

# INFO6205 Assignment 2 (3-SUM)

NAME: Bohan Feng

NUID: 001564249

Repository: <https://github.com/fengb3/INFO6205>

## Assignment02 Three Sum

**a) Screenshots of unit testing code and test pass**

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);
}
```

xiaohuanlin

@Ignore // Slow

```
public void testGetTriples3() {
    Supplier<int[]> intsSupplier = new Source( N: 1000, M: 1000).intsSupplier( safetyFactor: 10);
    int[] ints = intsSupplier.get();
    ThreeSum target = new ThreeSumQuadratic(ints);
    Triple[] triplesQuadratic = target.getTriples();
    Triple[] triplesCubic = new ThreeSumCubic(ints).getTriples();
    int expected1 = triplesCubic.length;
    assertEquals(expected1, triplesQuadratic.length);
}
```

xiaohuanlin

@Ignore // Slow

```
public void testGetTriples4() {
    Supplier<int[]> intsSupplier = new Source( N: 1500, M: 1000).intsSupplier( safetyFactor: 10);
    int[] ints = intsSupplier.get();
    ThreeSum target = new ThreeSumQuadratic(ints);
    Triple[] triplesQuadratic = target.getTriples();
    Triple[] triplesCubic = new ThreeSumCubic(ints).getTriples();
    int expected1 = triplesCubic.length;
    assertEquals(expected1, triplesQuadratic.length);
}
```

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);
}
```

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);
}
```

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);
}
```

```
28     Threesum target = new ThreesumquadraticWithCatalpers(ints);
29     System.out.println(Arrays.toString(ints));
30     Triple[] triples = target.getTriples();
```

Run: ThreeSumTest x

✓ Tests passed: 11 of 11 tests - 788 ms

Test Name	Duration	Output
✓ ThreeSumTest (edu.neu.coe.info6205.threesum)	788 ms	C:\Users\冯博藻\.jdk\openjdk-18.0.2.1\bin\java.exe ...
✓ testGetTriples0	5 ms	ints: [-40, -20, -10, 0, 5, 10, 30, 40]
✓ testGetTriples1	2 ms	triples: [Triple{x=-40, y=0, z=40}, Triple{x=-40, y=10, z=30}, Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2, z=49}, Triple{x=-29, y=5, z=24}]
✓ testGetTriples2	1 ms	
✓ testGetTriplesC0	0 ms	
✓ testGetTriplesC1	2 ms	
✓ testGetTriplesC2	1 ms	
✓ testGetTriplesC3	190 ms	ints: [-40, -20, -10, 0, 5, 10, 30, 40]
✓ testGetTriplesC4	587 ms	triples: [Triple{x=-40, y=0, z=40}, Triple{x=-40, y=10, z=30}, Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2, z=49}, Triple{x=-29, y=5, z=24}]
✓ testGetTriplesJ0	0 ms	
✓ testGetTriplesJ1	0 ms	
✓ testGetTriplesJ2	0 ms	

```

95     int expected1 = triplesCubic.length;
96     assertEquals(expected1, triplesQuadratic.length);
97 }
98
99 xiaohuanlin
100 @Test
101 public void testGetTriplesC0() {
102     int[] ints = new int[]{30, -40, -20, -10, 40, 0, 10, 5};
103     Arrays.sort(ints);
104     System.out.println("ints: " + Arrays.toString(ints));
105     ThreeSum target = new ThreeSumQuadratic(ints);
106     Triple[] triples = target.getTriples();
107     System.out.println("triples: " + Arrays.toString(triples));
108     assertEquals(expected: 4, triples.length);
109     assertEquals(expected: 4, new ThreeSumCubic(ints).getTriples().length);
110 }

```

```

111 xiaohuanlin +1
112 @Test
113 public void testGetTriplesC1() {
114     Supplier<int[]> intsSupplier = new Source(N: 20, M: 20, seed: 1L).intsSupplier(safetyFactor: 10);
115     int[] ints = intsSupplier.get();
116     ThreeSum target = new ThreeSumQuadraticWithCalipers(ints);
117     Triple[] triples = target.getTriples();
118     assertEquals(expected: 4, triples.length);
119     System.out.println(Arrays.toString(triples));
120     Triple[] triples2 = new ThreeSumCubic(ints).getTriples();
121     System.out.println(Arrays.toString(triples2));
122     assertEquals(expected: 4, triples2.length);
123 }

