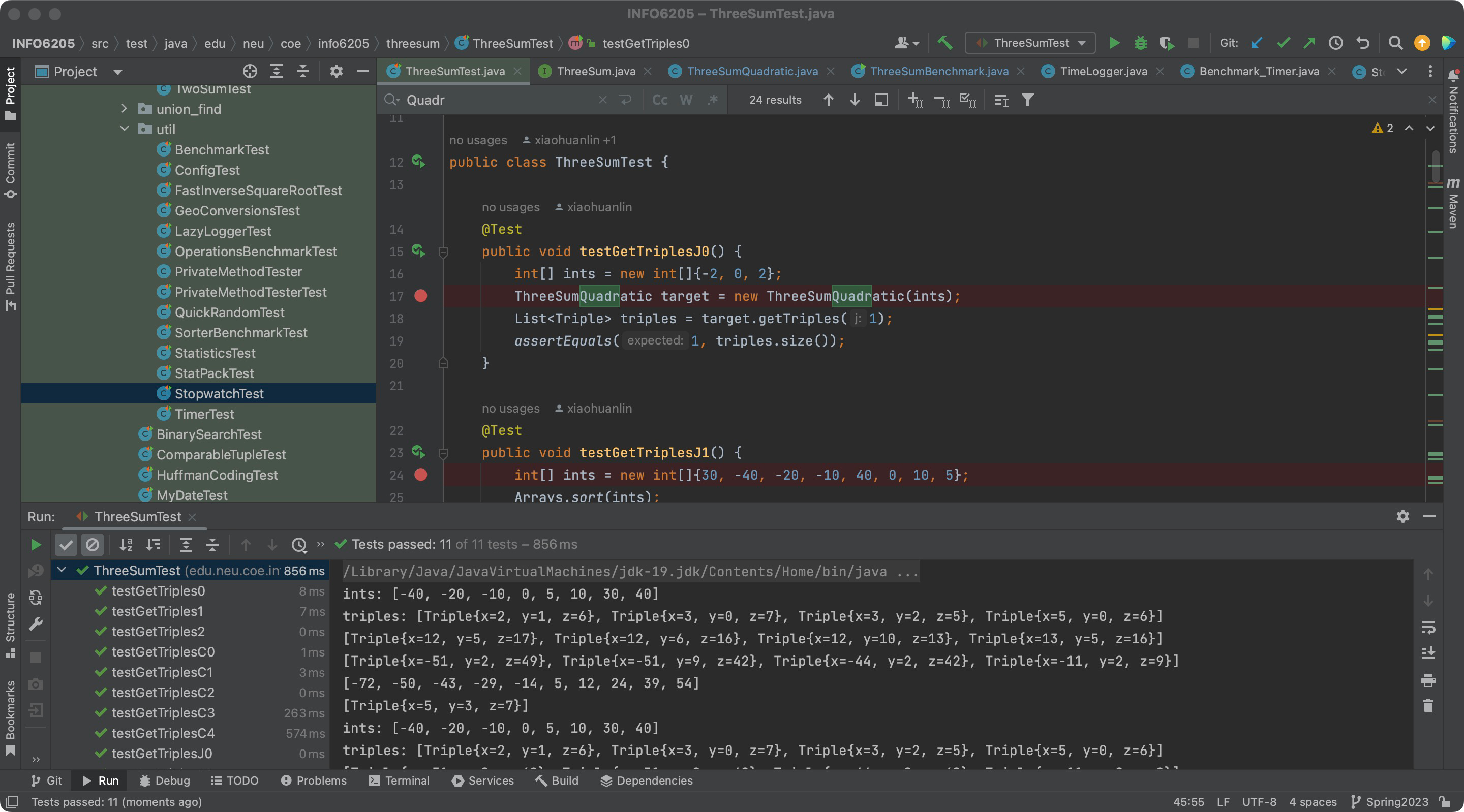
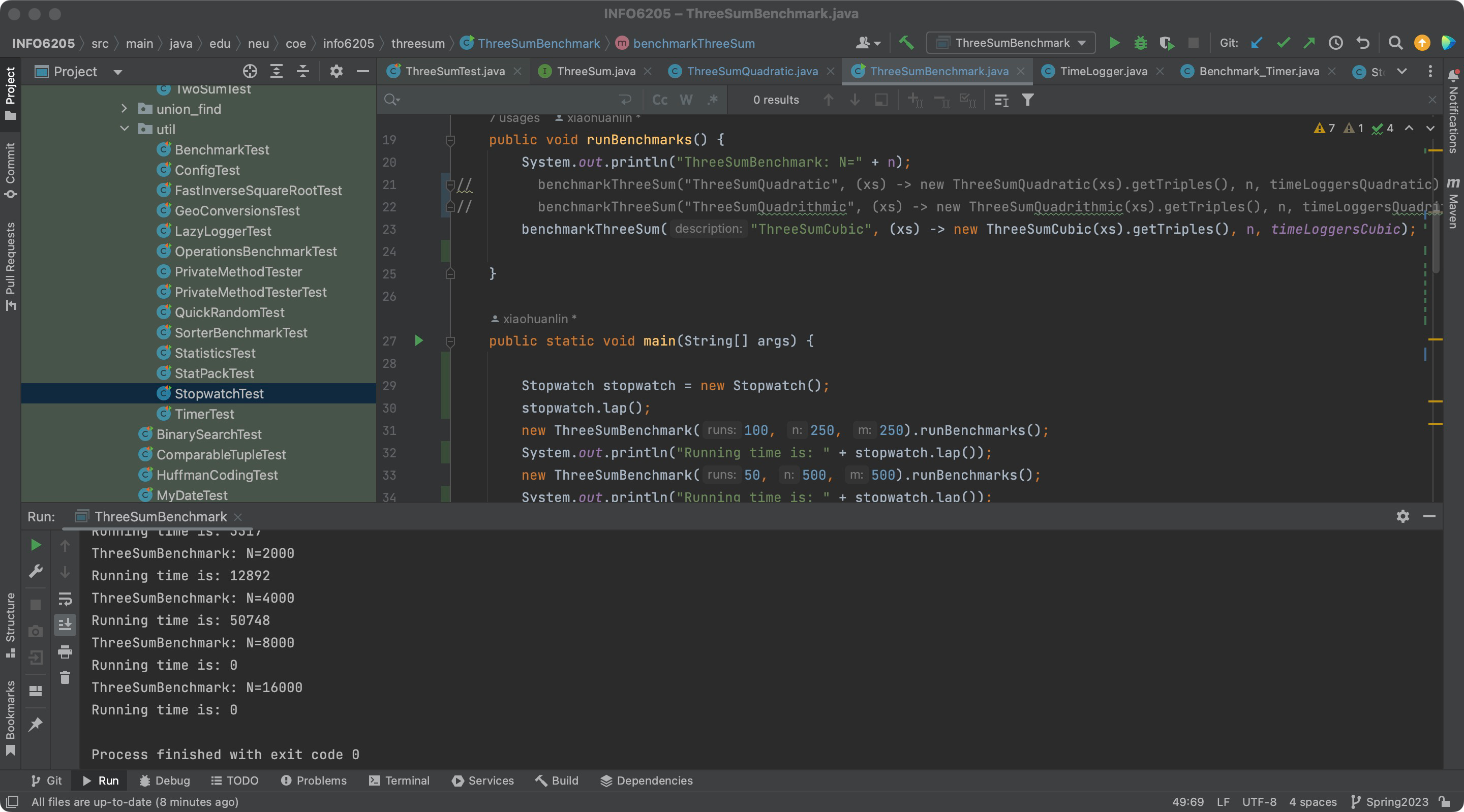
This is the report for INFO6205 Assignment 2.

