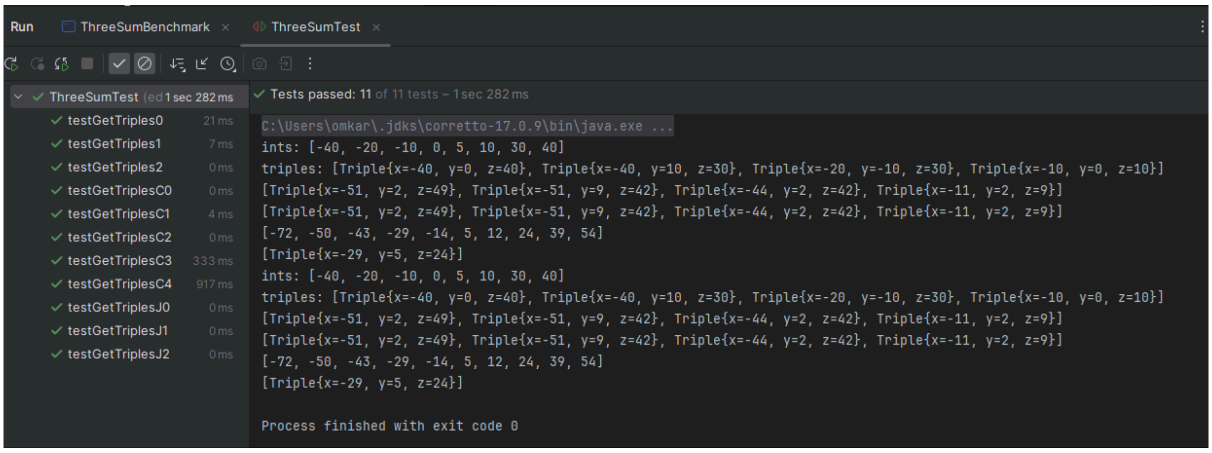
Program Structures and Algorithms
Spring 2024

NAME: Omkar Vaity
NUID: 002836207
GITHUB LINK: https://github.com/OmkarVaity/INFO6205

Task: Assignment 2 (3-SUM)

(a) evidence (screenshot) of your unit tests running (try to show the actual unit test code as well as the green strip);

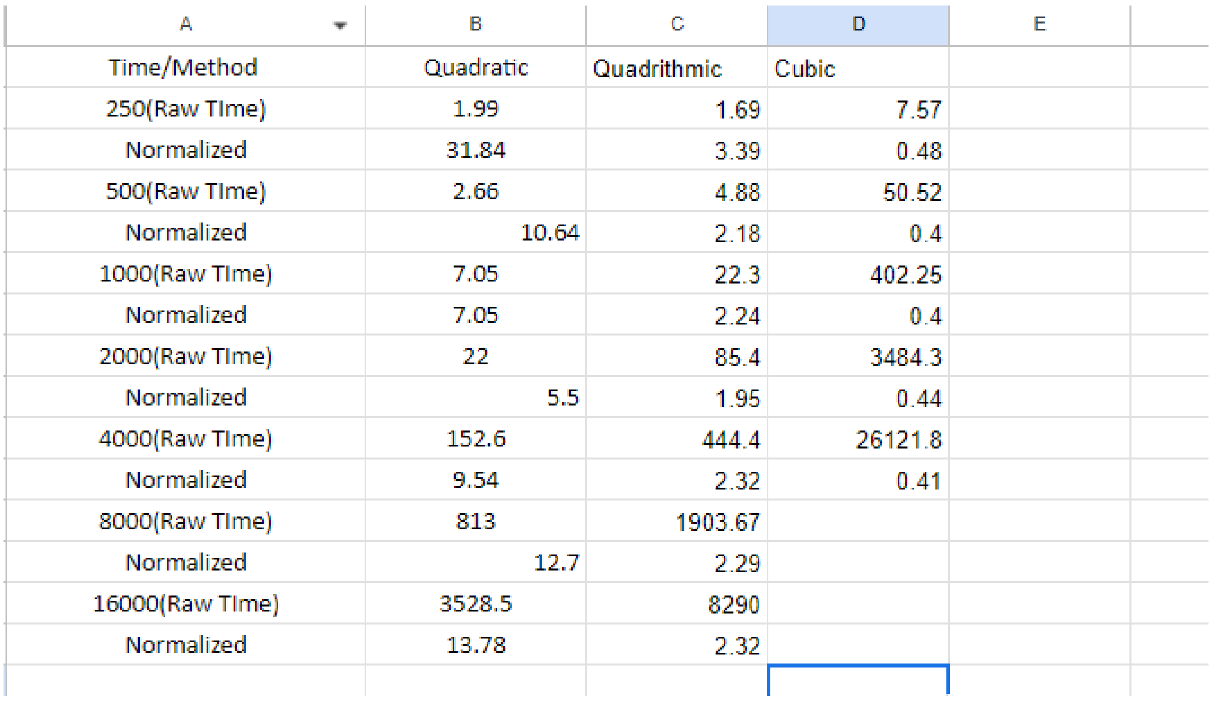


```
ThreeSumTest (ed 1 sec 282 ms
  testGetTriples0    21 ms
  testGetTriples1     7 ms
  testGetTriples2     0 ms
  testGetTriplesC0    0 ms
  testGetTriplesC1    4 ms
  testGetTriplesC2    0 ms
  testGetTriplesC3  333 ms
  testGetTriplesC4  917 ms
  testGetTriplesJ0    0 ms
  testGetTriplesJ1    0 ms
  testGetTriplesJ2    0 ms
```

