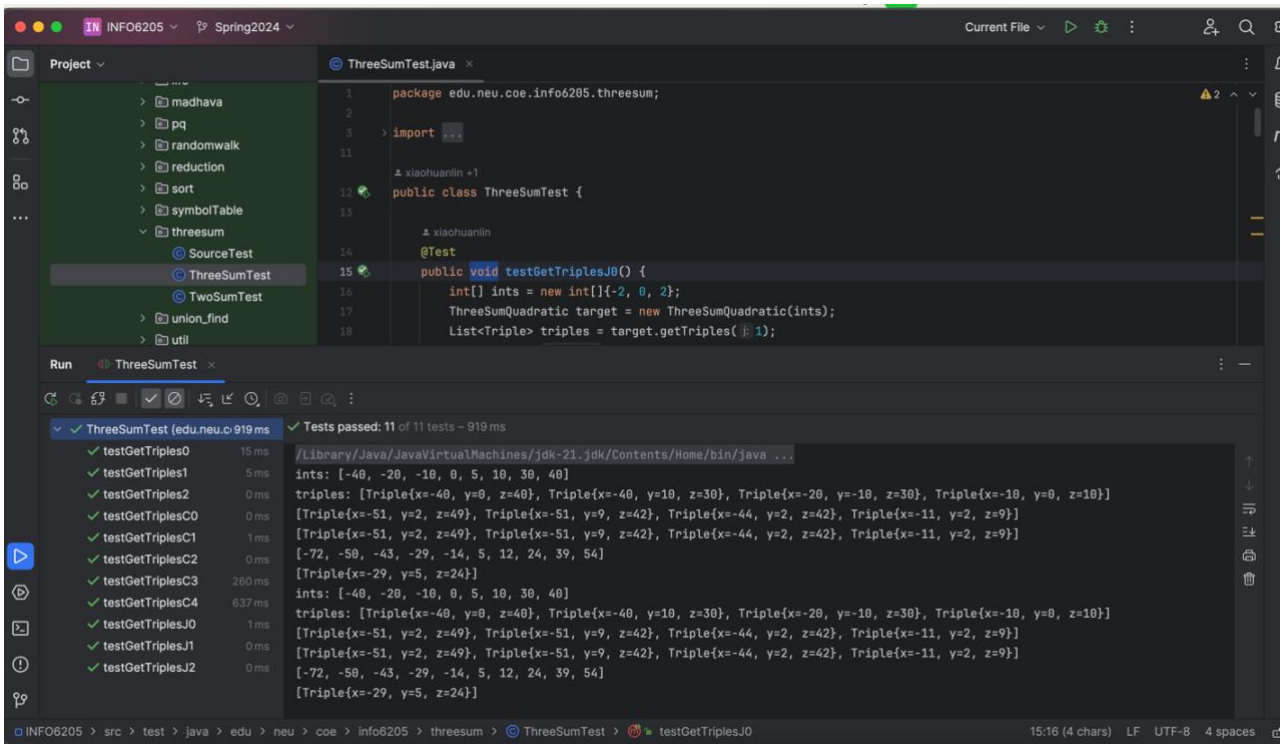
Program Structures and Algorithms
Assignment 2 (3-SUM)
Spring 2024

NAME: Suchita Arvind Dabir
NUID: 002957879
GITHUB LINK: https://github.com/suchitadabir/INFO6205

Task: Solve 3-SUM using the Quadrithmic, Quadratic, and QuadraticWithCalipers

(a) Unit Test Screenshots:

All 11 of 11 tests passed:



(b) Timing Observations:

I used Stopwatch class in the repository to measure the run time for ThreeSumQuadratic, ThreeSumQuadrithmic, ThreeSumCubic, and ThreeSumQuadraticWithCalipers algorithms. I choose seven different values of N as 250, 500, 1000, 2000, 4000, 8000 and 16000 and finally I modify the Benchmark file to dump all this data to a csv file. A table below shows run time in millisecond for each of this algorithm at chosen value of N.

Function	N	Time (ms)
ThreeSumQuadratic	250	3
ThreeSumQuadrithmic	250	2
ThreeSumCubic	250	8
ThreeSumQuadraticWithCalipers	250	1
ThreeSumQuadratic	500	3
ThreeSumQuadrithmic	500	4
ThreeSumCubic	500	31
ThreeSumQuadraticWithCalipers	500	1
ThreeSumQuadratic	1000	9
ThreeSumQuadrithmic	1000	18
ThreeSumCubic	1000	210
ThreeSumQuadraticWithCalipers	1000	4
ThreeSumQuadratic	2000	20
ThreeSumQuadrithmic	2000	53
ThreeSumCubic	2000	1292
ThreeSumQuadraticWithCalipers	2000	10
ThreeSumQuadratic	4000	79
ThreeSumQuadrithmic	4000	247
ThreeSumCubic	4000	10682
ThreeSumQuadraticWithCalipers	4000	85
ThreeSumQuadratic	8000	421
ThreeSumQuadrithmic	8000	1216
ThreeSumQuadraticWithCalipers	8000	293
ThreeSumQuadratic	16000	1827
ThreeSumQuadrithmic	16000	5231
ThreeSumQuadraticWithCalipers	16000	1698

(c) Explanation why the quadratic method(s) work:

I wrote a simple python script to plot the run time in millisecond along Y-axis and N across X-axis.

For ThreeSumCubic algorithm (green line), the time complexity is $O(n^3)$; i.e., the time it takes to run the algorithm is proportional to the cube of number of input elements. Hence, as the number of input increases from 2000 to 4000, the run time spikes sharply from ~1.5 to ~11 seconds. And same pattern would have been continued if we increase the N beyond 4000.

On the contrary, the time complexity for ThreeSumQuadrithmic is $O(n^2 \log n)$. Hence, the rate of increase in run time for ThreeSumQuadrithmic algorithm (orange line) is much smaller than that of ThreeSumCubic.

Moreover, the other two algorithms ThreeSumQuadratic and ThreeSumQuadraticWithCalipers have the time complexity of $O(n^2)$ with the later one being slightly faster than former. The reason for this is that instead of looping thrice, we take advantage of array being sorted and use 2 pointers to move across the array. This includes moving (i) from left to right and (k) from right to left until both converge at the middle index (j) in case of ThreeSumQuadratic algorithm OR moving (j) and (k) towards the center until they converge like in case of Caliper. Thus, we only need two loops – one to iterate over given array where one index is fixed and another loop to get all the triples with sum = 0. Hence, the overall running time gets proportional to the square of number of inputs which is significantly (N times) lower than cube of number of inputs. That's why Quadratic methods (red and blue lines) work effectively over other mentioned algorithms.