Program Structures and Algorithms
Spring 2023(SEC –01)

NAME: **Harshini Venkata Chalam**
NUID: **002934047**

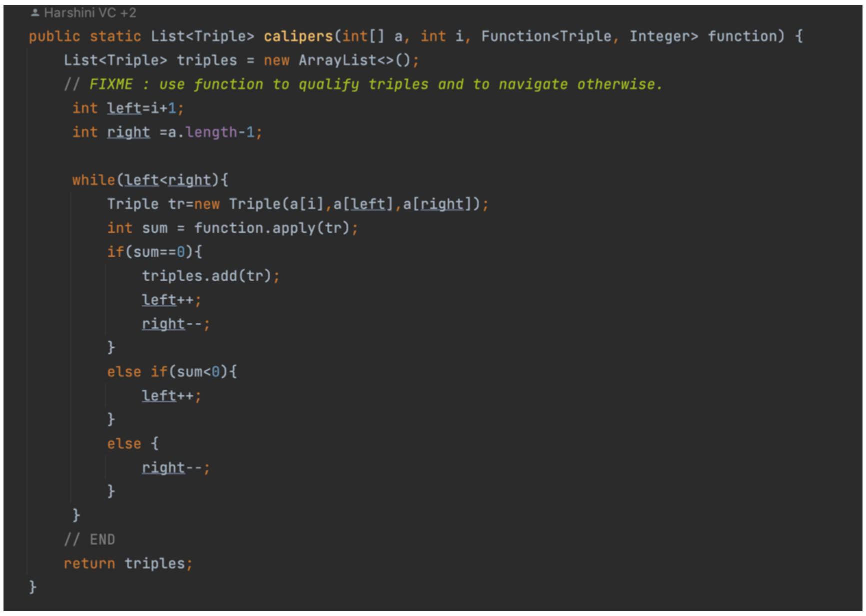
Task:
To Solve 3-SUM using the Quadrithmic, Quadratic, and quadraticWithCalipers approaches. Timing observations of values of N for each of the algorithms.

Relationship Conclusion:

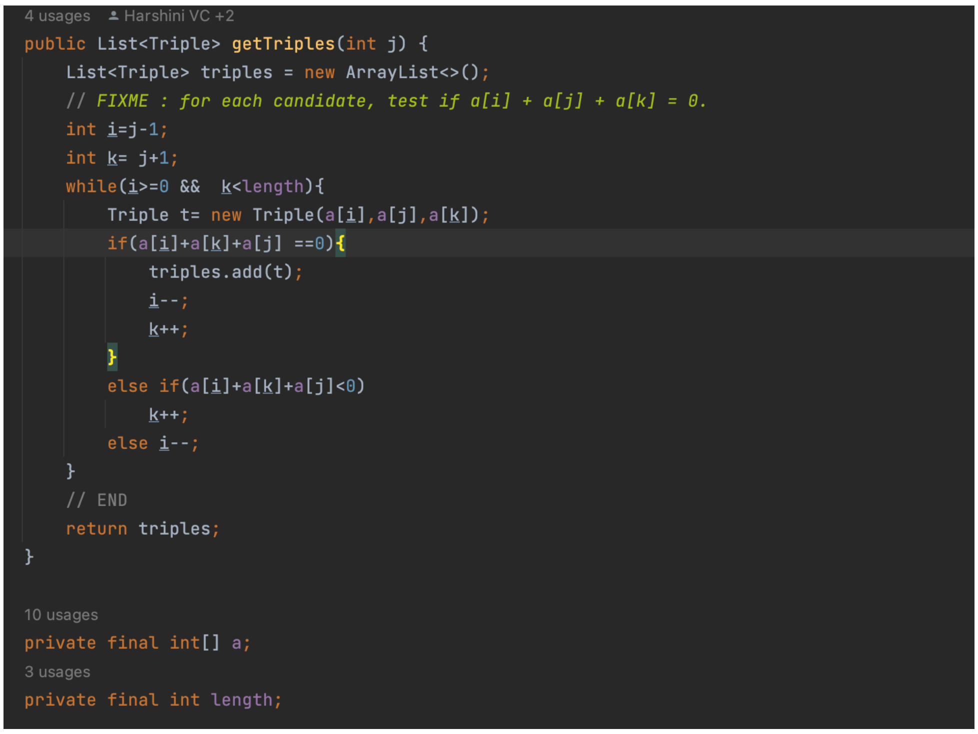
3-Sum for O(N^2) complexity can be solved by considering the [j-1], [j], [j-1] and traversing the array using [j-1] and [j+1] pointers as a[j-1]+a[j]+a[j+1]=0 or a[i] +a[k]= -a[j] .For each number, we iterate through the rest of the set which is n-1 and for n elements .
Comparing to the cubic approach with complexity O(n^3) which involves checking all possible combinations of 3 numbers from the input set and determining if their sum is equal to a target value.
QuadraticWithCalipers, iterates through each number in the set, and for each number, initialized two pointers, one pointing to the next element and the other pointing to the last element of the set. Then for each number we check the sum of the number and numbers pointed to by the two pointers.
If the sum is equal to the target value, return the three numbers or If the sum is less than the target value, move the left pointer to the right or If the sum is greater than the target value, move the right pointer to the left.
Hence it can be concluded after the implementations and observations that quadratic is much faster approach for increasing values of N.