1. evidence of running tests.



1. By filling code in ThreeSumBenchmark, we tests three methods (which are already implemented in corresponding classes) ,and then record and print out the time (million second / ms) they took by using the stopwatch instances. The results are listed below:
2. Cubic

Screenshot.



Since the time complexity is comparably high in Cubic method, tests with length that greater than 4000 is skipped by default in this program. Results are as given.

cubic

ThreeSumBenchmark: N=250

Running time is: 342

ThreeSumBenchmark: N=500

Running time is: 1097

ThreeSumBenchmark: N=1000

Running time is: 3374

ThreeSumBenchmark: N=2000

Running time is: 12833

ThreeSumBenchmark: N=4000

Running time is: 50915

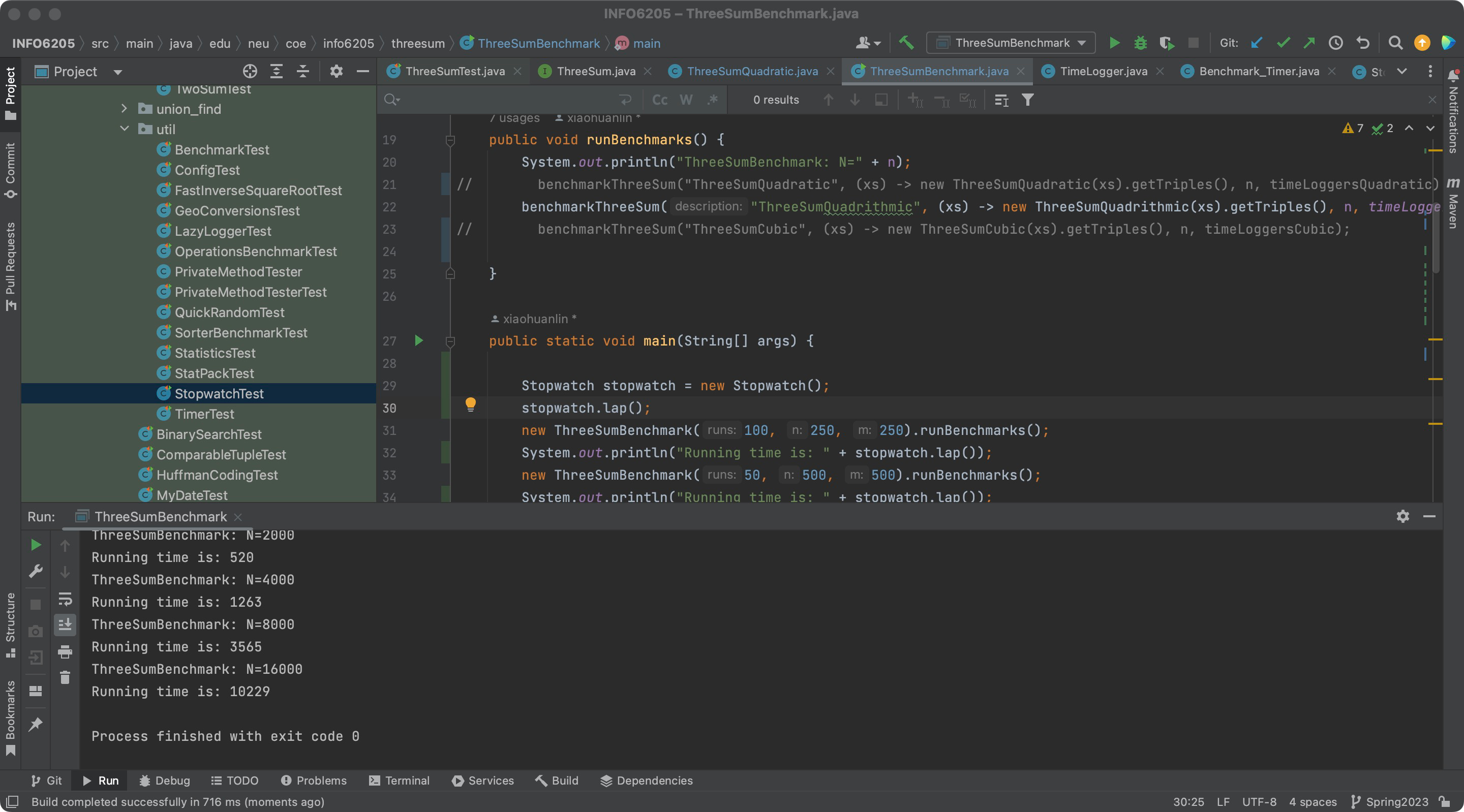
ThreeSumBenchmark: N=8000

Running time is: 0

ThreeSumBenchmark: N=16000

Running time is: 0

1. Quadrithmic



ThreeSumBenchmark: N=250

Running time is: 91

ThreeSumBenchmark: N=500

Running time is: 135

ThreeSumBenchmark: N=1000

