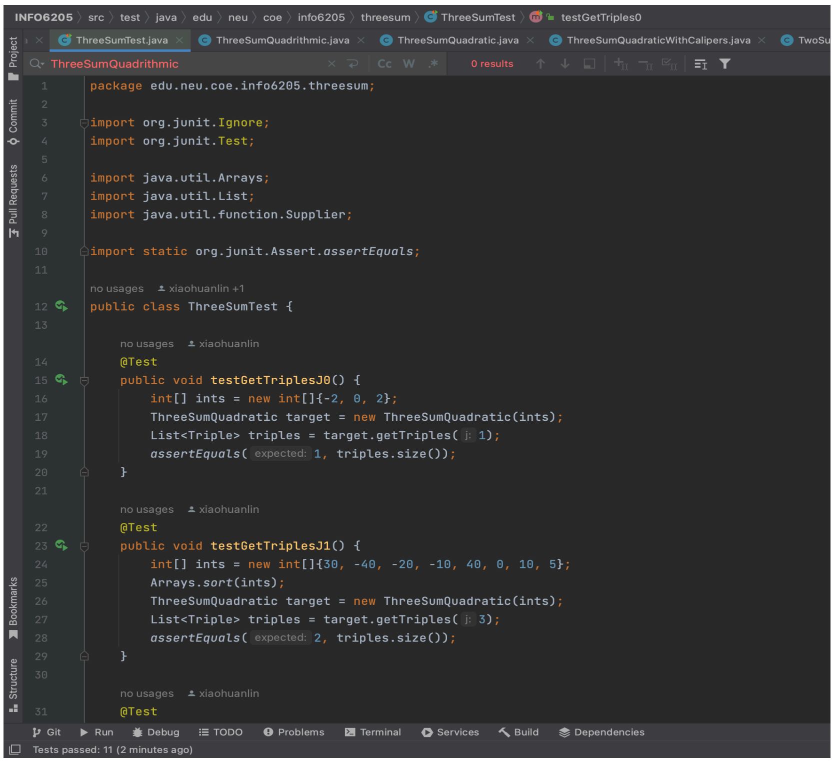
Assignment 2 (3-SUM)