Evidence to support that conclusion:

ThreeSumQuadratic,ThreeSumQuadrithmic,ThreeSumCubic and ThreeSumQuadraticWithCalipers were run through ThreeSumBenchmark to calculate the endTime-StartTime for each of the algorithms for different values of N.


```
Harshini VC +2
public static List<Triple> calipers(int[] a, int i, Function<Triple, Integer> function) {
    List<Triple> triples = new ArrayList<>();
    // FIXME : use function to qualify triples and to navigate otherwise.
     int left=i+1;
     int right =a.length-1;

     while(left<right){
         Triple tr=new Triple(a[i],a[left],a[right]);
         int sum = function.apply(tr);
         if(sum==0){
             triples.add(tr);
             left++;
             right--;
         }
         else if(sum<0){
             left++;
         }
         else {
             right--;
         }
     }
    // END
    return triples;
}
```
ThreeSumQuadraticWithCalipers



```
4 usages   Harshini VC +2
public List<Triple> getTriples(int j) {
    List<Triple> triples = new ArrayList<>();
    // FIXME : for each candidate, test if a[i] + a[j] + a[k] = 0.
    int i=j-1;
    int k= j+1;
    while(i>=0 &&  k<length){
        Triple t= new Triple(a[i],a[j],a[k]);
        if(a[i]+a[k]+a[j] ==0){
            triples.add(t);
            i--;
            k++;
        }
        else if(a[i]+a[k]+a[j]<0)
            k++;
        else i--;
    }
    // END
    return triples;
}

10 usages
private final int[] a;
3 usages
private final int length;
```

