Program Structures and Algorithms
Fall 2023

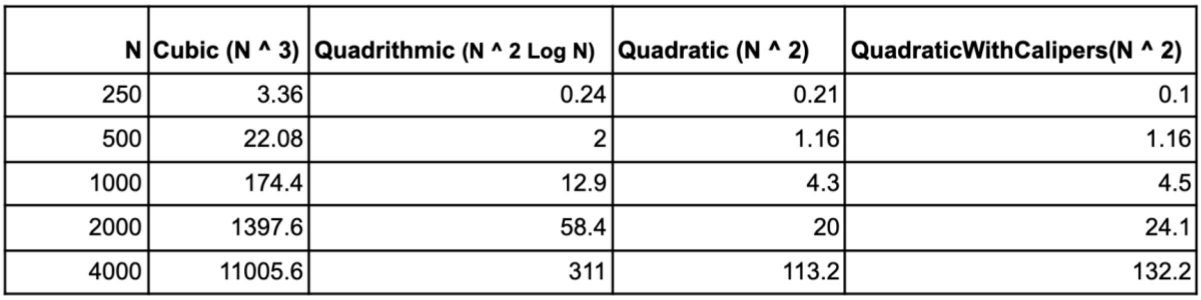
NAME:Shangqing Hu
NUID:001374342

Task:
Solve 3-SUM using the *Quadrithmic*, *Quadratic*, and (bonus point) *quadraticWithCalipers* approaches, as shown in skeleton code in the repository. There are hints at the end of Lesson 2.5 Entropy.
There are also hints in the comments of the existing code. There are a number of unit tests which you should be able to run successfully.
Submit (in your own repository--see instructions elsewhere--include the source code and the unit tests of course):
(a) evidence (screenshot) of your unit tests running (try to show the actual unit test code as well as the green strip);
(b) a spreadsheet showing your timing observations--using the doubling method for at least five values of N--for each of the algorithms (include cubic); Timing should be performed either with an actual stopwatch (e.g. your iPhone) or using the Stopwatch class in the repository.
(c) your brief explanation of why the quadratic method(s) work.

Relationship Conclusion:
The Quadratic method is a more efficient solution for solving the three-sum problem compared to the brute force method, primarily due to its exploitation of sorted array properties to minimize the number of necessary comparisons. In the brute force approach, one must examine every conceivable combination of three numbers in the array, resulting in a time complexity of O(n^3), which becomes impractical for large inputs.

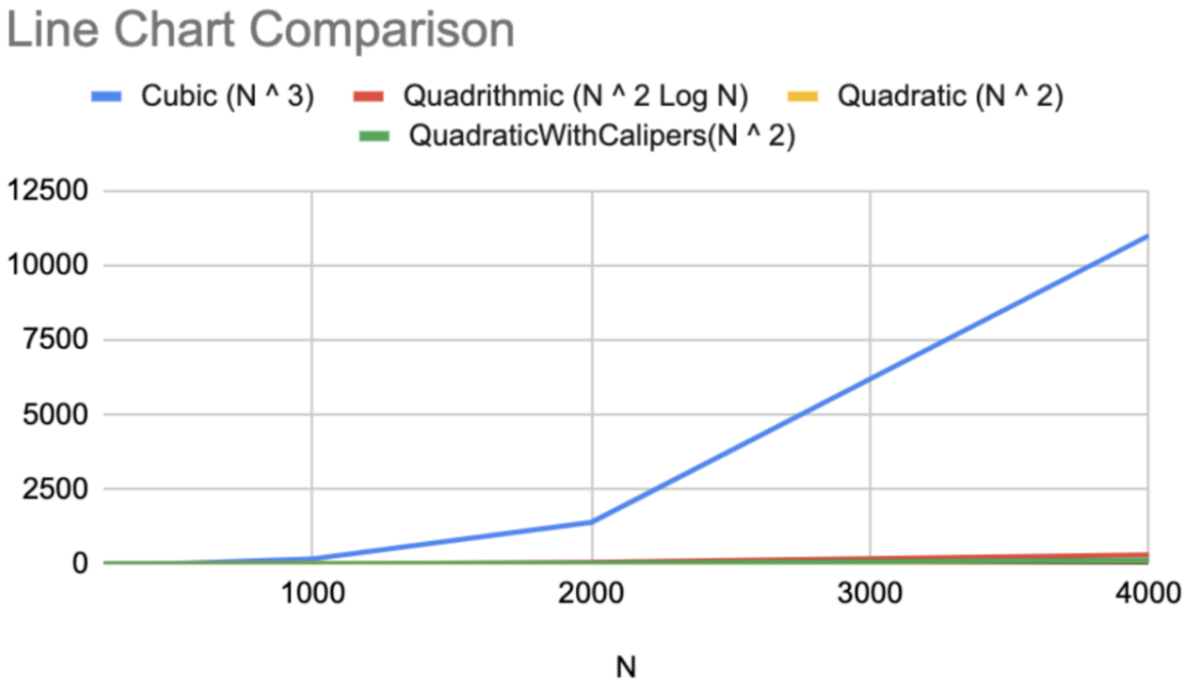
In contrast, the Quadratic method leverages the sorted nature of the input array. It traverses the array while employing two pointers to identify pairs of numbers that sum to the target value, significantly reducing unnecessary comparisons. By iterating through the array just once and utilizing two pointers that converge towards each other, the time complexity is reduced to O(n^2).

