

Program Structures and Algorithms

Spring 2023(Section - 1)

Name: Mani Charan Reddy Loka

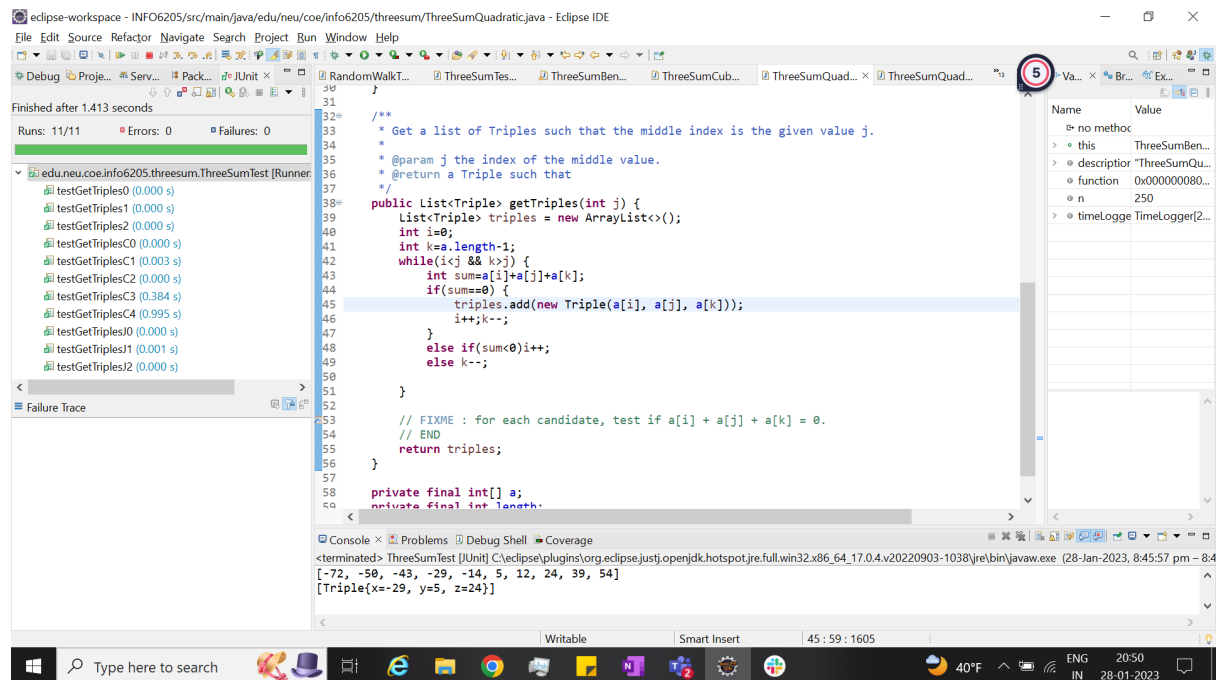
NUID: 002727403

Task:

Solve 3-SUM using the *Quadrithmic*, *Quadratic*, and (bonus point) *quadraticWithCalipers* approaches, as shown in skeleton code in the repository.

Code to solve 3-SUM using Quadratic approach:

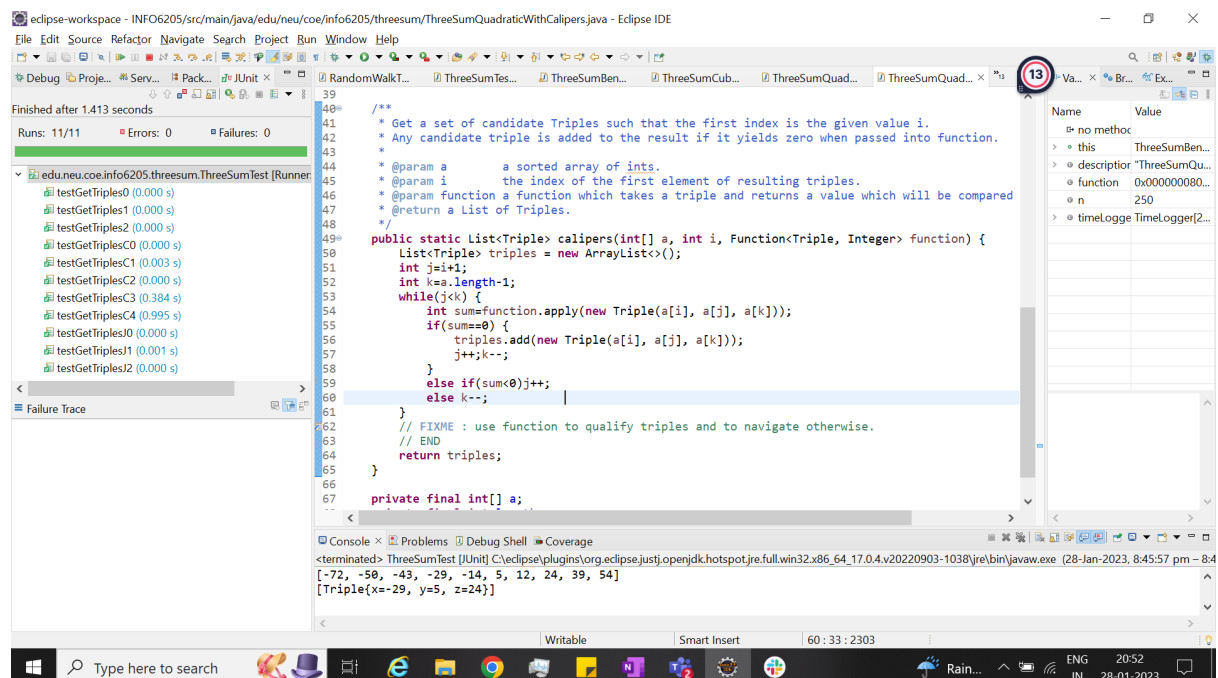
3Sum Quadratic



```
/**
 * Get a list of Triples such that the middle index is the given value j.
 *
 * @param j the index of the middle value.
 * @return a Triple such that
 */
public List<Triple> getTriples(int j) {
    List<Triple> triples = new ArrayList<>();
    int i=0;
    int k=a.length-1;
    while(i<j && k>j) {
        int sum=a[i]+a[j]+a[k];
        if(sum==0) {
            triples.add(new Triple(a[i], a[j], a[k]));
            i++;k--;
        }
        else if(sum<0)i++;
        else k--;
    }
    // FIXME : for each candidate, test if a[i] + a[j] + a[k] = 0.
    // END
    return triples;
}

private final int[] a;
private final int length;
```

3 Sum Quadratic with Calipers



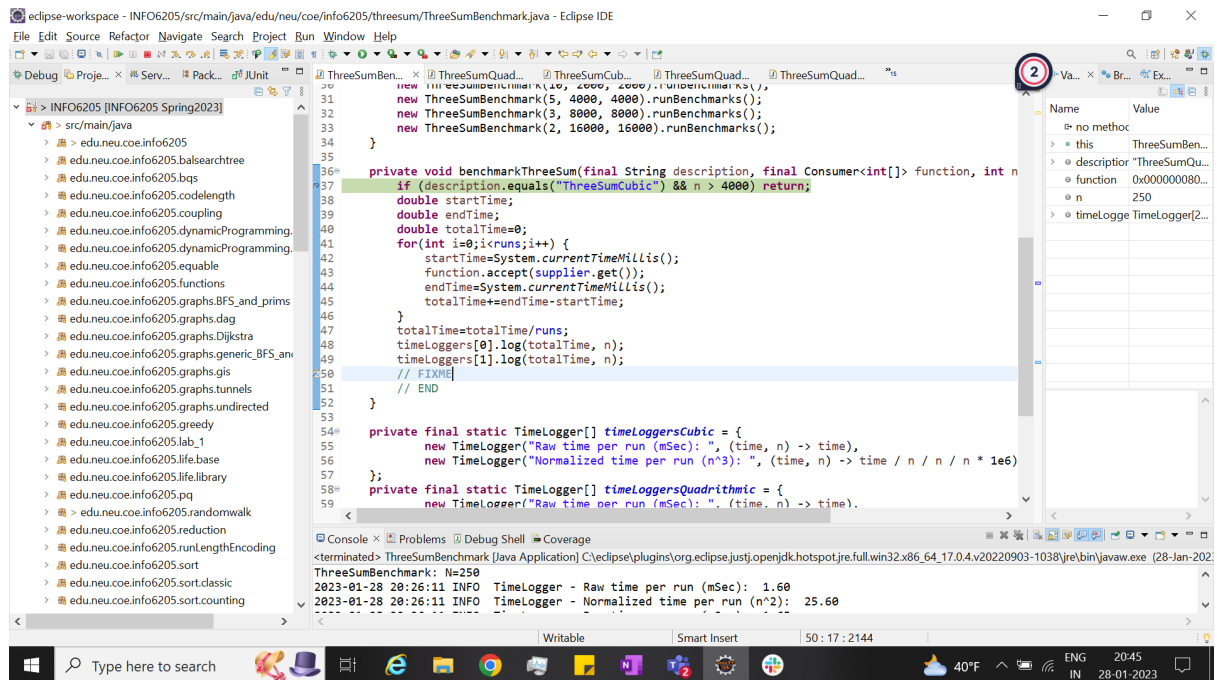
```
/**
 * Get a set of candidate Triples such that the first index is the given value i.
 * Any candidate triple is added to the result if it yields zero when passed into function.
 *
 * @param a a sorted array of ints.
 * @param i the index of the first element of resulting triples.
 * @param function a function which takes a triple and returns a value which will be compared
 * @return a List of Triples.
 */
public static List<Triple> calipers(int[] a, int i, Function<Triple, Integer> function) {
    List<Triple> triples = new ArrayList<>();
    int j=i+1;
    int k=a.length-1;
    while(j<k) {
        int sum=function.apply(new Triple(a[i], a[j], a[k]));
        if(sum==0) {
            triples.add(new Triple(a[i], a[j], a[k]));
            j++;k--;
        }
        else if(sum<0)j++;
        else k--;
    }
    // FIXME : use function to qualify triples and to navigate otherwise.
    // END
    return triples;
}

private final int[] a;
```