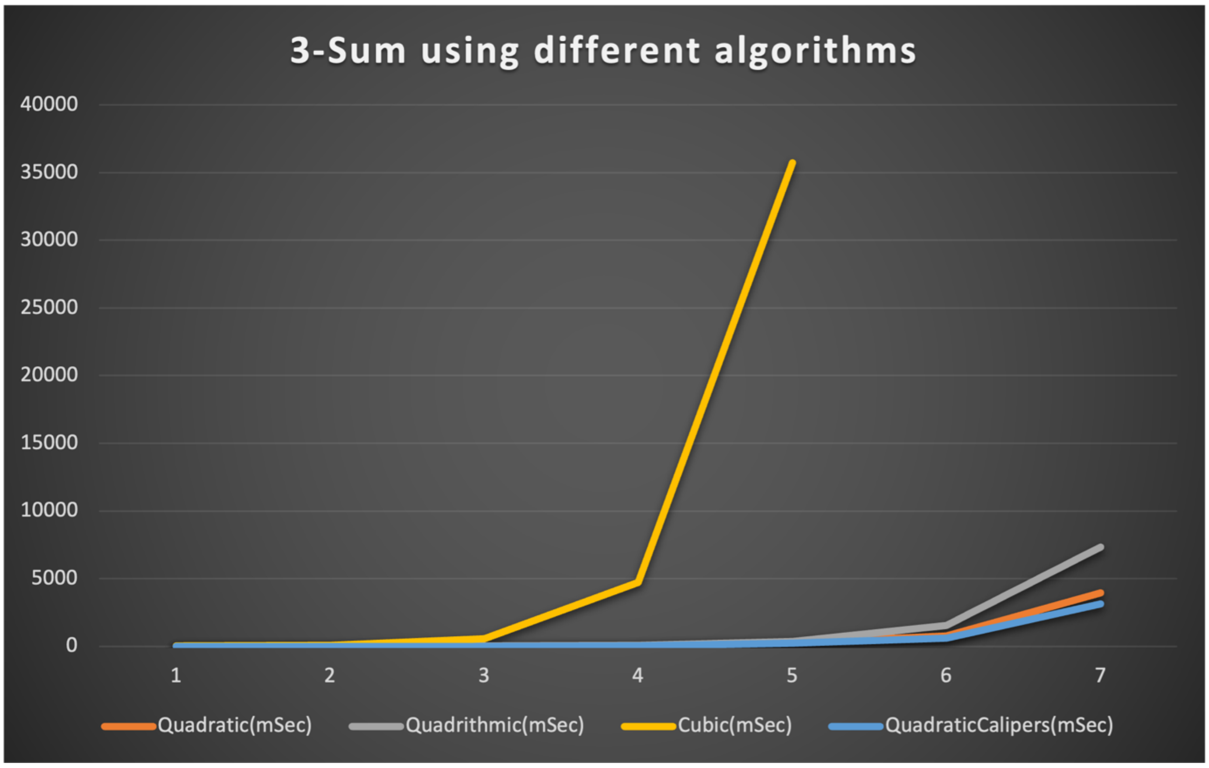
ThreeSumQuadratic

Below is the table of observations for each algorithm timed in mSec for values of N ranging from 250-16000.

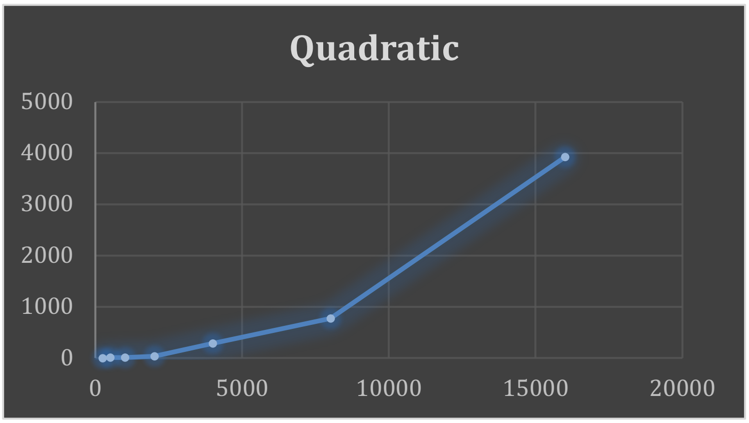
Raw Time per run

N	Quadratic(mSec)	Quadrithmic(mSec)	Cubic(mSec)	QuadraticCalipers(mSec)
250	2.27	1.17	9.52	0.91
500	3.36	3	70.76	1.78
1000	9.2	15	557.15	5.05
2000	34.3	72.4	4718	32.4
4000	288.2	370.4	35724.8	244
8000	768	1527		615.33
16000	3926	7345		3136

