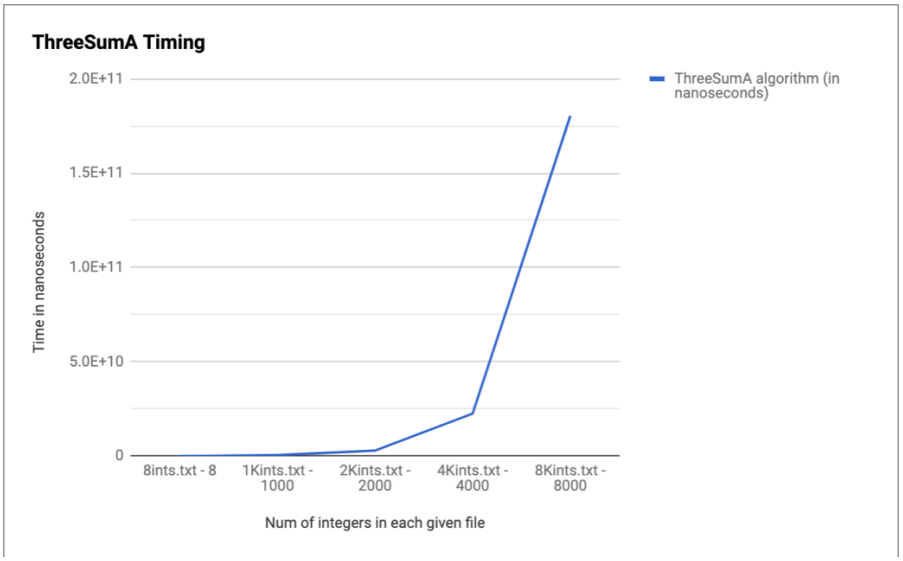
ThreeSumA Algorithm

```
public static int count(int[] a) {
    int n = a.length;
    int count = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i+1; j < n; j++) {
            for (int k = j+1; k < n; k++) {
                if (a[i] + a[j] + a[k] == 0) {
                    count++;
                }
            }
        }
    }
    return count;
}
```

We can see this algorithm has order of growth of N^3
This is because it uses a triply nested loop and takes a proportion of N in each nested loop.

Inputted File	ThreeSumA algorithm (in nanoseconds)
8ints.txt - 8	44968.00
1Kints.txt - 1000	496604579.00
2Kints.txt - 2000	2884082319.00
4Kints.txt - 4000	22508802150.00
8Kints.txt - 8000	180382506884.00

These are the timings I got for each inputted file of integers in nanoseconds. I chose to measure in nanoseconds as I provides a more precise measurement of time. Below is a graphical representation of the above table of results.



While I couldn't get a timing result for '16Kints.txt' and '32Kints.txt', we can still see with the results we do have that this algorithm had begun to form a exponential growth in time meaning that this algorithm is only good for smaller sets of integers.

ThreeSumB Algorithm

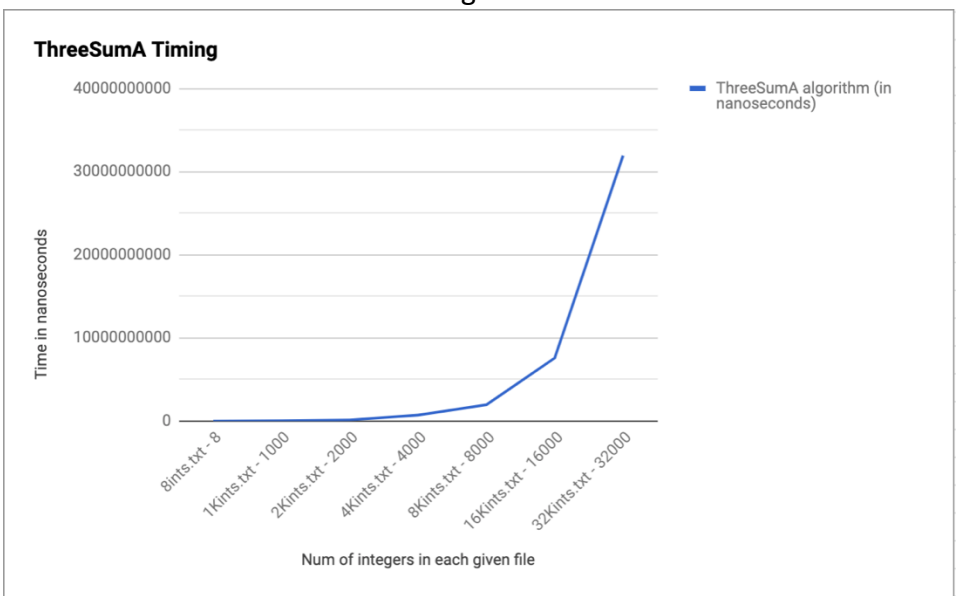
```
public static int count(int[] a) {
    int n = a.length;
    Arrays.sort(a);
    if (containsDuplicates(a)) throw new IllegalArgumentException("array contains duplicate integers");
    int count = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i+1; j < n; j++) {
            int k = Arrays.binarySearch(a, -(a[i] + a[j]));
            if (k > j) count++;
        }
    }
    return count;
}
```

This algorithm's order of growth is $N^2 \log N$ as it uses a nested loop which takes a proportion of N from the prior loop.

Table of timing result's

Inputted File	ThreeSumA algorithm (in nanoseconds)
8ints.txt - 8	305639.00
1Kints.txt - 1000	38964314.00
2Kints.txt - 2000	143120050.00
4Kints.txt - 4000	716968634.00
8Kints.txt - 8000	1962967820.00
16Kints.txt - 16000	7583400856.00
32Kints.txt - 32000	31919250863.00

Unlike the table from ThreeSumA, this one contains results for the input files '16Kints.txt' and '32Kints.txt'. This is due the fact that these two files took too long for a time to be given and as such I couldn't get a result for the table.



While this algorithm eventually still results in an exponential growth in elapsed time, the growth didn't occur until a 32000 integers had been passed into it unlike ThreeSumA which had its growth occur at 8000 integers and then took too long to continue with the remaining sets of integers.