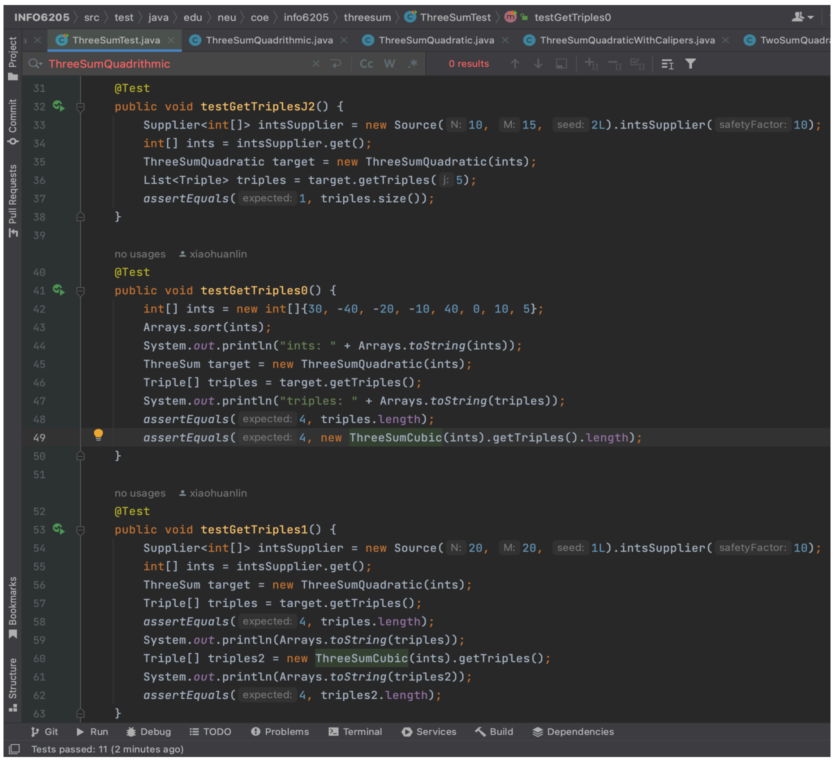
1. Evidence

Screenshots of all running unit tests



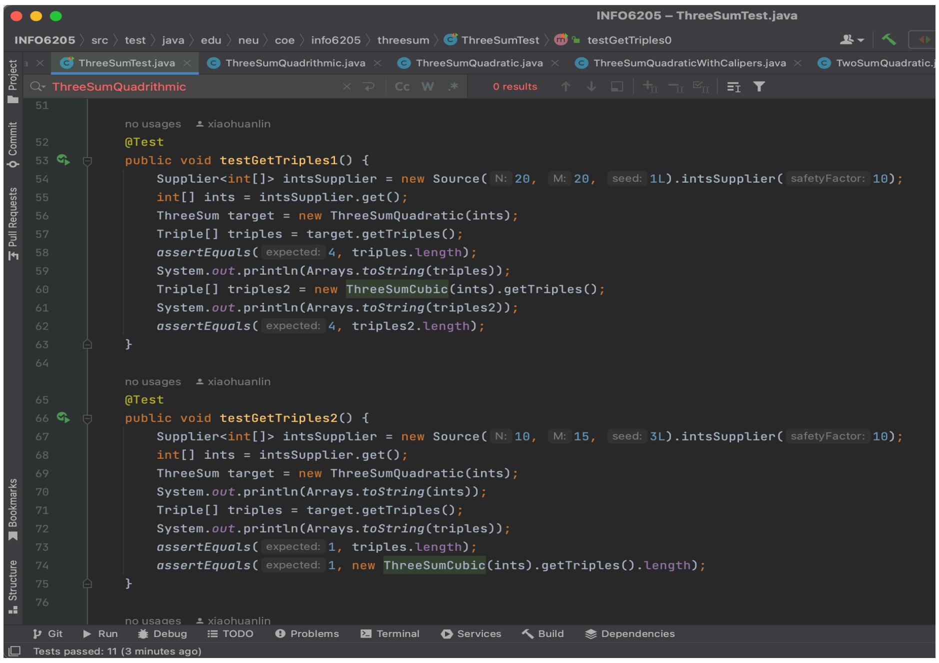
```
package edu.neu.coe.info6205.threesum;

import org.junit.Ignore;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
import java.util.function.Supplier;

import static org.junit.Assert.assertEquals;

no usages   xiaohuanlin +1
public class ThreeSumTest {

    no usages   xiaohuanlin
    @Test
    public void testGetTriplesJ0() {
        int[] ints = new int[]{-2, 0, 2};
        ThreeSumQuadratic target = new ThreeSumQuadratic(ints);
        List<Triple> triples = target.getTriples( j: 1);
        assertEquals( expected: 1, triples.size());
    }

    no usages   xiaohuanlin
    @Test
    public void testGetTriplesJ1() {
        int[] ints = new int[]{30, -40, -20, -10, 40, 0, 10, 5};
        Arrays.sort(ints);
        ThreeSumQuadratic target = new ThreeSumQuadratic(ints);
        List<Triple> triples = target.getTriples( j: 3);
        assertEquals( expected: 2, triples.size());
    }

    no usages   xiaohuanlin
    @Test
```

