**Assignment 2: Three Sum**

Code Changes:

**ThreeSumQuadratic.java**

**public** List<Triple> getTriples(**int** j) {  
 List<Triple> triples = **new** ArrayList<>();  
 *//* ***FIXME : for each candidate, test if [i] + a[j] a+ a[k] = 0.* for**(**int** i =0; i < **a**.**length**; i++){  
 **int** l = i+1;  
 **int** r= **a**.**length**-1;  
 **while**(l<r){  
 **int** sum = **a**[i]+**a**[r]+**a**[j];  
 **if**(sum==0 && (l==j)){  
 triples.add(**new** Triple(**a**[i],**a**[j],**a**[r]));  
 l++;  
 r--;  
 } **else if** ( sum > 0) {  
 r--;  
 }**else** {  
 l++;  
 }  
 }  
 }  
 **return** triples;  
}

**ThreeSumQuadraticWithCalipers.java**

**public static** List<Triple> calipers(**int**[] a, **int** i, Function<Triple, Integer> function) {  
 List<Triple> triples = **new** ArrayList<>();  
 *//* ***FIXME : use function to qualify triples and to navigate otherwise.*** *// END* **for**(**int** j=0;j<a.**length**; j++){  
  
 **int** l=i+1;*//using i only for add so always* **int** r=a.**length**-1;  
  
 **while**(l<r){  
 **int** sum = a[i]+a[l]+a[r];  
  
 **if**(sum == 0){  
 triples.add(**new** Triple(a[i],a[l],a[r]));  
 l++;  
 r--;  
 } **else if** (sum >0) {  
 r--;  
 }**else** {  
 l++;  
 }  
 }  
 }  
 **return** triples;  
}

Rest CodeChanges in Repo

**Graphs:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | ThreeSumQuadratic  (ms) | ThreeSumQuadraticWithCalipers  (ms) | ThreeSumCubic  (ms) | ThreeSumQuadrathmic  (ms) |
| 250 | 29.18 | 0.037 | 9.04 | 2.34 |
| 500 | 177.39 | 0.043 | 50.50 | 6.69 |
| 1000 | 1427.06 | 0.19 | 359.52 | 32.70 |
| 2000 | 9968.30 | 0.75 | 2805.88 | 144.13 |
| 4000 | 81678 | 3.20 | 23277 | 576.58 |

ThreeSumQuadratic:

Chart, line chart

Description automatically generated

Three SumCubic:

Chart, line chart

Description automatically generated

ThreeSumQuadraticWithCalipers:

Chart, line chart

Description automatically generated

ThreeSumQuadrithmic:

Chart, line chart

Description automatically generated

Unit Test Cases:

Text

Description automatically generated

Reason why this works:

Implementation of ThreeSumQuadratic which follows the approach of dividing the solution-space into N subspaces.We use 2 pointers in this.

Wherein we are passing an index of middle element of an array.

Here each subspaces points to a fixed value for the middle index of the array.

Then Each subspace is solved by expanding the scope of the other 2 indices outwards from starting point.

As we are using 2-for loops and each subspace can be solved in O(N)time the overall complexity is O(N^2)