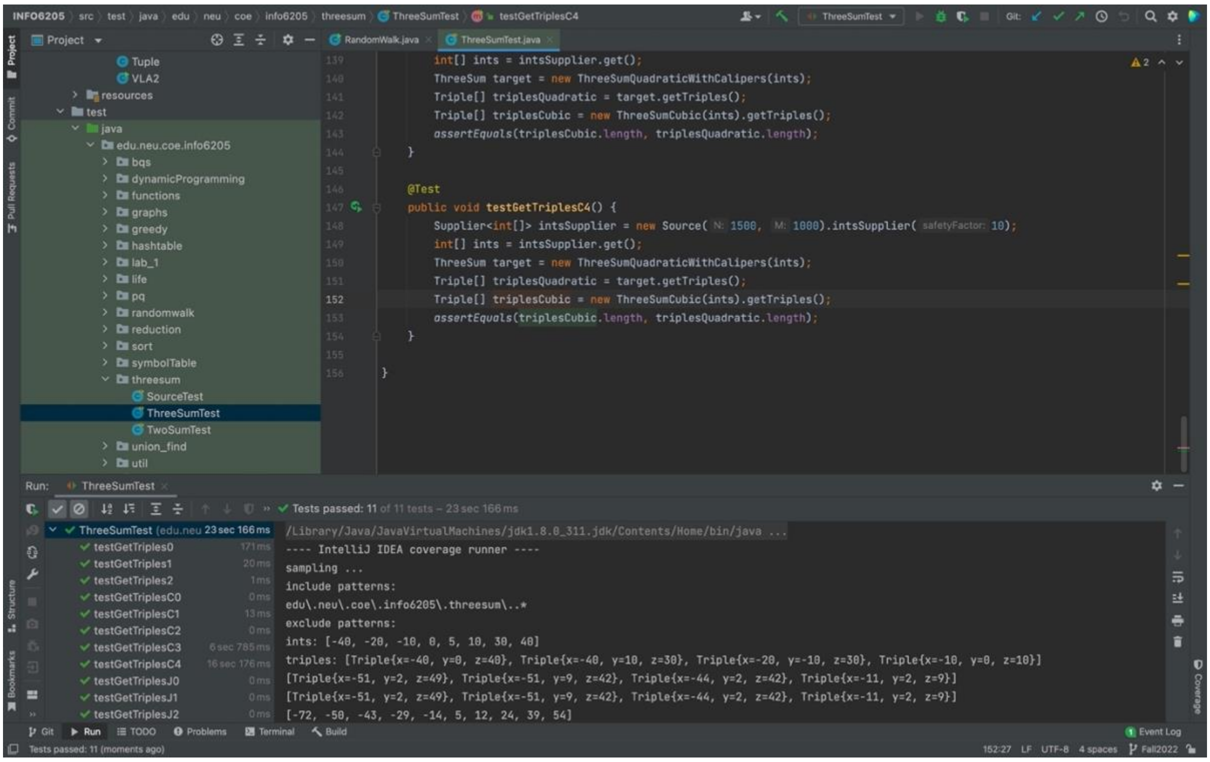
Assignment 2- 3 Sum

A: Evidence



```java
        int[] ints = intsSupplier.get();
        ThreeSum target = new ThreeSumQuadraticWithCalipers(ints);
        Triple[] triplesQuadratic = target.getTriples();
        Triple[] triplesCubic = new ThreeSumCubic(ints).getTriples();
        assertEquals(triplesCubic.length, triplesQuadratic.length);
    }

    @Test
    public void testGetTriplesC4() {
        Supplier<int[]> intsSupplier = new Source( N: 1500,  M: 1000).intsSupplier( safetyFactor: 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);
    }

}
```