```

```

124 xiaohuanlin +1
125 @Test
126 public void testGetTriplesC2() {
127     Supplier<int[]> intsSupplier = new Source(N: 10, M: 15, seed: 3L).intsSupplier(safetyFactor: 10);
128     int[] ints = intsSupplier.get();
129     ThreeSum target = new ThreeSumQuadraticWithCalipers(ints);
130     System.out.println(Arrays.toString(ints));
131     Triple[] triples = target.getTriples();
132     System.out.println(Arrays.toString(triples));
133     assertEquals(expected: 1, triples.length);
134     assertEquals(expected: 1, new ThreeSumCubic(ints).getTriples().length);
135 }

```

```

136 xiaohuanlin +1
137 @Test
138 public void testGetTriplesC3() {
139     Supplier<int[]> intsSupplier = new Source(N: 1000, M: 1000).intsSupplier(safetyFactor: 10);
140     int[] ints = intsSupplier.get();
141     ThreeSum target = new ThreeSumQuadraticWithCalipers(ints);
142     Triple[] triplesQuadratic = target.getTriples();
143     Triple[] triplesCubic = new ThreeSumCubic(ints).getTriples();
144     assertEquals(triplesCubic.length, triplesQuadratic.length);
145 }

```

```

146 xiaohuanlin +1
147 @Test
148 public void testGetTriplesC4() {
149     Supplier<int[]> intsSupplier = new Source(N: 1500, M: 1000).intsSupplier(safetyFactor: 10);
150     int[] ints = intsSupplier.get();
151     ThreeSum target = new ThreeSumQuadraticWithCalipers(ints);
152     Triple[] triplesQuadratic = target.getTriples();
153     Triple[] triplesCubic = new ThreeSumCubic(ints).getTriples();
154     assertEquals(triplesCubic.length, triplesQuadratic.length);
155 }

```

```

156 }

```



```

1 package edu.neu.coe.info6205.threesum;
2
3 import ...
4
11
12 public class ThreeSumTest {
13
14     @Test
15     public void testGetTriplesJ0() {
16         int[] ints = new int[]{-2, 0, 2};
17         ThreeSumQuadratic target = new ThreeSumQuadratic(ints);
18         List<Triple> triples = target.getTriples(j: 1);
19         assertEquals( expected: 1, triples.size());
20     }
21
22     @Test
23     public void testGetTriplesJ1() {
24         int[] ints = new int[]{30, -40, -20, -10, 40, 0, 10, 5};
25         Arrays.sort(ints);
26         ThreeSumQuadratic target = new ThreeSumQuadratic(ints);
27         List<Triple> triples = target.getTriples(j: 3);
28         assertEquals( expected: 2, triples.size());
29     }
30
31     @Test
32     public void testGetTriplesJ2() {
33         Supplier<int[]> intsSupplier = new Source( N: 10, M: 15, seed: 2L).intsSupplier( safetyFactor: 10);
34         int[] ints = intsSupplier.get();
35         ThreeSumQuadratic target = new ThreeSumQuadratic(ints);
36         List<Triple> triples = target.getTriples(j: 5);
37         assertEquals( expected: 1, triples.size());
38     }
39
40     @Test
41     public void testGetTriples0() {
42         int[] ints = new int[]{30, -40, -20, -10, 40, 0, 10, 5};
43         Arrays.sort(ints);
44         System.out.println("ints: " + Arrays.toString(ints));
45         ThreeSum target = new ThreeSumQuadratic(ints);
46         Triple[] triples = target.getTriples();
47         System.out.println("triples: " + Arrays.toString(triples));
48         assertEquals( expected: 4, triples.length);
49         assertEquals( expected: 4, new ThreeSumCubic(ints).getTriples().length);
50     }
51
52     @Test
53     public void testGetTriples1() {
54         Supplier<int[]> intsSupplier = new Source( N: 20, M: 20, seed: 1L).intsSupplier( safetyFactor: 10);
55         int[] ints = intsSupplier.get();
56         ThreeSum target = new ThreeSumQuadratic(ints);
57         Triple[] triples = target.getTriples();
58         assertEquals( expected: 4, triples.length);
59         System.out.println(Arrays.toString(triples));
60         Triple[] triples2 = new ThreeSumCubic(ints).getTriples();
61         System.out.println(Arrays.toString(triples2));
62         assertEquals( expected: 4, triples2.length);
63     }
64
65     @Test
66     public void testGetTriples2() {
67         Supplier<int[]> intsSupplier = new Source( N: 10, M: 15, seed: 3L).intsSupplier( safetyFactor: 10);
68         int[] ints = intsSupplier.get();
69         ThreeSum target = new ThreeSumQuadratic(ints);
70         System.out.println(Arrays.toString(ints));