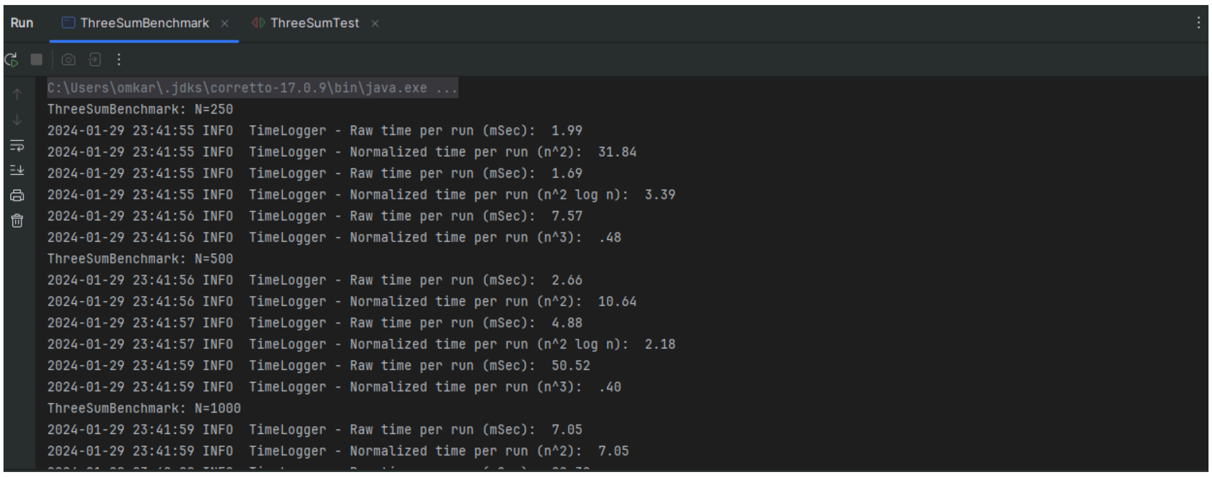
```
Tests passed: 11 of 11 tests – 1 sec 282 ms
C:\Users\omkar\.jdks\corretto-17.0.9\bin\java.exe ...
ints: [-40, -20, -10, 0, 5, 10, 30, 40]
triples: [Triple{x=-40, y=0, z=40}, Triple{x=-40, y=10, z=30}, Triple{x=-20, y=-10, z=30}, Triple{x=-10, y=0, z=10}]
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2, z=42}, Triple{x=-11, y=2, z=9}]
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2, z=42}, Triple{x=-11, y=2, z=9}]
[-72, -50, -43, -29, -14, 5, 12, 24, 39, 54]
[Triple{x=-29, y=5, z=24}]
ints: [-40, -20, -10, 0, 5, 10, 30, 40]
triples: [Triple{x=-40, y=0, z=40}, Triple{x=-40, y=10, z=30}, Triple{x=-20, y=-10, z=30}, Triple{x=-10, y=0, z=10}]
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2, z=42}, Triple{x=-11, y=2, z=9}]
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2, z=42}, Triple{x=-11, y=2, z=9}]
[-72, -50, -43, -29, -14, 5, 12, 24, 39, 54]
[Triple{x=-29, y=5, z=24}]

Process finished with exit code 0
```