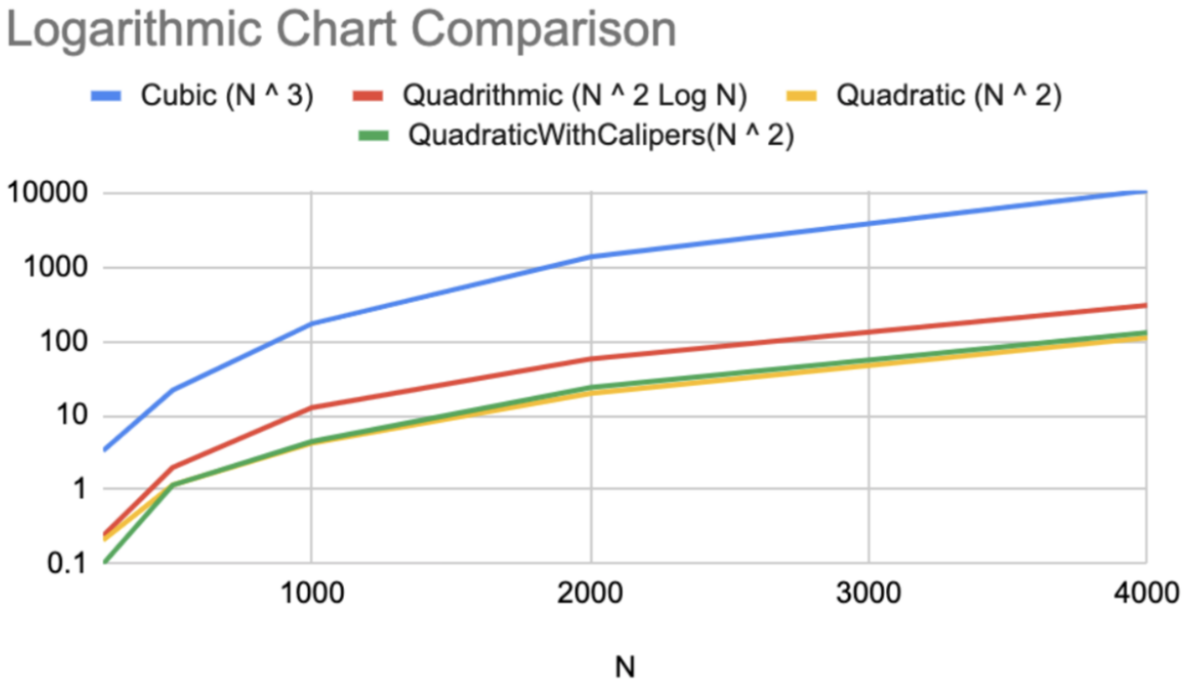
Moreover, the two-pointer method enables the rapid dismissal of pairs of numbers that cannot form a valid solution. This occurs as soon as the method detects that their sum exceeds or falls short of the target value, making it more efficient than the cubic method.
Evidence to support that conclusion:



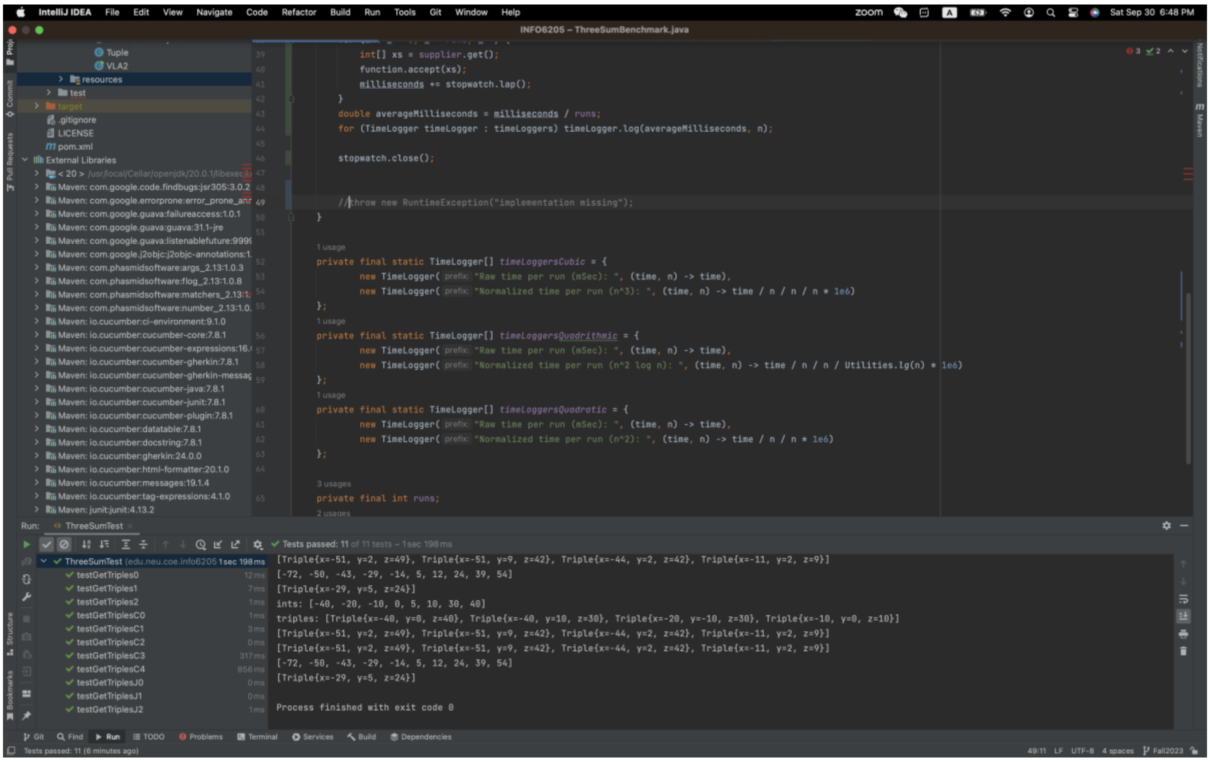
N	Cubic (N ^ 3)	Quadrithmic (N ^ 2 Log N)	Quadratic (N ^ 2)	QuadraticWithCalipers(N ^ 2)
250	3.36	0.24	0.21	0.1
500	22.08	2	1.16	1.16
1000	174.4	12.9	4.3	4.5
2000	1397.6	58.4	20	24.1
4000	11005.6	311	113.2	132.2

Graphical Representation:





Screenshots of run and/or Unit Test:


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
zoom
Sat Sep 30 6:48 PM
INFO6205 – ThreeSumBenchmark.java
Tuple
VLA2
resources
test
target
.gitignore
LICENSE
pom.xml
External Libraries
< 20 > /usr/local/Cellar/openjdk/20.0.1/libexec
Maven: com.google.code.findbugs:jsr305:3.0.2
Maven: com.google.errorprone:error_prone_anr
Maven: com.google.guava:failureaccess:1.0.1
Maven: com.google.guava:guava:31.1-jre
Maven: com.google.guava:listenablefuture:9999
Maven: com.google.j2objc:j2objc-annotations:1.
Maven: com.phasmidsoftware:args_2.13:1.0.3
Maven: com.phasmidsoftware:flog_2.13:1.0.8
Maven: com.phasmidsoftware:matchers_2.13:1.
Maven: com.phasmidsoftware:number_2.13:1.0.
Maven: io.cucumber:ci-environment:9.1.0
Maven: io.cucumber:cucumber-core:7.8.1
Maven: io.cucumber:cucumber-expressions:16.
Maven: io.cucumber:cucumber-gherkin:7.8.1
Maven: io.cucumber:cucumber-gherkin-messag
Maven: io.cucumber:cucumber-java:7.8.1
Maven: io.cucumber:cucumber-junit:7.8.1
Maven: io.cucumber:cucumber-plugin:7.8.1
Maven: io.cucumber:datatable:7.8.1
Maven: io.cucumber:docstring:7.8.1
Maven: io.cucumber:gherkin:24.0.0
Maven: io.cucumber:html-formatter:20.1.0
Maven: io.cucumber:messages:19.1.4
Maven: io.cucumber:tag-expressions:4.1.0
Maven: junit:junit:4.13.2
Project
Commit
Pull Requests
Structure
Bookmarks
Notifications
Maven
Run:
ThreeSumTest
Git
Find
Run
TODO
Problems
Terminal
Services
Build
Dependencies
49:11 LF UTF-8 4 spaces
Fall2023

```java
            int[] xs = supplier.get();
            function.accept(xs);
            milliseconds += stopwatch.lap();
        }
        double averageMilliseconds = milliseconds / runs;
        for (TimeLogger timeLogger : timeLoggers) timeLogger.log(averageMilliseconds, n);

        stopwatch.close();


        //throw new RuntimeException("implementation missing");
    }

    1 usage
    private final static TimeLogger[] timeLoggersCubic = {
            new TimeLogger( prefix: "Raw time per run (mSec): ", (time, n) -> time),
            new TimeLogger( prefix: "Normalized time per run (n^3): ", (time, n) -> time / n / n / n * 1e6)
    };
    1 usage
    private final static TimeLogger[] timeLoggersQuadrithmic = {
            new TimeLogger( prefix: "Raw time per run (mSec): ", (time, n) -> time),
            new TimeLogger( prefix: "Normalized time per run (n^2 log n): ", (time, n) -> time / n / n / Utilities.lg(n) * 1e6)
    };
    1 usage
    private final static TimeLogger[] timeLoggersQuadratic = {
            new TimeLogger( prefix: "Raw time per run (mSec): ", (time, n) -> time),
            new TimeLogger( prefix: "Normalized time per run (n^2): ", (time, n) -> time / n / n * 1e6)
    };

    3 usages
    private final int runs;
    2 usages
```

✓ Tests passed: 11 of 11 tests – 1 sec 198 ms

```
✓ ThreeSumTest (edu.neu.coe.info6205 1 sec 198 ms
    ✓ testGetTriples0     12 ms
    ✓ testGetTriples1      7 ms
    ✓ testGetTriples2      1 ms
    ✓ testGetTriplesC0     1 ms
    ✓ testGetTriplesC1     3 ms
    ✓ testGetTriplesC2     0 ms
    ✓ testGetTriplesC3   317 ms
    ✓ testGetTriplesC4   856 ms
    ✓ testGetTriplesJ0     0 ms
    ✓ testGetTriplesJ1     0 ms
    ✓ testGetTriplesJ2     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}]
[-72, -50, -43, -29, -14, 5, 12, 24, 39, 54]
[Triple{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=-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
```

Tests passed: 11 (6 minutes ago)