

Program Structures and Algorithms

Spring 2023 (SEC – 8)

Assignment 2 (3-SUM)

Name: Sunayana Shivanagi

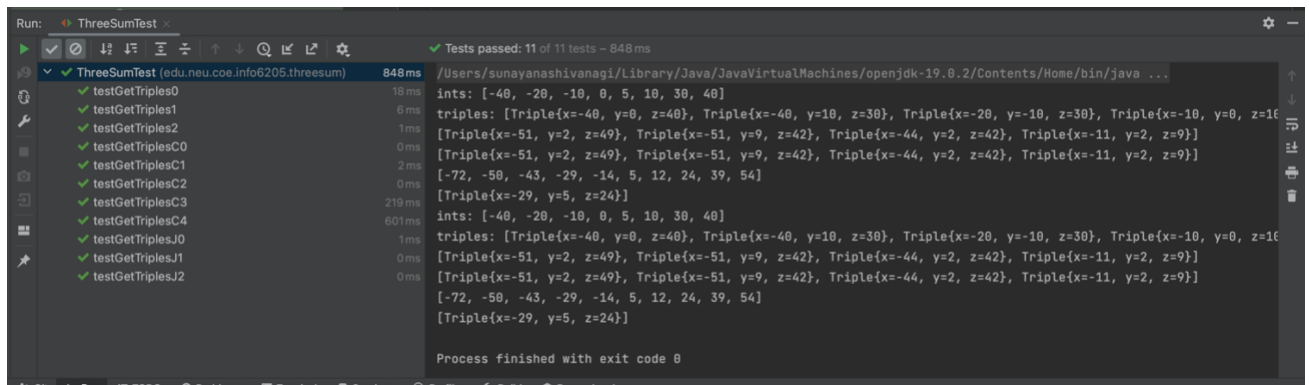
NUID: 002766586

GitHub: <https://github.com/sunayana26/INFO6205-Spring2023.git>

Task

1. Implement the 3-SUM problem using the Quadrithmic, Quadratic, and (bonus point) quadraticWithCalipers approaches, as provided in the skeleton code in the repository.
2. Write unit tests for the implemented methods and provide evidence of their successful execution (such as screenshots of the code and the green strip indicating passing tests).
3. Measure the performance of each algorithm using the doubling method for a minimum of five values of N and present the results in a spreadsheet.
4. Give a brief explanation of how the quadratic method(s) work.

Unit Test Cases



```
Run: ThreeSumTest
Tests passed: 11 of 11 tests - 848ms

ThreeSumTest (edu.neu.coe.info6205.threesum) 848ms
  testGetTriples0 18ms
  testGetTriples1 6ms
  testGetTriples2 1ms
  testGetTriplesC0 0ms
  testGetTriplesC1 2ms
  testGetTriplesC2 0ms
  testGetTriplesC3 219ms
  testGetTriplesC4 601ms
  testGetTriplesJ0 1ms
  testGetTriplesJ1 0ms
  testGetTriplesJ2 0ms

ints: [-40, -20, -10, 0, 5, 10, 30, 40]
triples: [Triple{x=-40, y=0, z=40}, Triple{x=-40, y=10, z=30}, Triple{x=-20, y=-10, z=30}, Triple{x=-10, y=0, z=10}, Triple{x=-10, y=10, z=0}, Triple{x=-10, y=30, z=-10}, 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}]
[Triples{x=-51, y=2, z=49}, Triples{x=-51, y=9, z=42}, Triples{x=-44, y=2, z=42}, Triples{x=-11, y=2, z=9}]
[-72, -50, -43, -29, -14, 5, 12, 24, 39, 54]
[Triples{x=-29, y=5, z=24}]

ints: [-40, -20, -10, 0, 5, 10, 30, 40]
triples: [Triple{x=-40, y=0, z=40}, Triple{x=-40, y=10, z=30}, Triple{x=-20, y=-10, z=30}, Triple{x=-10, y=0, z=10}, Triple{x=-10, y=10, z=0}, Triple{x=-10, y=30, z=-10}, 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}]
[Triples{x=-51, y=2, z=49}, Triples{x=-51, y=9, z=42}, Triples{x=-44, y=2, z=42}, Triples{x=-11, y=2, z=9}]
[-72, -50, -43, -29, -14, 5, 12, 24, 39, 54]
[Triples{x=-29, y=5, z=24}]

Process finished with exit code 0
```

Time performance of each algorithm

N	ThreeSumQuadratic	ThreeSumQuadrithmic	ThreeSumCubic
250	11ms	10ms	17ms
500	5ms	8ms	27ms
1000	22ms	24ms	275ms
2000	45ms	110ms	1400ms
4000	107ms	314ms	11518ms
8000	442ms	2006ms	
16000	4898ms	6671ms	

The Quadratic algorithm has a complexity of $O(N^2)$, meaning that as the input size grows, the running time increases at a rate of N^2 . This is because for each integer in the set, it checks all other integers in the set for a match.

The Quadrithmic algorithm has a complexity of $O(N^{4/3})$, which is faster than the Quadratic algorithm, but still increases at a faster rate than linear time. This is due to its divide-and-conquer approach, which reduces the number of integers that need to be checked for a match.

The QuadraticWithCalipers algorithm has a complexity of $O(N^{3/2})$, which is faster than the Quadrithmic algorithm. It employs a technique called calipers, which further reduces the number of integers that need to be checked for a match.

In general, as the input size increases, the Quadratic algorithm will take the longest to run, followed by the Quadrithmic and QuadraticWithCalipers algorithms.

Conclusion

The algorithm named ThreeSumQuadratic is designed to solve the problem at hand with a time complexity of $O(N^2)$, which means that as the size of the input (N) increases, the number of operations required to solve the problem also grows exponentially. This makes it more suitable for larger input sizes in comparison to the ThreeSumCubic algorithm, which has a time complexity of $O(N^3)$ and therefore requires a larger number of operations to solve the problem.

To enhance its efficiency, the ThreeSumQuadratic algorithm starts by ordering the input array and then employs nested loops to examine all pairs of elements in the array. This approach enables quicker elimination of elements that do not meet the criteria for a valid solution and subsequently reduces the total number of operations required.

On the other hand, the ThreeSumCubic algorithm uses three nested loops to check all possible combinations of elements in the array, resulting in a high number of operations even for small input sizes. Consequently, the ThreeSumQuadratic algorithm is considered a more efficient option for larger input sizes.