```

```

70     System.out.println(Arrays.toString(ints));
71     Triple[] triples = target.getTriples();
72     System.out.println(Arrays.toString(triples));

```

Run: ThreeSumTest x

Tests passed: 11 of 11 tests - 788 ms

Test Name	Duration
ThreeSumTest (edu.neu.coe.info6205.threesum)	788 ms
testGetTriples0	5 ms
testGetTriples1	2 ms
testGetTriples2	1 ms
testGetTriplesC0	0 ms
testGetTriplesC1	2 ms
testGetTriplesC2	1 ms
testGetTriplesC3	190 ms
testGetTriplesC4	587 ms
testGetTriplesJ0	0 ms
testGetTriplesJ1	0 ms
testGetTriplesJ2	0 ms

```

C:\Users\冯博藻\.jdk\openjdk-18.0.2.1\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{
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2
[-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{
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2
[-72, -50, -43, -29, -14, 5, 12, 24, 39, 54]
[Triple{x=-29, y=5, z=24}]

```

b) Spread sheet showing timing observations

N	ThreeSumQuadratic	ThreeSumQuadrithmic	ThreeSumCubic
250	1.95	2.17	8.57
500	2.01	3.7	21.94
1000	4.87	14.18	135.8
2000	15.74	56.7	952.79
4000	57.1	241.13	7306.84

c) Explanation of why quadratic method works

code :

```

public Triple[] getTriples() {
    List<Triple> result = new ArrayList<>();
    int length = a.length;

    // check validation
    if(a == null || length < 3)
        return result.toArray(new Triple[0]);

    // sort the array
    Arrays.sort(a);

    // first loop from start to end of the array
    for(int i = 0; i < length - 2; i++)
    {
        // since the array is sorted, there is not a chance that sum is 0, if we loop to a
        if(a[i]>0)
            break;

        // skip if we meet duplicate number
        if(i > 0 && a[i] == a[i - 1])
            continue;

        // 2 pointers that loop from start and end to the center
        int left = i + 1;
        int right = length - 1;

        // second loop
        while(left < right)
        {
            int sum = a[i] + a[left] + a[right];

            // check the sum
            if(sum == 0)
            {
                // if the sum is 0, which is what we want, remember it
                result.add(new Triple(a[i], a[left], a[right]));

                // then move 2 pointer one step the the center
                left++;
                right--;

                // skip duplicate numbers
                while(left < right && a[left] == a[left - 1])
                    left++;

                while(left < right && a[right] == a[right + 1])
                    right--;
            }

            // if the sum is smaller than 0, we need to move left pointer, to make the sum b
            else if(sum < 0)

```



```

        left++;

        // similarly, move the right pointer to make the sum small to approach 0
        else
            right--;
    }
}

return result.stream().distinct().toArray(Triple[]::new);
}

```

The quadratic method is basically an optimized version of cubic method.

In cubic method, we have three layer of loops to find 3 numbers.

In quadratic method, we combine second and third layer loops into a 2 pointer check to find second and third number we want.

In first layer we loop the array from the start to end, the check **i** th number in array.

We set up 2 pointers (**Left** and **right**) at both sides of the rest of the array.

Then we check the sum of **i**, **left** and **right**

There are 3 different situation in our 2 pointer check :

- if the sum is smaller than 0, we apply `left++` .
- if the sum is greater than 0, we apply `right++` .
- if the sum is equal to 0, we found one of the answer, add it to the list.

until the **left** is equal to **right**

Then we can start the next iteration of the first layer loop to find other answers

## Time complexity

sort the array :  $O(n * \text{Log} n)$ ,

iterate over array :  $O(n)$ ,

2 pointer search :  $O(n)$ ,

total :  $O(n * \text{log} n) + O(n) * O(n) \Rightarrow O(n^2)$

## Space complexity

we are not using extra space so the space complexity is  $O(1)$