Running time is: 224

ThreeSumBenchmark: N=2000

Running time is: 519

ThreeSumBenchmark: N=4000

Running time is: 1258

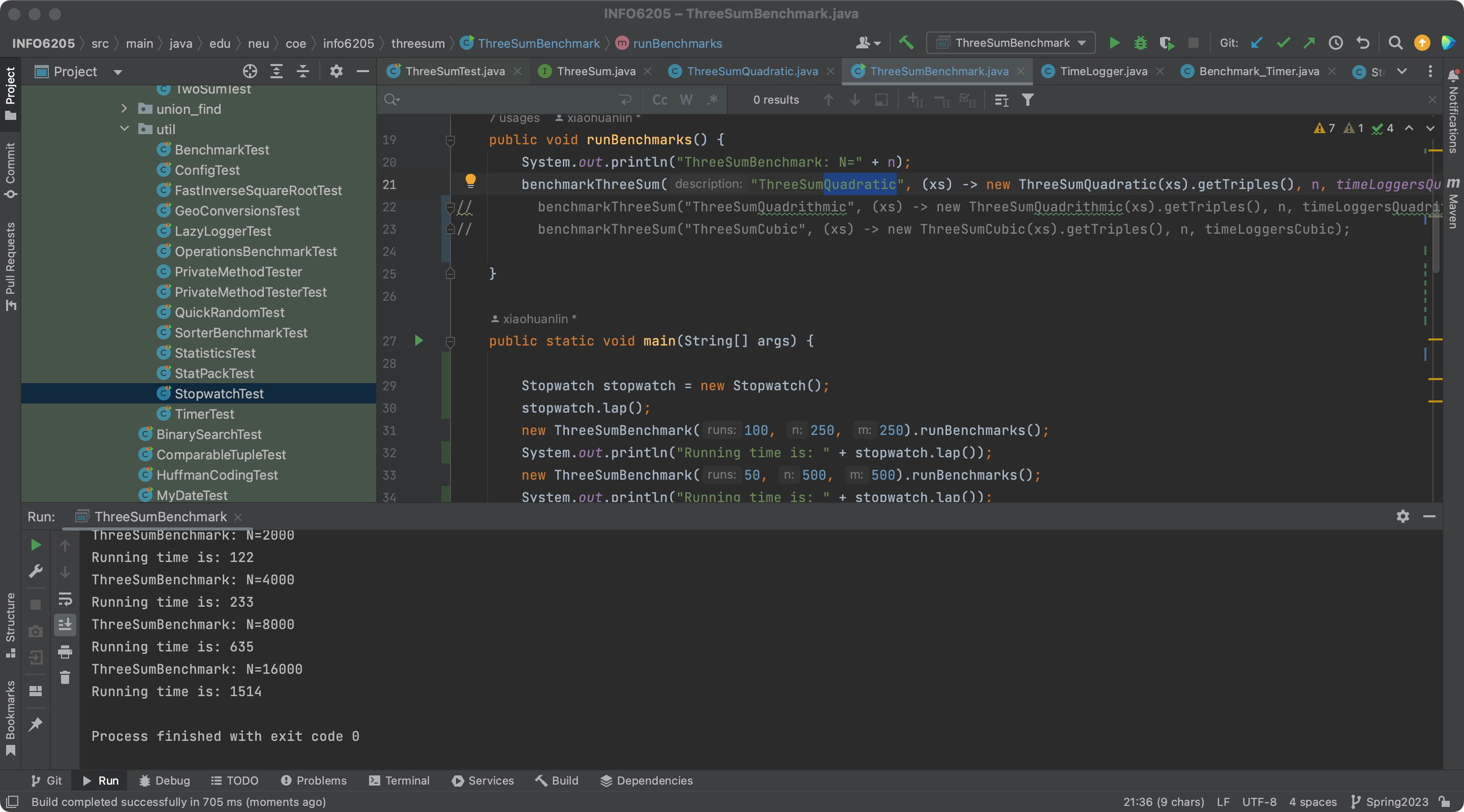
ThreeSumBenchmark: N=8000

Running time is: 3564

ThreeSumBenchmark: N=16000

Running time is: 10150

1. quadratic



ThreeSumBenchmark: N=250

Running time is: 73

ThreeSumBenchmark: N=500

Running time is: 71

ThreeSumBenchmark: N=1000

Running time is: 93

ThreeSumBenchmark: N=2000

Running time is: 122

ThreeSumBenchmark: N=4000

Running time is: 233

ThreeSumBenchmark: N=8000

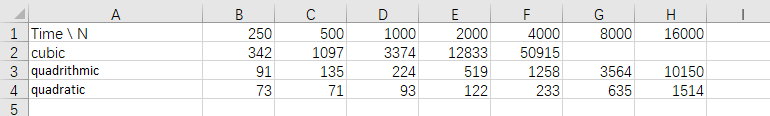
Running time is: 635

ThreeSumBenchmark: N=16000

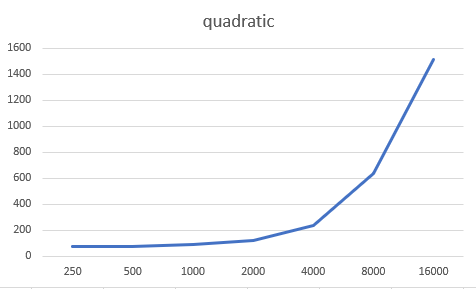
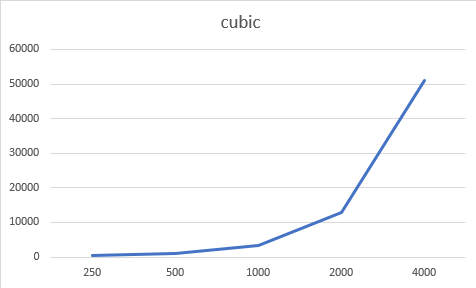
Running time is: 1514

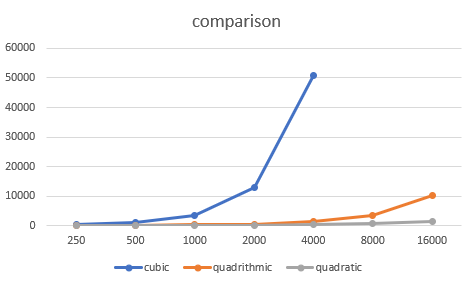
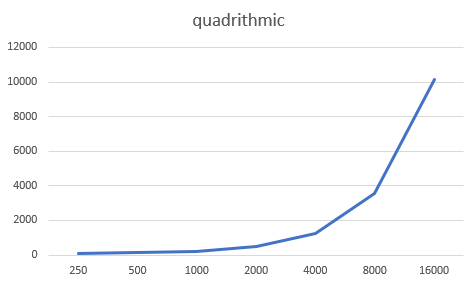
Analyze.

Now we have form like this:



We Draw diagrams as followed, where the x-axis means the length of int array, while the y-axis represents the running time (million seconds):





By out analyzation on default, we have the time complexity of ‘cubic’ is O(n^3), while the later two is O(n^2). However, here we found that the running time (in ms) cannot be perfectly written in y = c \* n^3 or y = c \* n^2. This might be because the running time is not strictly determined by cubic / square of array’s length; other calculations is also something in this project.

But we do have that changing rate of ‘cubic’ is faster than the other 2 lines, since when n rise from 2000 to 4000, the running time became considerable large: almost 4 times growth.

c) The quadratic method works because the two pointer is a great way of dealing with sorted array. By adjusting pointers, the program managed to iterate single array (with length n) in time complexity of O(n), and ensure no pairs (that sum of them equals to target value) should be ignored.

After proceeding this method, we can reach all desired companions with another outer loop, so that the inner two-sum problem has a target value.