Tests passed: 11 (2 minutes ago)



```
    @Test
    public void testGetTriplesJ2() {
        Supplier<int[]> intsSupplier = new Source( N: 10,  M: 15,  seed: 2L).intsSupplier( safetyFactor: 10);
        int[] ints = intsSupplier.get();
        ThreeSumQuadratic target = new ThreeSumQuadratic(ints);
        List<Triple> triples = target.getTriples( j: 5);
        assertEquals( expected: 1, triples.size());
    }

    no usages   xiaohuanlin
    @Test
    public void testGetTriples0() {
        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( expected: 4, triples.length);
        assertEquals( expected: 4, new ThreeSumCubic(ints).getTriples().length);
    }

    no usages   xiaohuanlin
    @Test
    public void testGetTriples1() {
        Supplier<int[]> intsSupplier = new Source( N: 20,  M: 20,  seed: 1L).intsSupplier( safetyFactor: 10);
        int[] ints = intsSupplier.get();
        ThreeSum target = new ThreeSumQuadratic(ints);
        Triple[] triples = target.getTriples();
        assertEquals( expected: 4, triples.length);
        System.out.println(Arrays.toString(triples));
        Triple[] triples2 = new ThreeSumCubic(ints).getTriples();
        System.out.println(Arrays.toString(triples2));
        assertEquals( expected: 4, triples2.length);
    }
```

Tests passed: 11 (2 minutes ago)



INFO6205 – ThreeSumTest.java

INFO6205 〉 src 〉 test 〉 java 〉 edu 〉 neu 〉 coe 〉 info6205 〉 threesum 〉 ThreeSumTest 〉 testGetTriples0

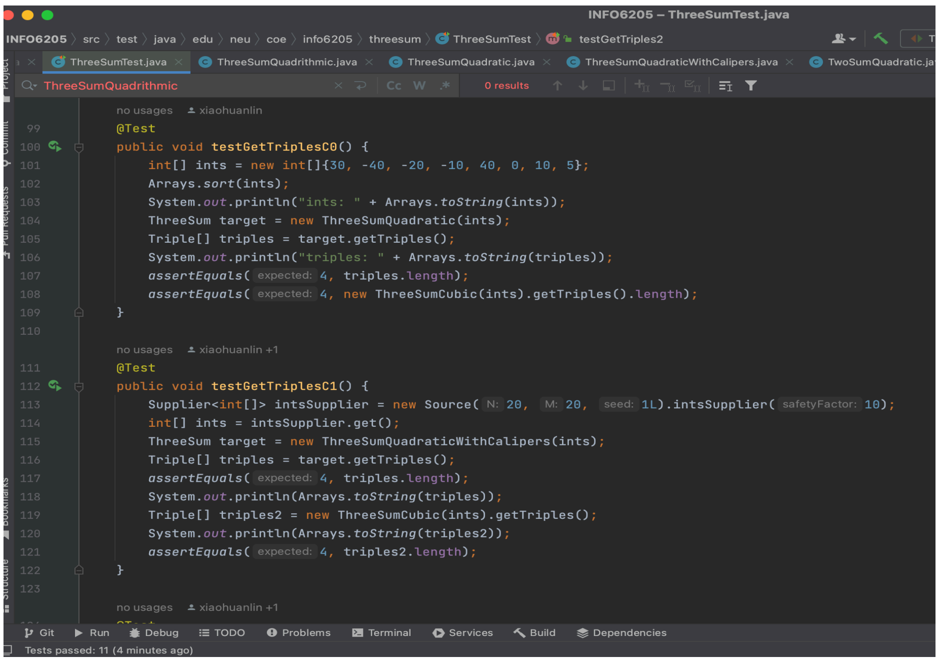
ThreeSumTest.java × | ThreeSumQuadrithmic.java × | ThreeSumQuadratic.java × | ThreeSumQuadraticWithCalipers.java × | TwoSumQuadratic.

