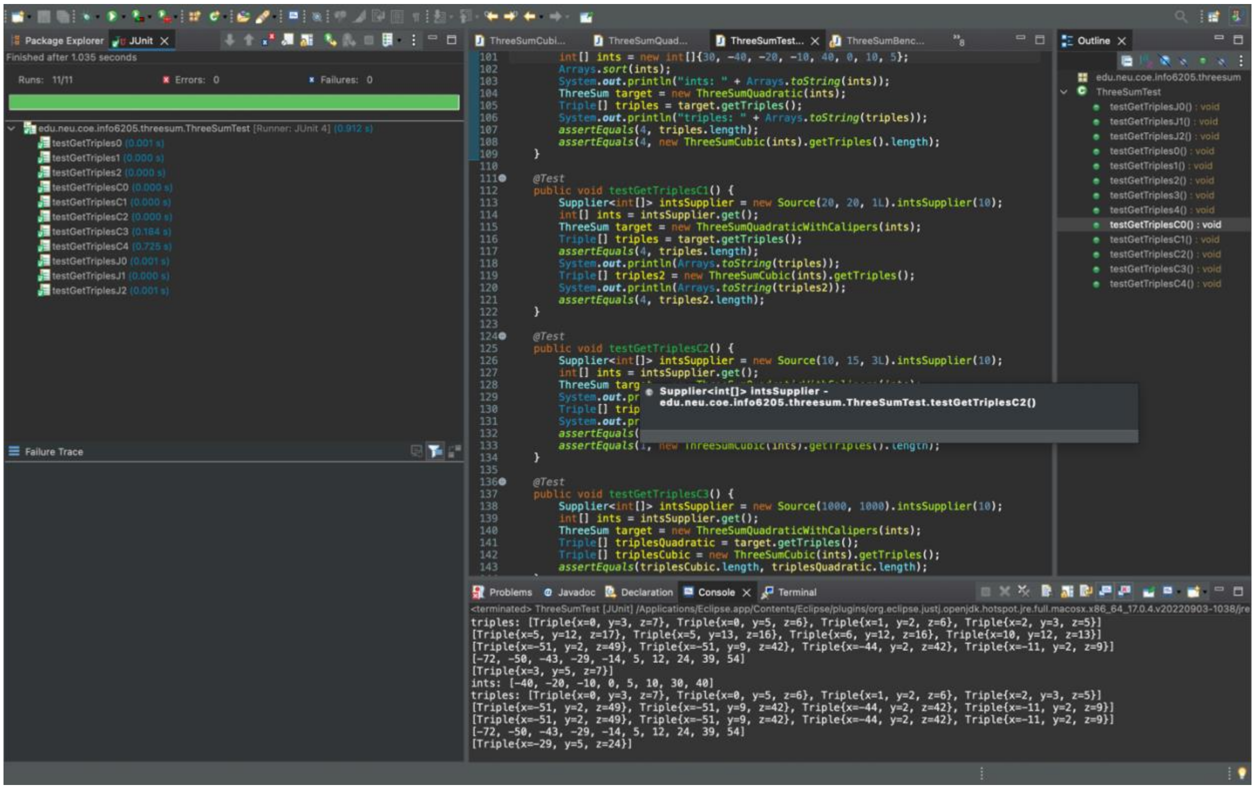
Program Structures and Algorithms
Spring 2023 Section – 01
Name – Naman Diwan
NUID – 002724115