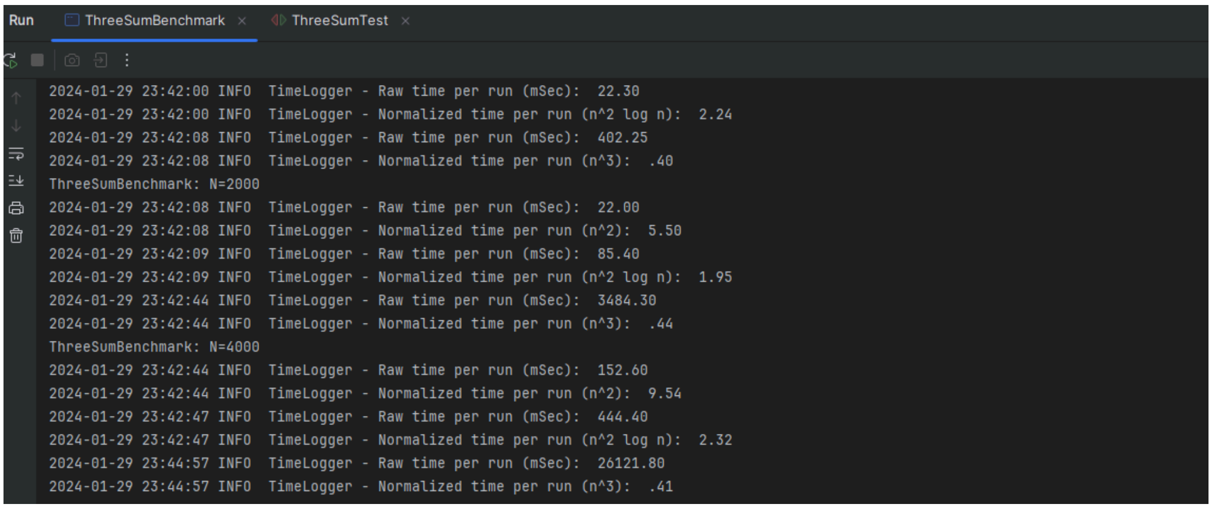
b) a spreadsheet showing your timing observations