Run: ThreeSumTest

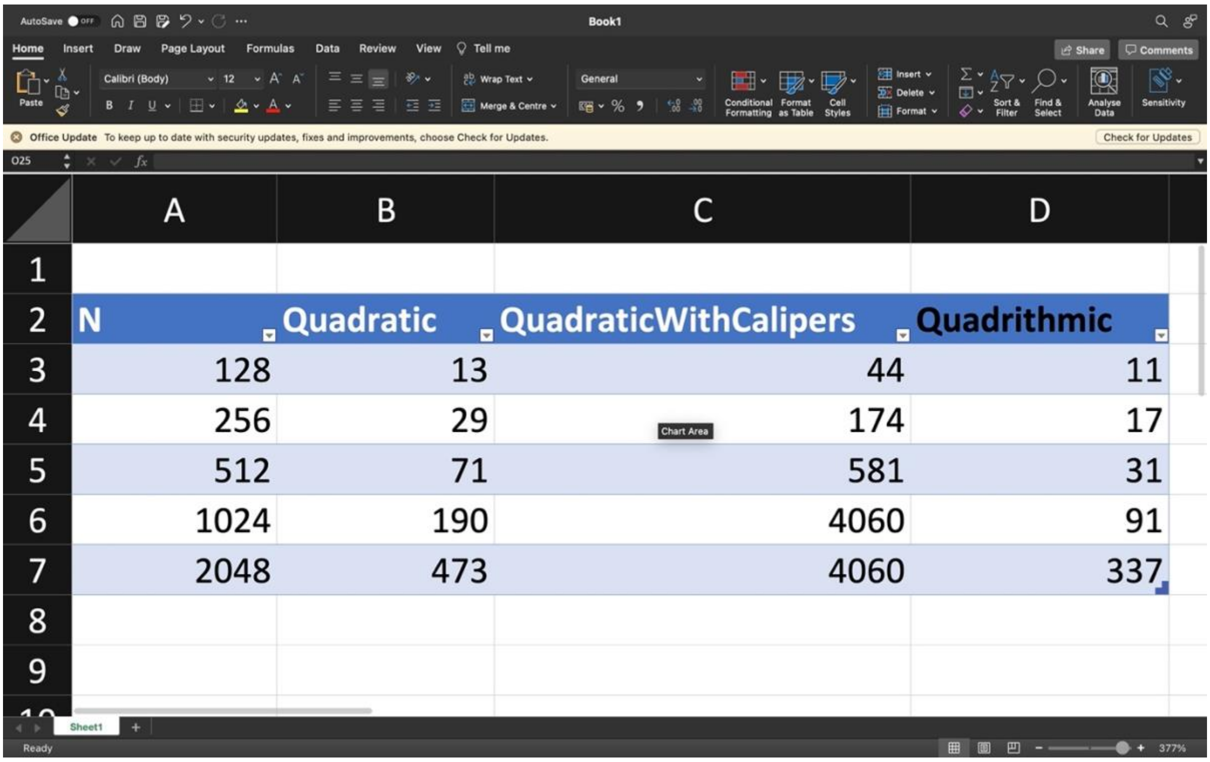
✓ Tests passed: 11 of 11 tests – 23 sec 166 ms

```
✓ ThreeSumTest (edu.neu 23 sec 166 ms
    ✓ testGetTriples0           171 ms
    ✓ testGetTriples1            20 ms
    ✓ testGetTriples2             1 ms
    ✓ testGetTriplesC0            0 ms
    ✓ testGetTriplesC1           13 ms
    ✓ testGetTriplesC2            0 ms
    ✓ testGetTriplesC3   6 sec 785 ms
    ✓ testGetTriplesC4  16 sec 176 ms
    ✓ testGetTriplesJ0            0 ms
    ✓ testGetTriplesJ1            0 ms
    ✓ testGetTriplesJ2            0 ms
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_311.jdk/Contents/Home/bin/java ...
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
edu\.neu\.coe\.info6205\.threesum\..*
exclude patterns:
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]
```