ThreeSumQuadrithmic — 0 results

```
    no usages  xiaohuanlin
    @Test
    public void testGetTriples1() {
        Supplier<int[]> intsSupplier = new Source( N: 20,  M: 20,  seed: 1L).intsSupplier( safetyFactor: 10);
        int[] ints = intsSupplier.get();
        ThreeSum target = new ThreeSumQuadratic(ints);
        Triple[] triples = target.getTriples();
        assertEquals( expected: 4, triples.length);
        System.out.println(Arrays.toString(triples));
        Triple[] triples2 = new ThreeSumCubic(ints).getTriples();
        System.out.println(Arrays.toString(triples2));
        assertEquals( expected: 4, triples2.length);
    }

    no usages  xiaohuanlin
    @Test
    public void testGetTriples2() {
        Supplier<int[]> intsSupplier = new Source( N: 10,  M: 15,  seed: 3L).intsSupplier( safetyFactor: 10);
        int[] ints = intsSupplier.get();
        ThreeSum target = new ThreeSumQuadratic(ints);
        System.out.println(Arrays.toString(ints));
        Triple[] triples = target.getTriples();
        System.out.println(Arrays.toString(triples));
        assertEquals( expected: 1, triples.length);
        assertEquals( expected: 1, new ThreeSumCubic(ints).getTriples().length);
    }

    no usages  xiaohuanlin
```

Git | Run | Debug | TODO | Problems | Terminal | Services | Build | Dependencies

Tests passed: 11 (3 minutes ago)



INFO6205 – ThreeSumTest.java

INFO6205 〉 src 〉 test 〉 java 〉 edu 〉 neu 〉 coe 〉 info6205 〉 threesum 〉 ThreeSumTest 〉 testGetTriples2

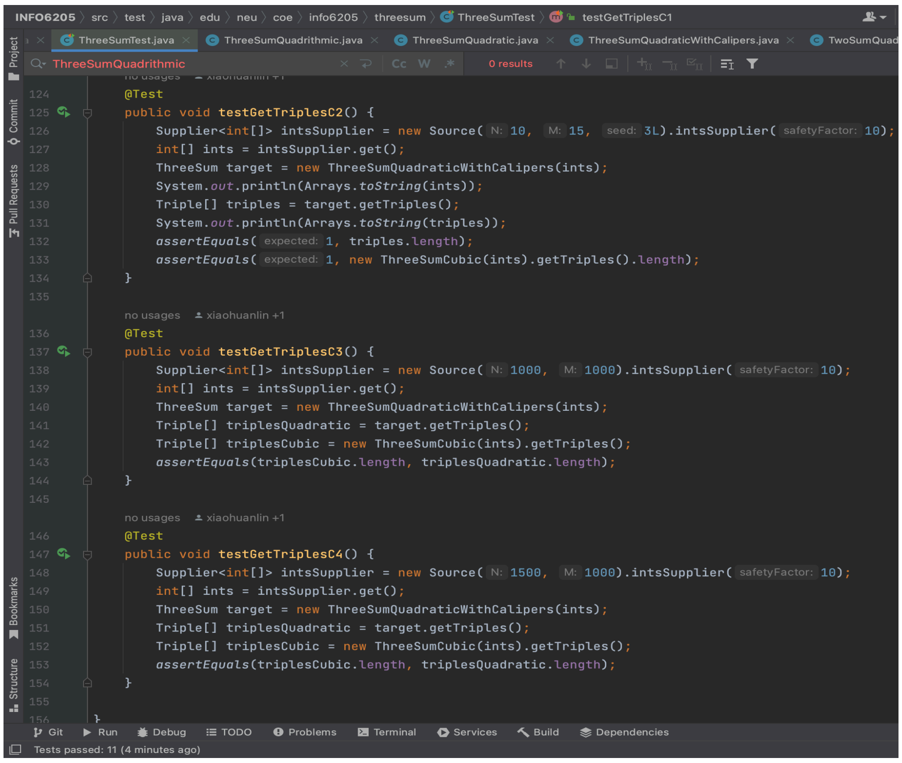
ThreeSumTest.java × | ThreeSumQuadrithmic.java × | ThreeSumQuadratic.java × | ThreeSumQuadraticWithCalipers.java × | TwoSumQuadratic.ja