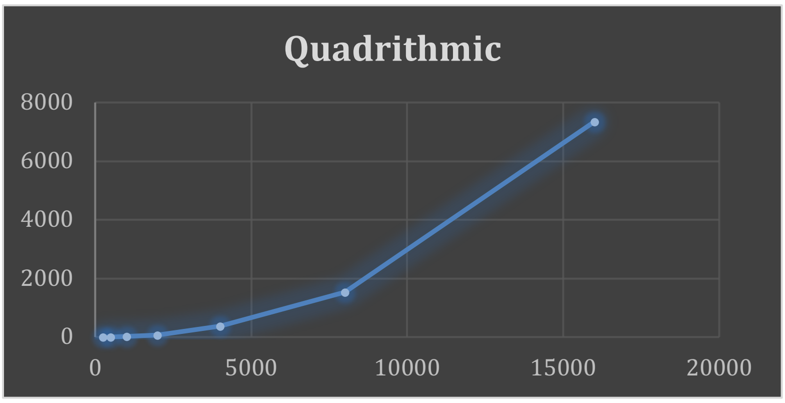
Graphical Representation:



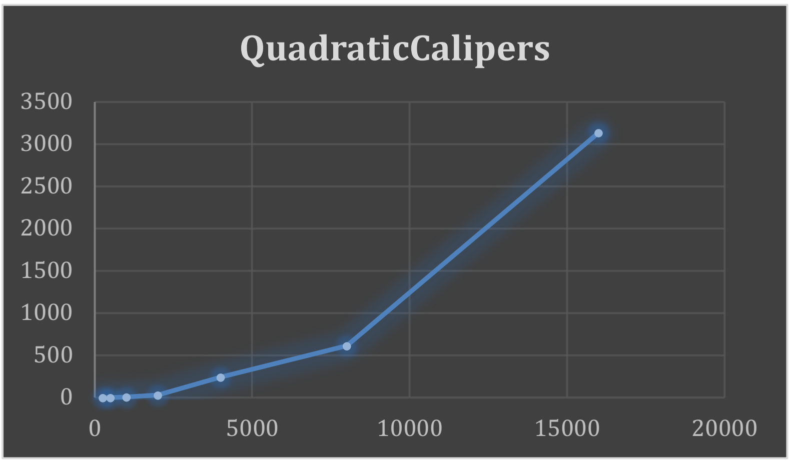
N	Quadratic
250	2.27
500	3.36
1000	9.2
2000	34.3
4000	288.2
8000	768
16000	3926



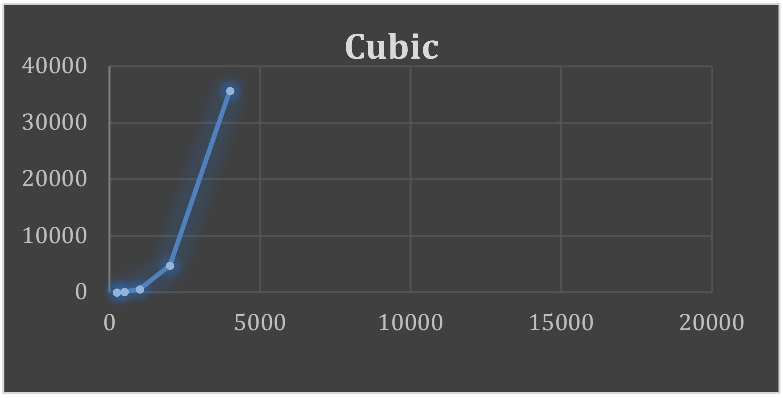
N	Quadrithmic
250	1.17
500	3
1000	15
2000	72.4
4000	370.4
8000	1527
16000	7345



