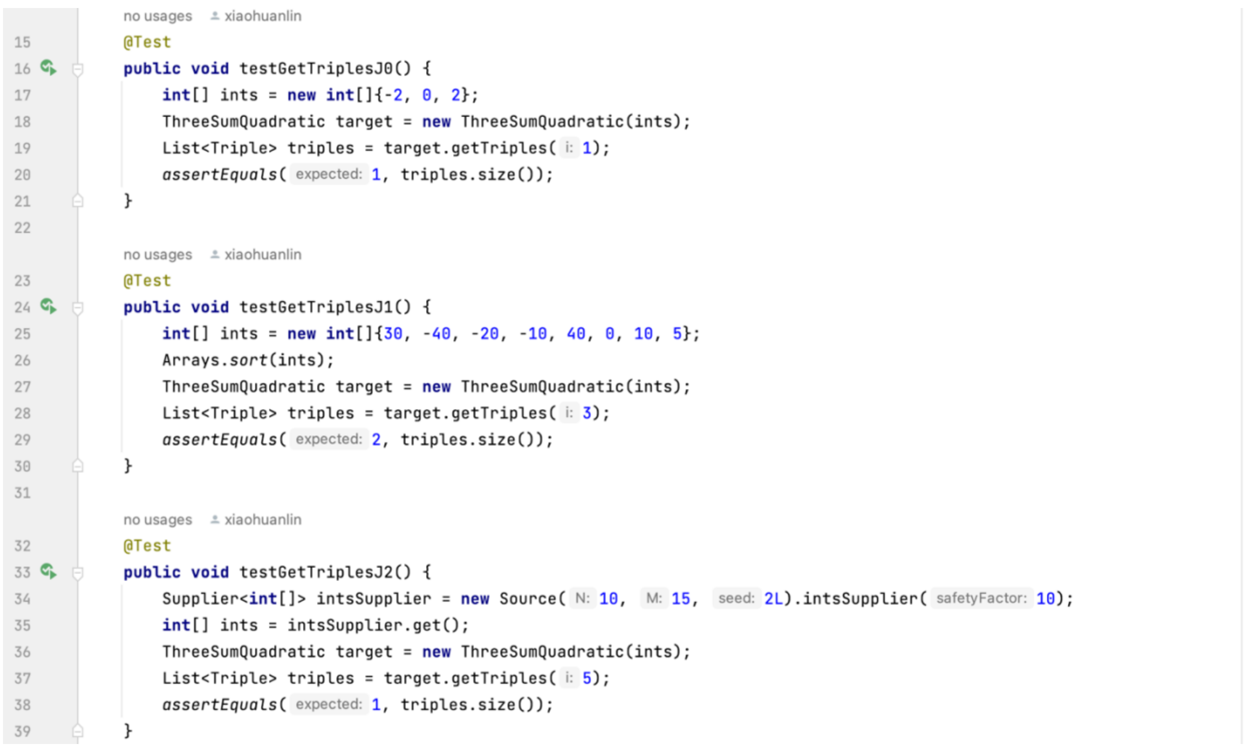
Assignment 2: Three Sum

Reason for why Quadratic Algorithm works better than Cubic Algorithm:

