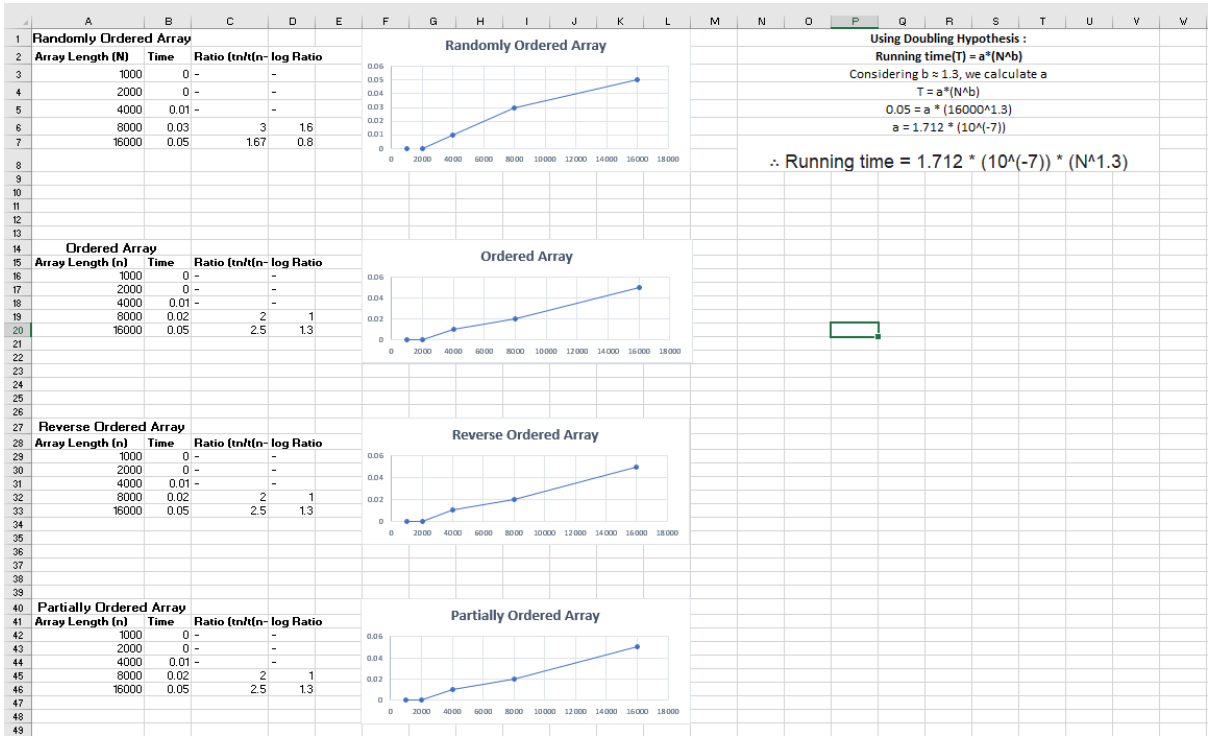
Using Doubling Hypothesis:
T = a * (N^b)
Where,
T = running time
Therefore, order of growth N ≈ 1.3

Evidence to support that conclusion:





Graphical Representation



Randomly Ordered Array

Array Length (N)	Time	Ratio (tn/t(n-	log Ratio
1000	0	-	-
2000	0	-	-
4000	0.01	-	-
8000	0.03	3	1.6
16000	0.05	1.67	0.8

Ordered Array

Array Length (n)	Time	Ratio (tn/t(n-	log Ratio
1000	0	-	-
2000	0	-	-
4000	0.01	-	-
8000	0.02	2	1
16000	0.05	2.5	1.3

Reverse Ordered Array

Array Length (n)	Time	Ratio (tn/t(n-	log Ratio
1000	0	-	-
2000	0	-	-
4000	0.01	-	-
8000	0.02	2	1
16000	0.05	2.5	1.3

Partially Ordered Array

Array Length (n)	Time	Ratio (tn/t(n-	log Ratio
1000	0	-	-
2000	0	-	-
4000	0.01	-	-
8000	0.02	2	1
16000	0.05	2.5	1.3

Using Doubling Hypothesis :

Running time(T) = a*(N^b)