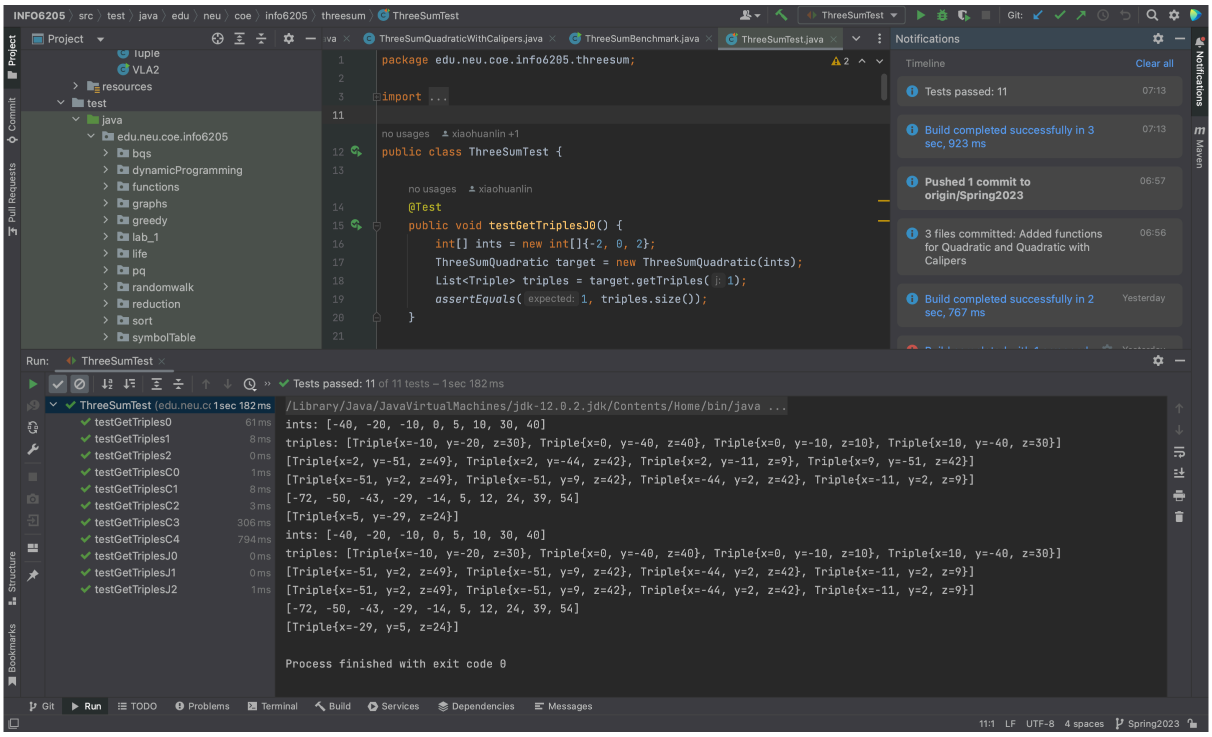
N	QuadraticCalipers
250	0.91
500	1.78
1000	5.05
2000	32.4
4000	244
8000	615.33
16000	3136



N	Cubic
250	9.52
500	70.76
1000	557.15
2000	4718
4000	35724.8
8000	
16000	



Unit Test Screenshots:



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

import ...