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

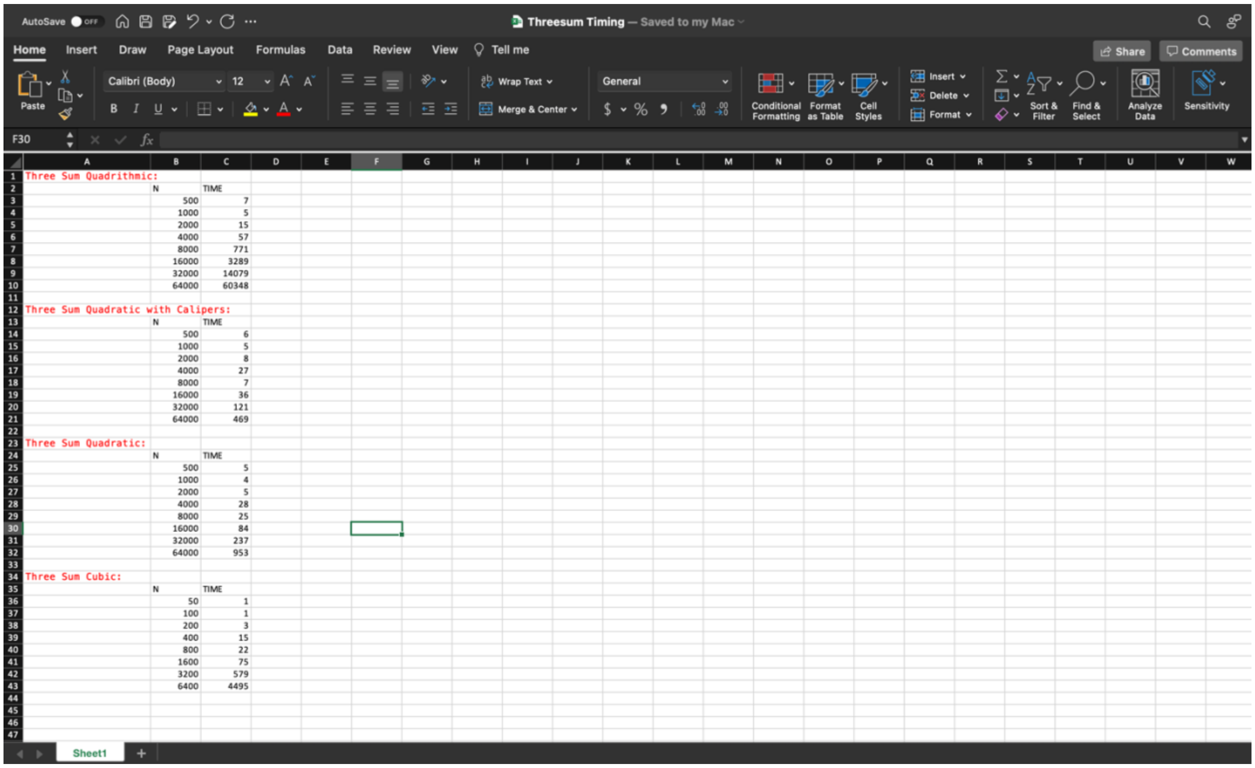

Package Explorer
JUnit
ThreeSumCubi...
ThreeSumQuad...
ThreeSumTest...
ThreeSumBenc...
Outline
edu.neu.coe.info6205.threesum
ThreeSumTest
testGetTriplesJ0() : void
testGetTriplesJ1() : void
testGetTriplesJ2() : void
testGetTriples0() : void
testGetTriples1() : void
testGetTriples2() : void
testGetTriples3() : void
testGetTriples4() : void
testGetTriplesC0() : void
testGetTriplesC1() : void
testGetTriplesC2() : void
testGetTriplesC3() : void
testGetTriplesC4() : void
Failure Trace
Problems
Javadoc
Declaration
Console
Terminal
Supplier<int[]> intsSupplier - edu.neu.coe.info6205.threesum.ThreeSumTest.testGetTriplesC2()

```
Finished after 1.035 seconds

Runs: 11/11          Errors: 0          Failures: 0

edu.neu.coe.info6205.threesum.ThreeSumTest [Runner: JUnit 4] (0.912 s)
    testGetTriples0 (0.001 s)
    testGetTriples1 (0.000 s)
    testGetTriples2 (0.000 s)
    testGetTriplesC0 (0.000 s)
    testGetTriplesC1 (0.000 s)
    testGetTriplesC2 (0.000 s)
    testGetTriplesC3 (0.184 s)
    testGetTriplesC4 (0.725 s)
    testGetTriplesJ0 (0.001 s)
    testGetTriplesJ1 (0.000 s)
    testGetTriplesJ2 (0.001 s)
```

```java
        int[] ints = new int[]{30, -40, -20, -10, 40, 0, 10, 5};
        Arrays.sort(ints);
        System.out.println("ints: " + Arrays.toString(ints));
        ThreeSum target = new ThreeSumQuadratic(ints);
        Triple[] triples = target.getTriples();
        System.out.println("triples: " + Arrays.toString(triples));
        assertEquals(4, triples.length);
        assertEquals(4, new ThreeSumCubic(ints).getTriples().length);
    }

    @Test
    public void testGetTriplesC1() {
        Supplier<int[]> intsSupplier = new Source(20, 20, 1L).intsSupplier(10);
        int[] ints = intsSupplier.get();
        ThreeSum target = new ThreeSumQuadraticWithCalipers(ints);
        Triple[] triples = target.getTriples();
        assertEquals(4, triples.length);
        System.out.println(Arrays.toString(triples));
        Triple[] triples2 = new ThreeSumCubic(ints).getTriples();
        System.out.println(Arrays.toString(triples2));
        assertEquals(4, triples2.length);
    }

    @Test
    public void testGetTriplesC2() {
        Supplier<int[]> intsSupplier = new Source(10, 15, 3L).intsSupplier(10);
        int[] ints = intsSupplier.get();
        ThreeSum targ
        System.out.pr
        Triple[] trip
        System.out.pr
        assertEquals(
        assertEquals(1, new ThreeSumCubic(ints).getTriples().length);
    }

    @Test
    public void testGetTriplesC3() {
        Supplier<int[]> intsSupplier = new Source(1000, 1000).intsSupplier(10);