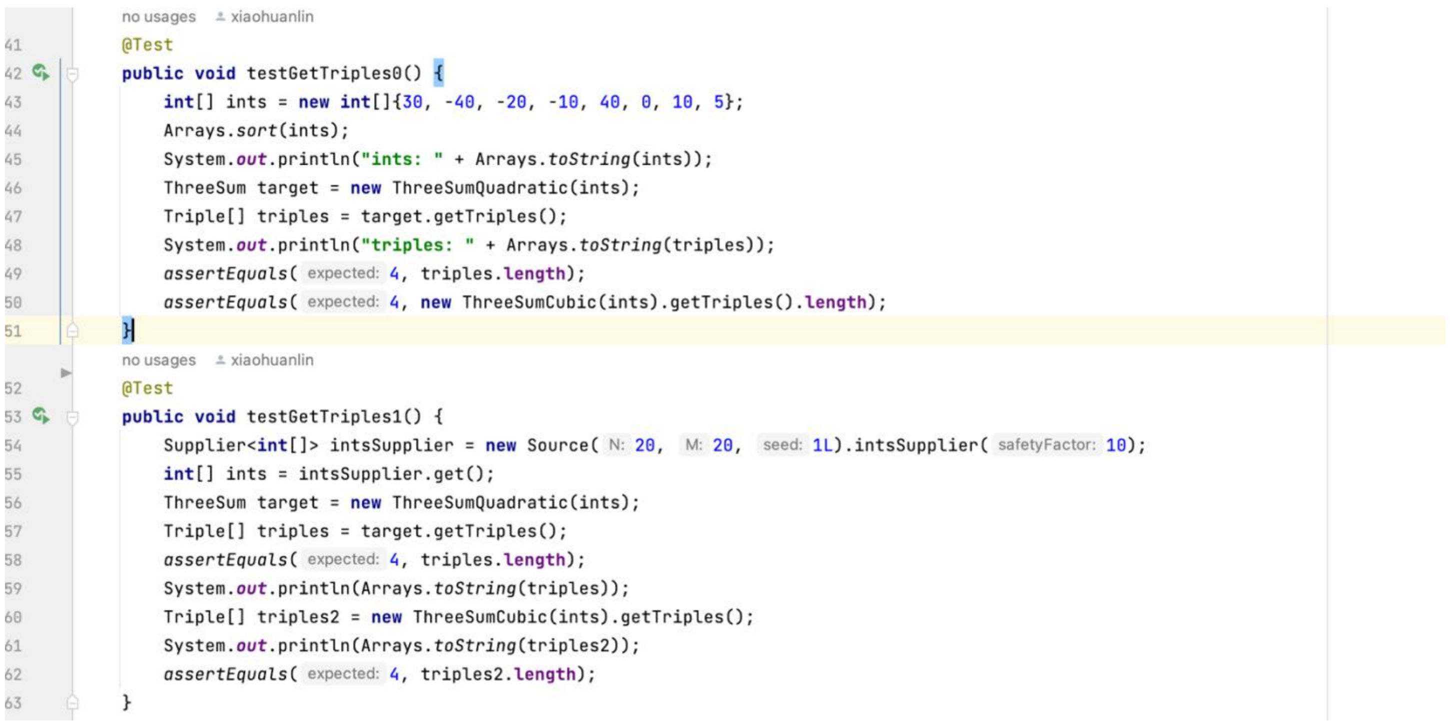
From the tables below, both the algorithms work fast when the array length is small. The difference in time is evident when the array length is more than 500. For an array length of 1000 the Quadratic method yields an output within 66ms whereas cubic algorithm produces the output in 187ms.

This is mainly because the Cubic algorithm uses three for loops for which the time complexity for the algorithm becomes $O(N^3)$. Alternatively, the Quadratic algorithm uses two loops, one to move the i value and another while loop to move the two pointers and hence the time complexity for Quadratic algorithm is $O(N^2)$. Hence when we consider big array lengths it is better to go with Quadratic Algorithm than Cubic Algorithm.

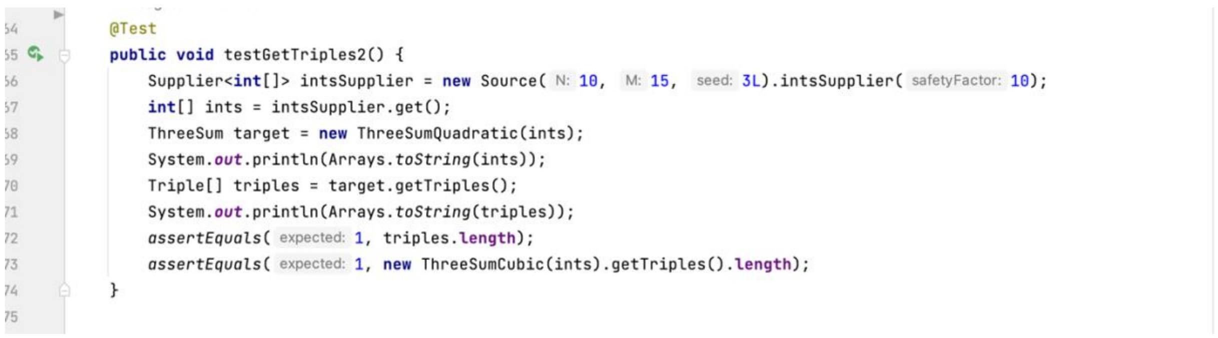
Evidence For Unit Test Cases:



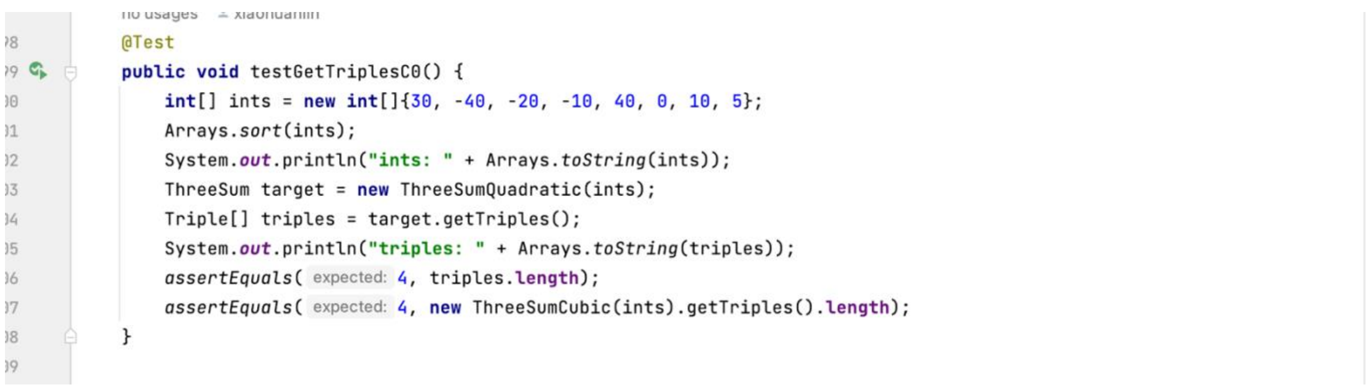
```
no usages   xiaohuanlin
@Test
public void testGetTriplesJ0() {
    int[] ints = new int[]{-2, 0, 2};
    ThreeSumQuadratic target = new ThreeSumQuadratic(ints);
    List<Triple> triples = target.getTriples( i: 1);
    assertEquals( expected: 1, triples.size());
}

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( i: 3);
    assertEquals( expected: 2, triples.size());
}

no usages   xiaohuanlin
@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( i: 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);
}
```



```
@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
@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);
}
```

Observations:

The table below shows the time taken to find Triples in a N length array using the Three Sum Quadratic Method.

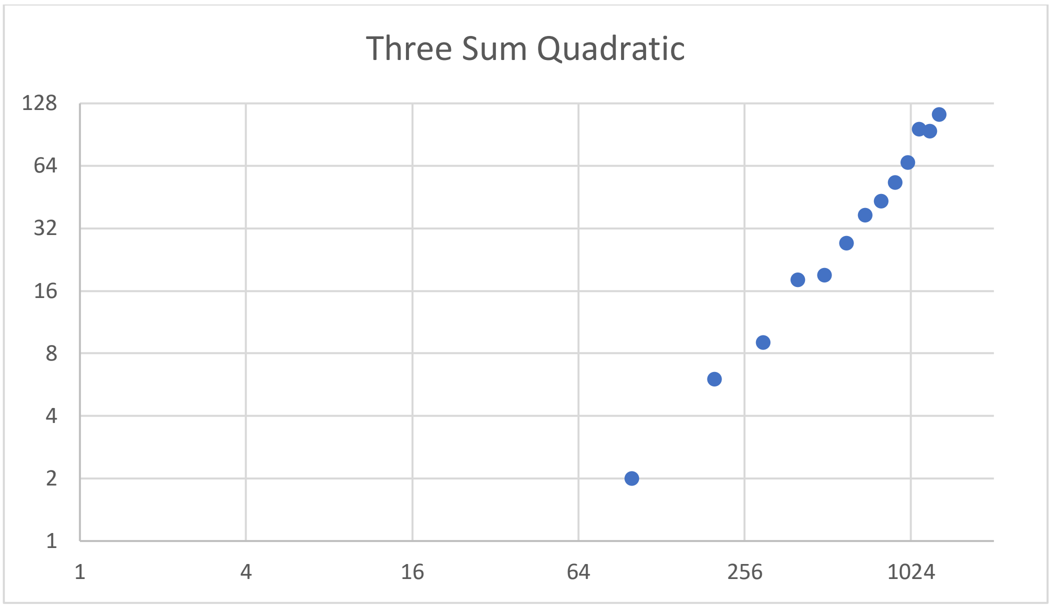
N	T for N^2
100	2
200	6
300	9
400	18
500	19
600	38
700	37
800	43
900	46
1000	66
1100	96
1200	94
1300	113