ThreeSumQuadrithmic — 0 results

```
    no usages  xiaohuanlin
    @Test
    public void testGetTriplesC0() {
        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( expected: 4, triples.length);
        assertEquals( expected: 4, new ThreeSumCubic(ints).getTriples().length);
    }

    no usages  xiaohuanlin +1
    @Test
    public void testGetTriplesC1() {
        Supplier<int[]> intsSupplier = new Source( N: 20,  M: 20,  seed: 1L).intsSupplier( safetyFactor: 10);
        int[] ints = intsSupplier.get();
        ThreeSum target = new ThreeSumQuadraticWithCalipers(ints);
        Triple[] triples = target.getTriples();
        assertEquals( expected: 4, triples.length);
        System.out.println(Arrays.toString(triples));
        Triple[] triples2 = new ThreeSumCubic(ints).getTriples();
        System.out.println(Arrays.toString(triples2));
        assertEquals( expected: 4, triples2.length);
    }

    no usages  xiaohuanlin +1
```

Git | Run | Debug | TODO | Problems | Terminal | Services | Build | Dependencies

Tests passed: 11 (4 minutes ago)

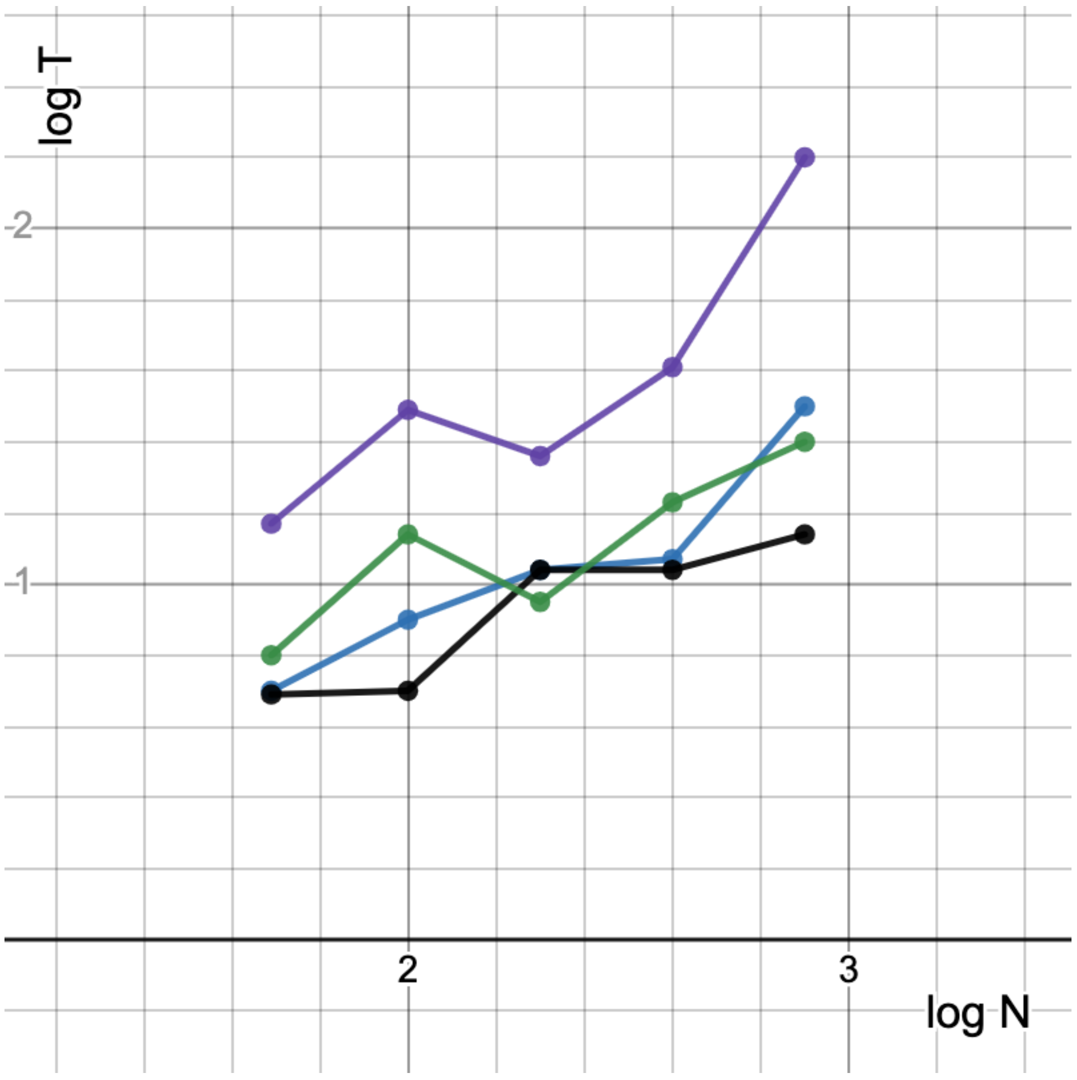


```
    no usages   xiaohuanlin +1
    @Test
    public void testGetTriplesC2() {
        Supplier<int[]> intsSupplier = new Source( N: 10,  M: 15,  seed: 3L).intsSupplier( safetyFactor: 10);
        int[] ints = intsSupplier.get();
        ThreeSum target = new ThreeSumQuadraticWithCalipers(ints);
        System.out.println(Arrays.toString(ints));
        Triple[] triples = target.getTriples();
        System.out.println(Arrays.toString(triples));
        assertEquals( expected: 1, triples.length);
        assertEquals( expected: 1, new ThreeSumCubic(ints).getTriples().length);
    }

    no usages   xiaohuanlin +1
    @Test
    public void testGetTriplesC3() {
        Supplier<int[]> intsSupplier = new Source( N: 1000,  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);
    }

    no usages   xiaohuanlin +1
    @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);
    }

}
```

