

Program Structures and Algorithms

Spring 2023 (SEC - 01)

NAME: Aravind Dasarathy

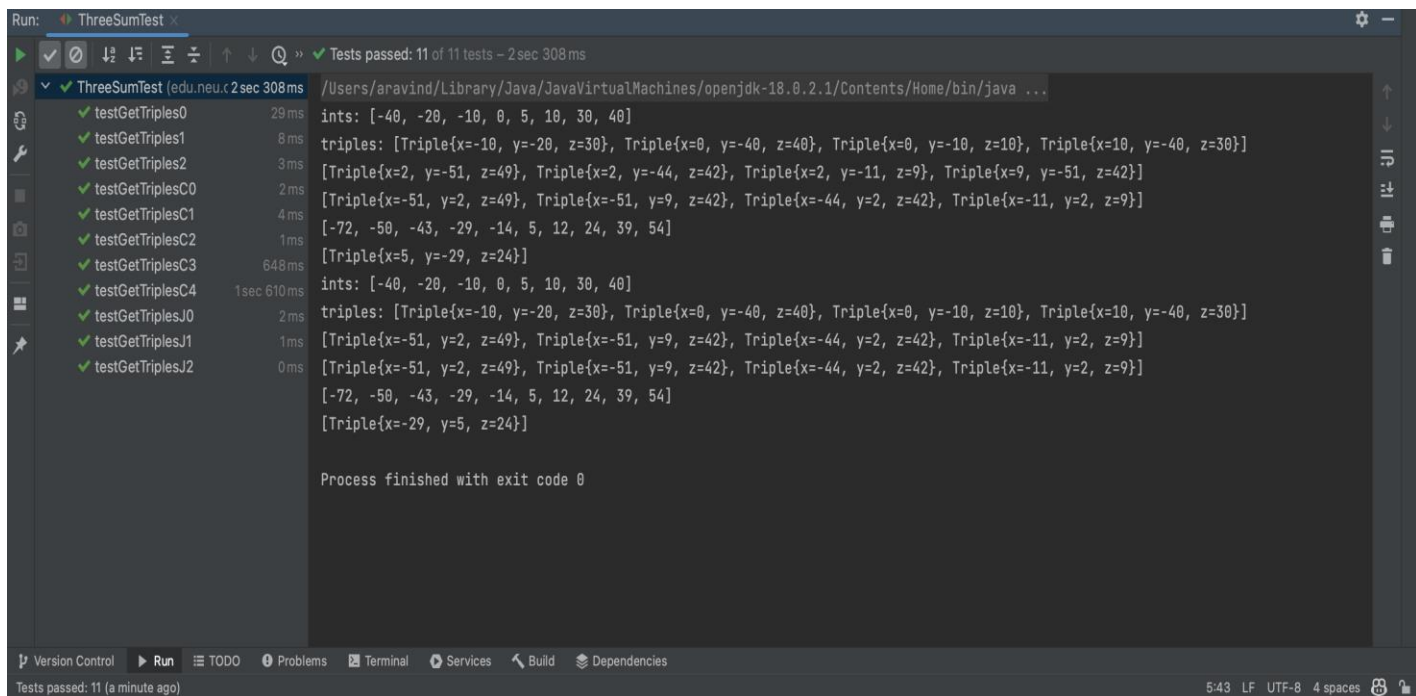
NUID: 002130918

Task:

The task is to solve the three-sum problem using Quadrithmic, Quadratic, and Quadratic with caliper approaches and perform timing observations by varying the algorithm parameters.