A	B	C	D	E
Time/Method	Quadratic	Quadrithmic	Cubic	
250(Raw TIme)	1.99	1.69	7.57	
Normalized	31.84	3.39	0.48	
500(Raw TIme)	2.66	4.88	50.52	
Normalized	10.64	2.18	0.4	
1000(Raw TIme)	7.05	22.3	402.25	
Normalized	7.05	2.24	0.4	
2000(Raw TIme)	22	85.4	3484.3	
Normalized	5.5	1.95	0.44	
4000(Raw TIme)	152.6	444.4	26121.8	
Normalized	9.54	2.32	0.41	
8000(Raw TIme)	813	1903.67		
Normalized	12.7	2.29		
16000(Raw TIme)	3528.5	8290		
Normalized	13.78	2.32		



```
Run   ThreeSumBenchmark ×   ThreeSumTest ×
C:\Users\omkar\.jdks\corretto-17.0.9\bin\java.exe ...
ThreeSumBenchmark: N=250
2024-01-29 23:41:55 INFO  TimeLogger - Raw time per run (mSec):  1.99
2024-01-29 23:41:55 INFO  TimeLogger - Normalized time per run (n^2):  31.84
2024-01-29 23:41:55 INFO  TimeLogger - Raw time per run (mSec):  1.69
2024-01-29 23:41:55 INFO  TimeLogger - Normalized time per run (n^2 log n):  3.39
2024-01-29 23:41:56 INFO  TimeLogger - Raw time per run (mSec):  7.57
2024-01-29 23:41:56 INFO  TimeLogger - Normalized time per run (n^3):  .48
ThreeSumBenchmark: N=500
2024-01-29 23:41:56 INFO  TimeLogger - Raw time per run (mSec):  2.66
2024-01-29 23:41:56 INFO  TimeLogger - Normalized time per run (n^2):  10.64
2024-01-29 23:41:57 INFO  TimeLogger - Raw time per run (mSec):  4.88
2024-01-29 23:41:57 INFO  TimeLogger - Normalized time per run (n^2 log n):  2.18
2024-01-29 23:41:59 INFO  TimeLogger - Raw time per run (mSec):  50.52
2024-01-29 23:41:59 INFO  TimeLogger - Normalized time per run (n^3):  .40
ThreeSumBenchmark: N=1000
2024-01-29 23:41:59 INFO  TimeLogger - Raw time per run (mSec):  7.05
2024-01-29 23:41:59 INFO  TimeLogger - Normalized time per run (n^2):  7.05
```



```
Run   ThreeSumBenchmark ×   ThreeSumTest ×
2024-01-29 23:42:00 INFO  TimeLogger - Raw time per run (mSec):  22.30
2024-01-29 23:42:00 INFO  TimeLogger - Normalized time per run (n^2 log n):  2.24
2024-01-29 23:42:08 INFO  TimeLogger - Raw time per run (mSec):  402.25
2024-01-29 23:42:08 INFO  TimeLogger - Normalized time per run (n^3):  .40
ThreeSumBenchmark: N=2000
2024-01-29 23:42:08 INFO  TimeLogger - Raw time per run (mSec):  22.00
2024-01-29 23:42:08 INFO  TimeLogger - Normalized time per run (n^2):  5.50
2024-01-29 23:42:09 INFO  TimeLogger - Raw time per run (mSec):  85.40
2024-01-29 23:42:09 INFO  TimeLogger - Normalized time per run (n^2 log n):  1.95
2024-01-29 23:42:44 INFO  TimeLogger - Raw time per run (mSec):  3484.30
2024-01-29 23:42:44 INFO  TimeLogger - Normalized time per run (n^3):  .44
ThreeSumBenchmark: N=4000
2024-01-29 23:42:44 INFO  TimeLogger - Raw time per run (mSec):  152.60
2024-01-29 23:42:44 INFO  TimeLogger - Normalized time per run (n^2):  9.54
2024-01-29 23:42:47 INFO  TimeLogger - Raw time per run (mSec):  444.40
2024-01-29 23:42:47 INFO  TimeLogger - Normalized time per run (n^2 log n):  2.32
2024-01-29 23:44:57 INFO  TimeLogger - Raw time per run (mSec):  26121.80
2024-01-29 23:44:57 INFO  TimeLogger - Normalized time per run (n^3):  .41
```

(c) your brief explanation of why the quadratic method(s) work.

The Quadratic Method for finding triplets in an array that add up to zero relies on the fact that the input array is sorted. Imagine you have a list of numbers, and you want to find groups of three numbers (triplets) where the sum is zero. The trick is to look at each number in the list one by one and find two other numbers around it so that when you add them up, you get zero. The smart part is that since the list is sorted, you can divide it into two parts around the number you're looking at. One part has smaller numbers on the left, and the other part has larger numbers on the right. This makes it easier to figure out which two numbers, when added to the current one, will give you zero. So, you start with a middle number and try to find two other numbers around it. If the sum of those three is too big, you know you need to make it smaller, so you look at smaller numbers. If the sum is too small, you look at larger numbers to make it bigger. This way, you go through the list just once, adjusting the numbers as needed, and quickly find all the triplets that add up to zero. This method is faster than other ways because it avoids using a nested loop and only goes through the list once, making it more efficient.