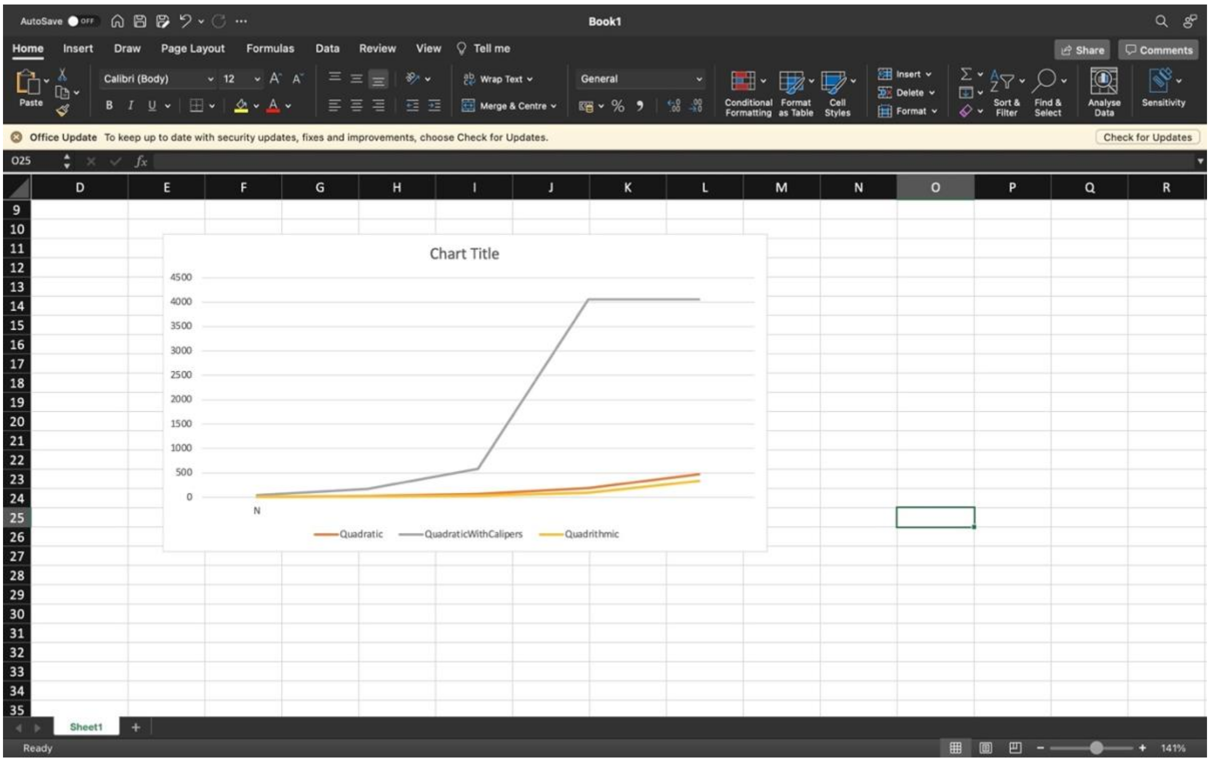
Tests passed: 11 (moments ago)

B: Observation



Office Update To keep up to date with security updates, fixes and improvements, choose Check for Updates.

N	Quadratic	QuadraticWithCalipers	Quadrithmic
128	13	44	11
256	29	174	17
512	71	581	31
1024	190	4060	91
2048	473	4060	337



Office Update To keep up to date with security updates, fixes and improvements, choose Check for Updates.

C: Explanation

Quadratic with calipers:

Once we sort the array, for each element, we try to find two elements to it’s right(greater than that element because the array is sorted) that are equal to target(-1*current element).

This is done through the two pointer technique, where we keep one pointer immediately to it’s right and one pointer at the end of the array.

If the sum of the elements pointed to by the pointers is greater than target, we move the right pointer to the left. If the sum of the elements pointed to by the pointers is lesser than the target, we move the left pointer to the right.

If the sum is equal to target, we add the triplet to the list.

Quadratic:

We store all elements in the array to a HashSet. Then, we iterate through all possible pairs and for each pair, compute the sum of the pair and check whether target is present in the HashSet.