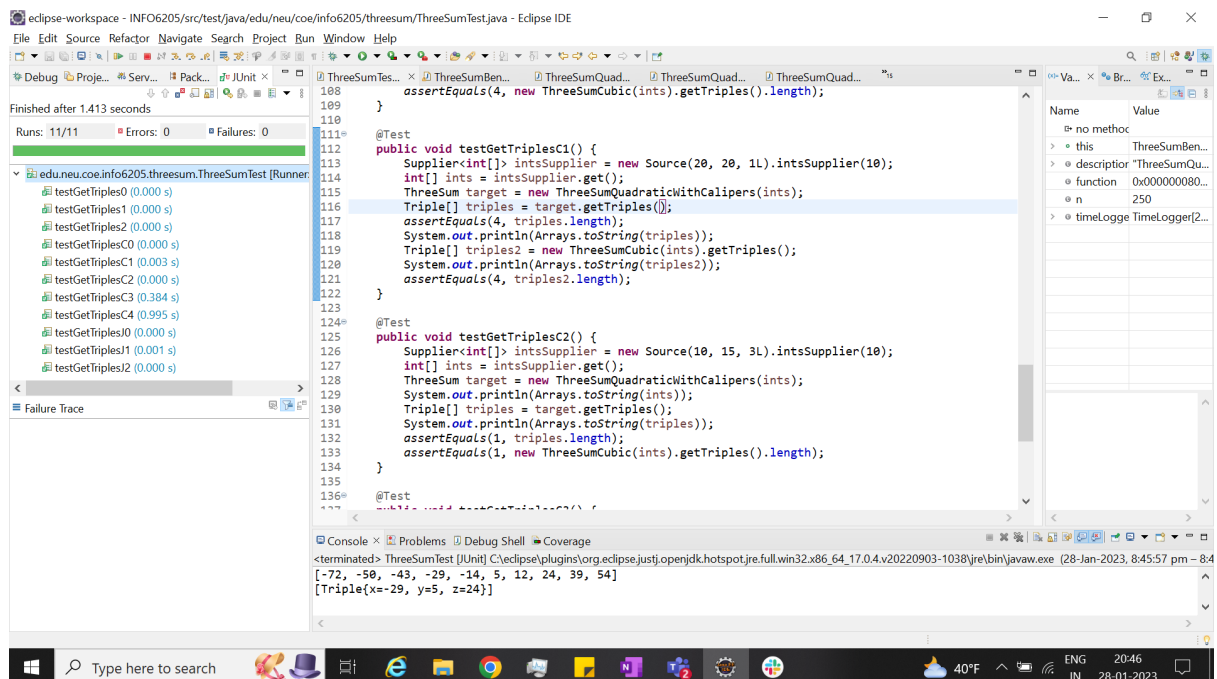
Program Structures and Algorithms

Spring 2023(Section - 1)

3 Sum Benchmark:



(a)Unit Test Screenshot:



Program Structures and Algorithms
Spring 2023(Section - 1)