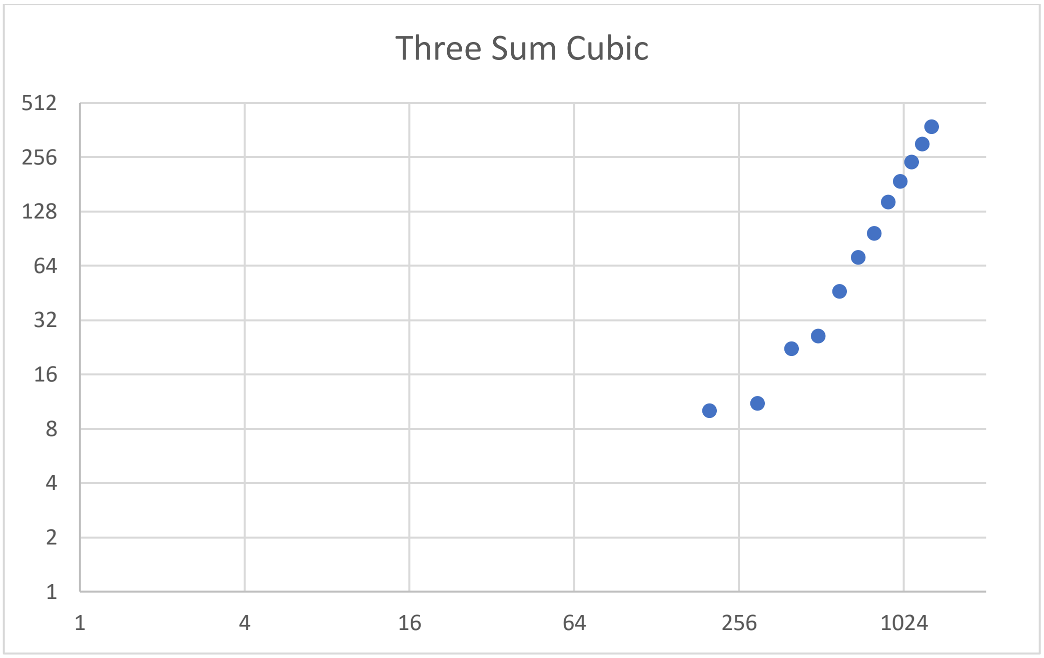
Similarly, this table shows the time taken to find Triples in a N length array using the Three Sum Cubic method.

N	T for N^3
100	2
200	10
300	11
400	22
500	26
600	46
700	71
800	96
900	144
1000	187
1100	239
1200	301
1300	375