        int[] ints = intsSupplier.get();
        ThreeSum target = new ThreeSumQuadraticWithCalipers(ints);
        Triple[] triplesQuadratic = target.getTriples();
        Triple[] triplesCubic = new ThreeSumCubic(ints).getTriples();
        assertEquals(triplesCubic.length, triplesQuadratic.length);
```

```
<terminated> ThreeSumTest [JUnit] /Applications/Eclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.4.v20220903-1038/jre
triples: [Triple{x=0, y=3, z=7}, Triple{x=0, y=5, z=6}, Triple{x=1, y=2, z=6}, Triple{x=2, y=3, z=5}]
[Triple{x=5, y=12, z=17}, Triple{x=5, y=13, z=16}, Triple{x=6, y=12, z=16}, Triple{x=10, y=12, z=13}]
[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=3, y=5, z=7}]
ints: [-40, -20, -10, 0, 5, 10, 30, 40]
triples: [Triple{x=0, y=3, z=7}, Triple{x=0, y=5, z=6}, Triple{x=1, y=2, z=6}, Triple{x=2, y=3, z=5}]
[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}]
```

2. a spreadsheet showing your timing observations--using the doubling method for at least five values of N--for each of the algorithms (include cubic); Timing should be performed either with an actual stopwatch (e.g. your iPhone) or using the Stopwatch class in the repository.



	A	B	C
1	Three Sum Quadrithmic:		
2		N	TIME
3		500	7
4		1000	5
5		2000	15
6		4000	57
7		8000	771
8		16000	3289
9		32000	14079
10		64000	60348
11			
12	Three Sum Quadratic with Calipers:		
13		N	TIME
14		500	6
15		1000	5
16		2000	8
17		4000	27
18		8000	7
19		16000	36
20		32000	121
21		64000	469
22			
23	Three Sum Quadratic:		
24		N	TIME
25		500	5
26		1000	4
27		2000	5
28		4000	28
29		8000	25
30		16000	84
31		32000	237
32		64000	953
33			
34	Three Sum Cubic:		
35		N	TIME
36		50	1
37		100	1
38		200	3
39		400	15
40		800	22
41		1600	75
42		3200	579
43		6400	4495

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

The quadratic method for solving the 3 sum problem in Java involves using nested loops to check every combination of three elements in the input array. The outer loop iterates over each element in the array, and the inner loop iterates over all the remaining elements in the array, starting at the next index after the outer loop's current index. For each combination of three elements, the sum is calculated and checked against zero. If the sum is zero, then the combination of elements is added to the result.

The time complexity of this method is O(n^2), since for each element in the array, we must iterate over all the remaining elements in the array. This makes it suitable for small input arrays, but for large arrays, it can be quite slow and inefficient. However, it is a simple and straightforward method for solving the 3 sum problem and is good for understanding the basic approach.