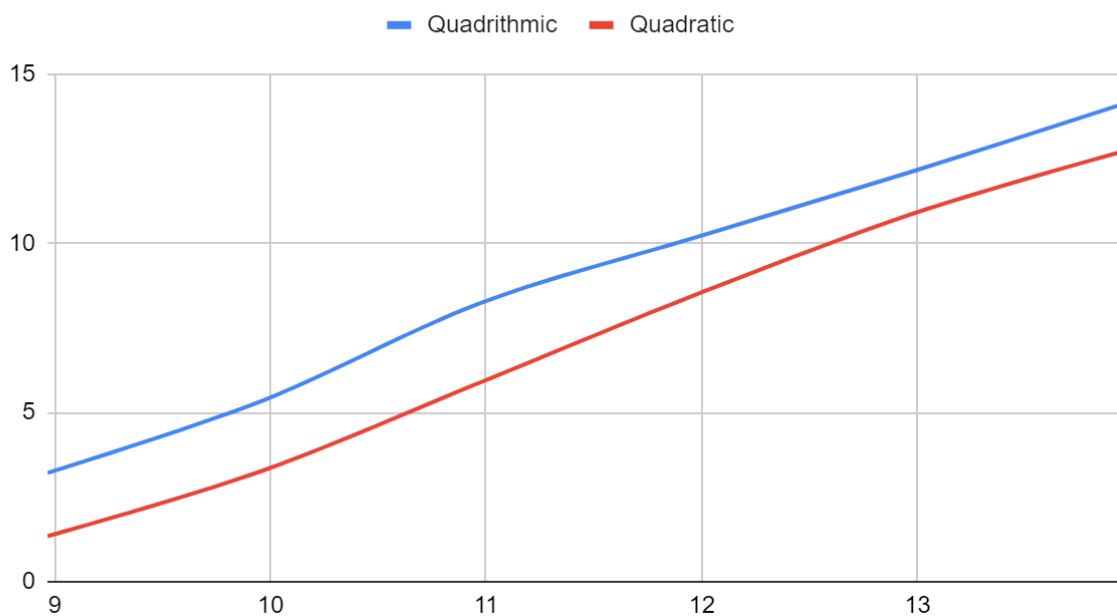
(b)Validating by using doubling benchmarkGraphical Representation:

Here, I used 3Sum Benchmark to time the observations and used the doubling method from $n=250$ until $n=16000$

| N | | Quadratic | | Quadrithmic | | Cubic | |
|-------|------------|------------|----------|-------------|-----------|-----------|----------|
| | | (millisec) | lg ratio | (millisec) | lg ration | (seconds) | lg ratio |
| 250 | Raw Time | 1.6 | | 2.86 | | 0.01547 | |
| | Normalized | 25.6 | | 5.74 | | 0.00099 | |
| 500 | Raw Time | 2.54 | 0.67 | 9.3 | 1.7 | 0.1037 | 2.74 |
| | Normalized | 10.16 | | 4.15 | | 0.00083 | |
| 1000 | Raw Time | 9.75 | 1.94 | 41 | 2.14 | 0.8893 | 3.1 |
| | Normalized | 9.75 | | 4.11 | | 0.00089 | |
| 2000 | Raw Time | 58.5 | 2.58 | 297 | 2.86 | 11.9446 | 3.75 |
| | Normalized | 14.63 | | 6.77 | | 0.00149 | |
| 4000 | Raw Time | 353.4 | 2.59 | 1147.4 | 1.95 | 70.6222 | 2.56 |
| | Normalized | 22.09 | | 5.99 | | 0.0011 | |
| 8000 | Raw Time | 1841.33 | 2.38 | 4388.33 | 1.94 | | |
| | Normalized | 28.77 | | 5.29 | | | |
| 16000 | Raw Time | 6824 | 1.89 | 18053 | 2.04 | | |
| | Normalized | 26.66 | | 5.05 | | | |

Here's the plot of $\log(N)$ vs $\log(\text{Time})$ for Quadratic and Quadrithmic approaches

$\log(\text{Time})$ vs $\log(N)$ for different approaches



(c)Explanation:

The optimization of Quadratic over Quadrithmic approach is that instead of running through 2 for loops on a sorted array for finding all possible pairs of sums of 2 numbers and then trying to do a binary search to find the complement of the sum, we can simply fix one value and iterate left pointer from one step ahead and iterate right pointer from the end and just simply check whether the sum is zero or less than zero or greater than zero and depending on that move the appropriate pointers by increasing the left or decreasing the right. This way, we can improve the time complexity by looping through the array only twice and save the time required to do a binary search.