Tests passed: 11 (4 minutes ago)

2. Timing Observations

Algorithm	N	Time (ms)	log N	log T
ThreeSumCubic	50	15	1.69	1.17
ThreeSumCubic	100	31	2	1.49
ThreeSumCubic	200	23	2.3	1.36
ThreeSumCubic	400	41	2.6	1.61
ThreeSumCubic	800	164	2.9	2.2
ThreeSumQuadratic	50	6	1.69	0.7
ThreeSumQuadratic	100	9	2	0.9
ThreeSumQuadratic	200	11	2.3	1.04
ThreeSumQuadratic	400	12	2.6	1.07
ThreeSumQuadratic	800	34	2.9	1.5
ThreeSumQuadraticWithCalipers	50	5	1.69	0.69
ThreeSumQuadraticWithCalipers	100	6	2	0.7
ThreeSumQuadraticWithCalipers	200	11	2.3	1.04
ThreeSumQuadraticWithCalipers	400	11	2.6	1.04
ThreeSumQuadraticWithCalipers	800	14	2.9	1.14
ThreeSumQuadrithmic	50	7	1.69	0.8
ThreeSumQuadrithmic	100	14	2	1.14
ThreeSumQuadrithmic	200	9	2.3	0.95
ThreeSumQuadrithmic	400	17	2.6	1.23
ThreeSumQuadrithmic	800	31	2.9	1.4



Graph based on experimental values

🟦	ThreeSumQuadratic
🟪	ThreeSumCubic
🟩	ThreeSumQuadrithmic
⬛	ThreeSumQuadraticWithCalipers

3. **Quadratic Method**
   Sorting the array takes O (n log n), so overall complexity is O (n log n + n^2) equivalent to O(n^2). We have used loops couple of times first we are iterating each element of array which makes complexity n times & in second loop, we are checking if index of left value is greater than 0 & index of right value is less than the length of array which again makes complexity n times. Hence, total time complexity O(n*n) = O(n^2). Quadratic method is time efficient and it has better performance.