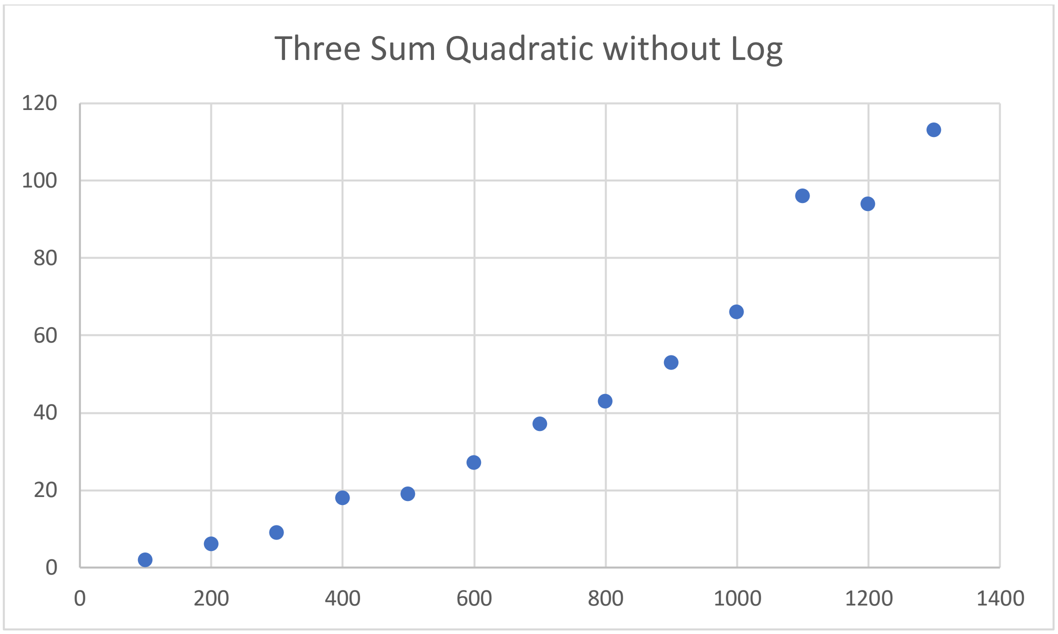
The value of the array lies between -1000 to 999.

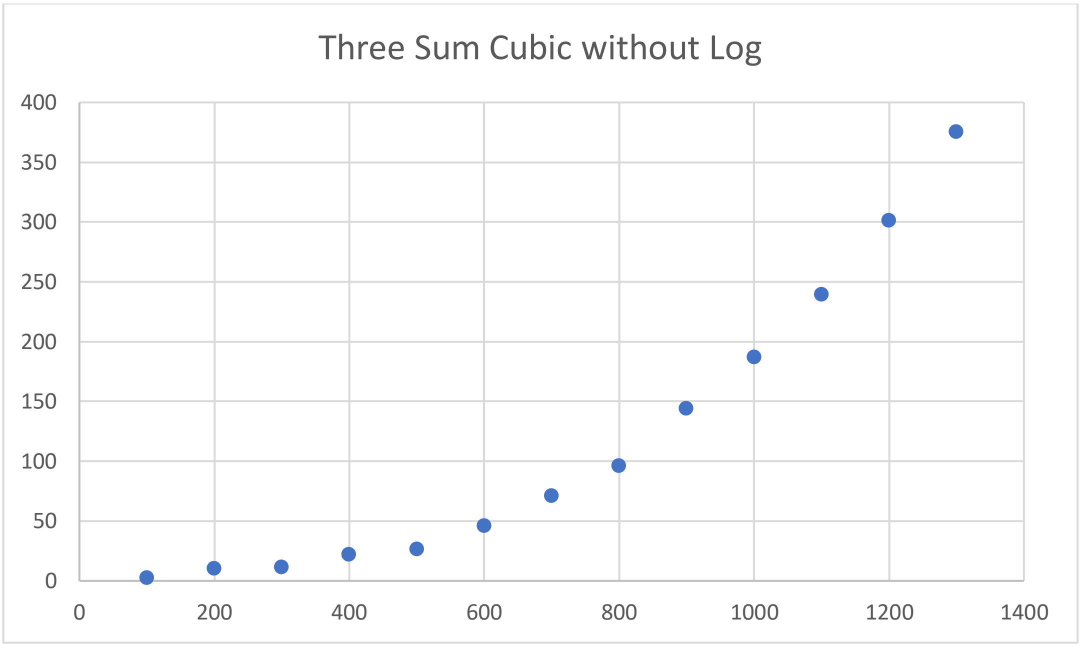
For the above values of N and T, a logarithmic graph is plotted for Log T vs Log N. It is evident that the algorithm to find the triplets using the Quadratic Algorithm is faster compared to Cubic Algorithm.





It can also be seen that the graphs plotted using log scales are linear compared to the graph plotted between T and N.


Three Sum Quadratic without Log
0
20
40
60
80
100
120
0
200
400
600
800
1000
1200
1400


Three Sum Cubic without Log
0
50
100
150
200
250
300
350
400
0
200
400
600
800
1000
1200
1400