Considering b ≈ 1.3, we calculate a

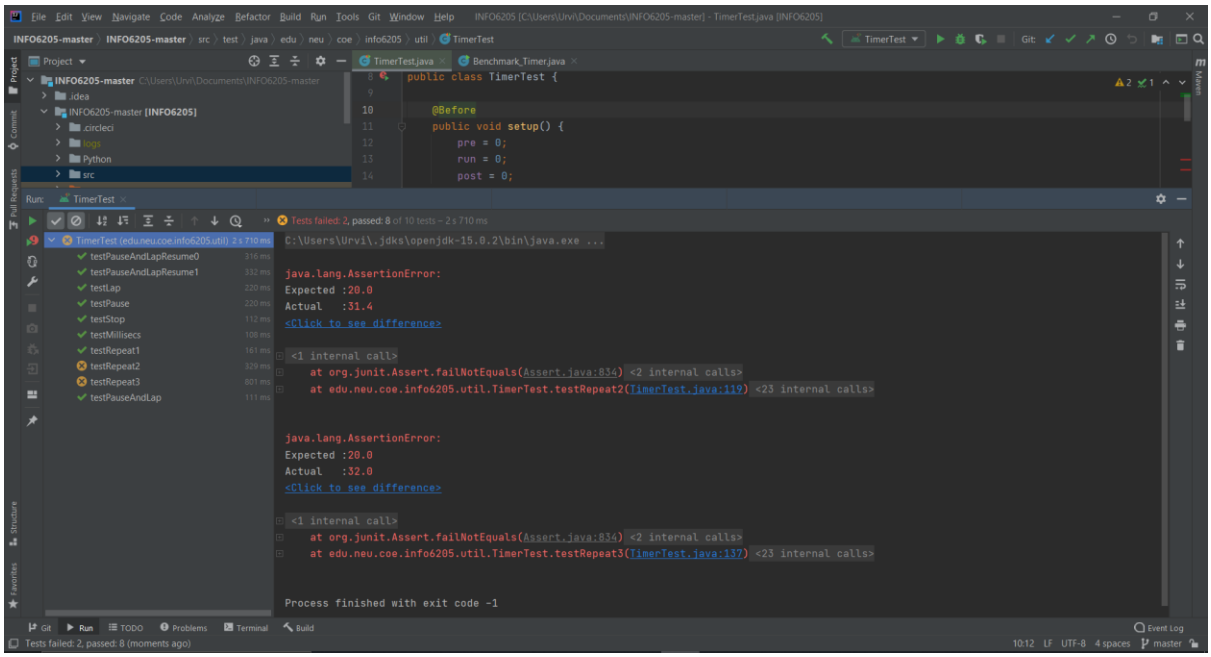
T = a*(N^b)

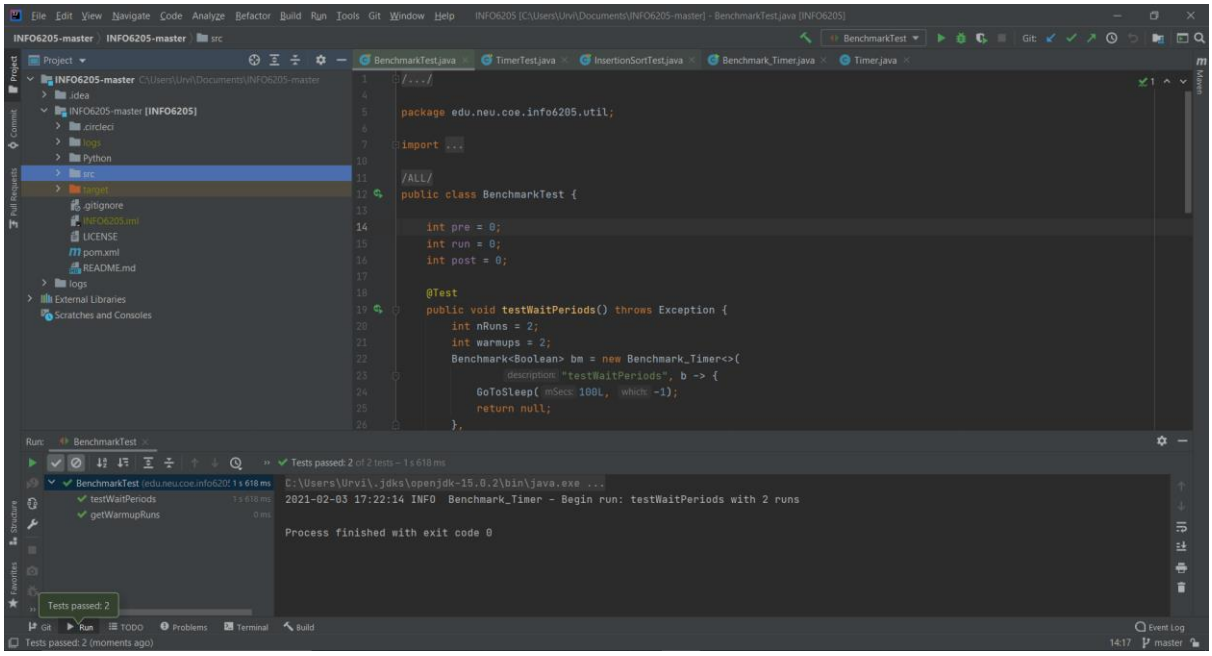
0.05 = a * (16000^1.3)

a = 1.712 * (10^(-7))

∴ Running time = 1.712 * (10^(-7)) * (N^1.3)

Unit Test Screenshots
Part 1: Timer Test





Part 2: Insertion Sort