Unit Tests:

All test cases are passed with respect to the ThreeSum project.



```
Run: ThreeSumTest x
Tests passed: 11 of 11 tests - 2 sec 308 ms

ThreeSumTest (edu.neu.c 2 sec 308 ms)
  testGetTriples0 29 ms ints: [-40, -20, -10, 0, 5, 10, 30, 40]
  testGetTriples1 8 ms 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}]
  testGetTriples2 3 ms [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}]
  testGetTriplesC0 2 ms [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}]
  testGetTriplesC1 4 ms [-72, -50, -43, -29, -14, 5, 12, 24, 39, 54]
  testGetTriplesC2 1 ms [Triple{x=5, y=-29, z=24}]
  testGetTriplesC3 648 ms ints: [-40, -20, -10, 0, 5, 10, 30, 40]
  testGetTriplesC4 1 sec 610 ms 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}]
  testGetTriplesJ0 2 ms [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}]
  testGetTriplesJ1 1 ms [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}]
  testGetTriplesJ2 0 ms [-72, -50, -43, -29, -14, 5, 12, 24, 39, 54]
  [Triple{x=-29, y=5, z=24}]

Process finished with exit code 0

Version Control Run TODO Problems Terminal Services Build Dependencies
Tests passed: 11 (a minute ago) 5:43 LF UTF-8 4 spaces
```

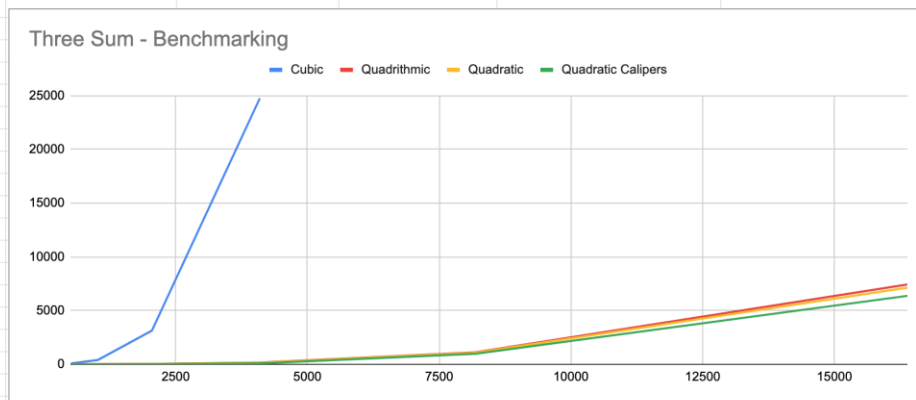
Timing Observation:

I've created a class named 'TimedThreeSum.java' to perform the timing observations.

Array Length	Cubic	Quadrithmic	Quadratic	Quadratic Caliper
512	77	6	5	5
1024	400	12	13	7
2048	3124	31	33	17
4096	24758	172	140	77
8192	-	1124	1079	981
16384	-	7421	7133	6370

Graphical Observations:

Array Length	Cubic (ms)	Quadrithmic (ms)	Quadratic (ms)	Quadratic Caliper (ms)
512	77	6	5	5
1024	400	12	13	7
2048	3124	31	33	17
4096	24758	172	140	77
8192	-	1124	1079	981
16384	-	7421	7133	6370



Logical Conclusion:

From the timing observations,

1. We can clearly notice that the cubic complexity is the poor performer as the time complexity of it is $O(n^3)$.
2. The Quadrithmic is the second poor performer as its time complexity of it is $O(n^2 * \log(n))$. The quadratic component is because of the two for-loops nested inside each other and the last ' $\log(n)$ ' component is because of the binary search being performed to search the complement of the sum performed in the top two nested loops.

3. The Quadratic Calipers algorithm performs a bit better than the Quadratic algorithm.

The explanation for point 3 is because of how the algorithms perform traverse in the nested loops using two pointers. In the quadratic approach, the two pointers spread outward with reference to the middle element. Hence, some elements will be traversed more than once.

On the other hand, the quadratic caliper approach does not do that. For each two-pointer traversal, the size of the traversal reduces but in the case of quadratic, the search space will always be the length of the input array. This is the reason why the quadratic caliper approach will be a bit faster than the normal quadratic approach, but the time complexity is $O(n^2)$ for both approaches.