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());
    }
```

Notifications

- Tests passed: 11 (07:13)
- Build completed successfully in 3 sec, 923 ms (07:13)
- Pushed 1 commit to origin/Spring2023 (06:57)
- 3 files committed: Added functions for Quadratic and Quadratic with Calipers (06:56)
- Build completed successfully in 2 sec, 767 ms (Yesterday)

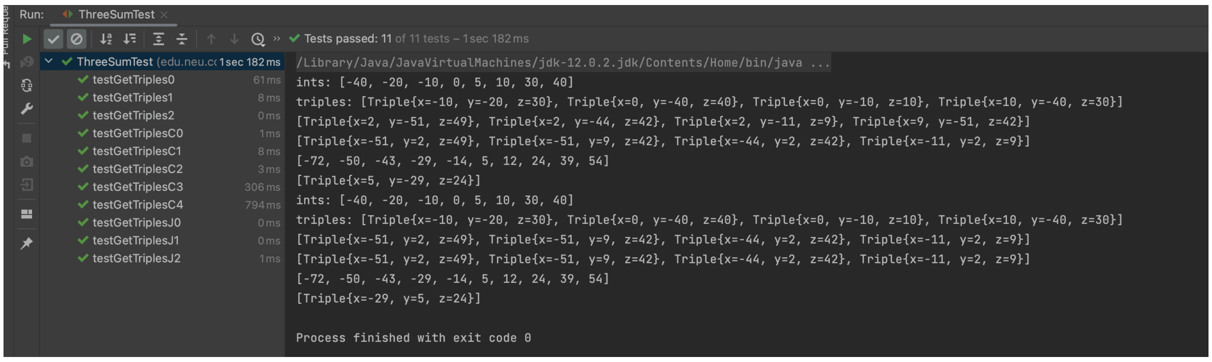
Run: ThreeSumTest

Tests passed: 11 of 11 tests – 1 sec 182 ms

```
ThreeSumTest (edu.neu.cc 1 sec 182 ms
  testGetTriples0      61 ms
  testGetTriples1       8 ms
  testGetTriples2       0 ms
  testGetTriplesC0      1 ms
  testGetTriplesC1      8 ms
  testGetTriplesC2      3 ms
  testGetTriplesC3    306 ms
  testGetTriplesC4    794 ms
  testGetTriplesJ0      0 ms
  testGetTriplesJ1      0 ms
  testGetTriplesJ2      1 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
ints: [-40, -20, -10, 0, 5, 10, 30, 40]
triples: [Triple{x=-10, y=-20, z=30}, Triple{x=0, y=-40, z=40}, Triple{x=0, y=-10, z=10}, Triple{x=10, y=-40, z=30}]
[Triple{x=2, y=-51, z=49}, Triple{x=2, y=-44, z=42}, Triple{x=2, y=-11, z=9}, Triple{x=9, y=-51, z=42}]
[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=5, y=-29, z=24}]
ints: [-40, -20, -10, 0, 5, 10, 30, 40]
triples: [Triple{x=-10, y=-20, z=30}, Triple{x=0, y=-40, z=40}, Triple{x=0, y=-10, z=10}, Triple{x=10, y=-40, z=30}]
[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}]

Process finished with exit code 0
```



Run: ThreeSumTest

Tests passed: 11 of 11 tests – 1 sec 182 ms

```
ThreeSumTest (edu.neu.cc 1 sec 182 ms
  testGetTriples0      61 ms
  testGetTriples1       8 ms
  testGetTriples2       0 ms
  testGetTriplesC0      1 ms
  testGetTriplesC1      8 ms
  testGetTriplesC2      3 ms
  testGetTriplesC3    306 ms
  testGetTriplesC4    794 ms
  testGetTriplesJ0      0 ms
  testGetTriplesJ1      0 ms
  testGetTriplesJ2      1 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
ints: [-40, -20, -10, 0, 5, 10, 30, 40]
triples: [Triple{x=-10, y=-20, z=30}, Triple{x=0, y=-40, z=40}, Triple{x=0, y=-10, z=10}, Triple{x=10, y=-40, z=30}]
[Triple{x=2, y=-51, z=49}, Triple{x=2, y=-44, z=42}, Triple{x=2, y=-11, z=9}, Triple{x=9, y=-51, z=42}]
[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=5, y=-29, z=24}]
ints: [-40, -20, -10, 0, 5, 10, 30, 40]
triples: [Triple{x=-10, y=-20, z=30}, Triple{x=0, y=-40, z=40}, Triple{x=0, y=-10, z=10}, Triple{x=10, y=-40, z=30}]
[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}]

Process finished with exit code 0
```