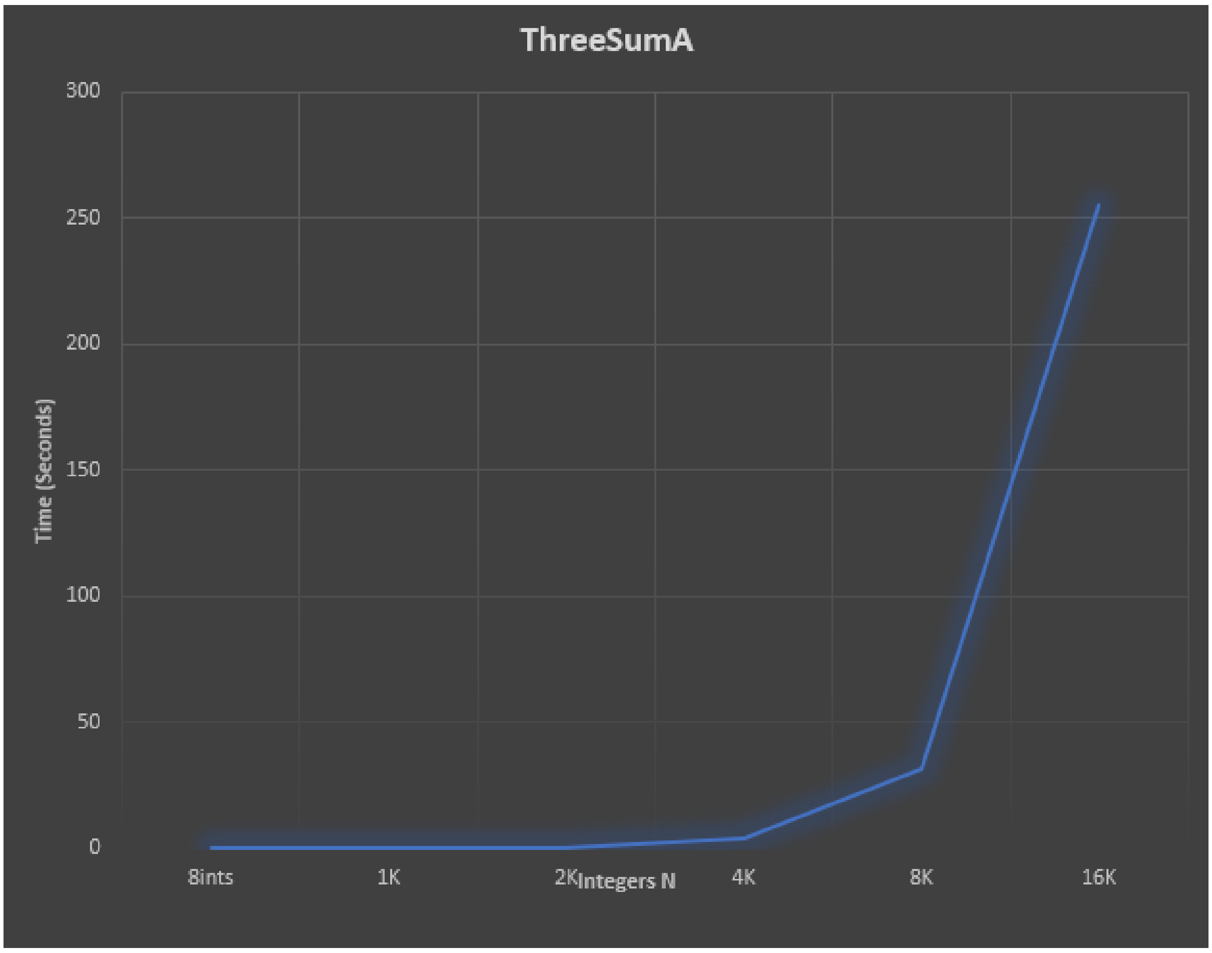
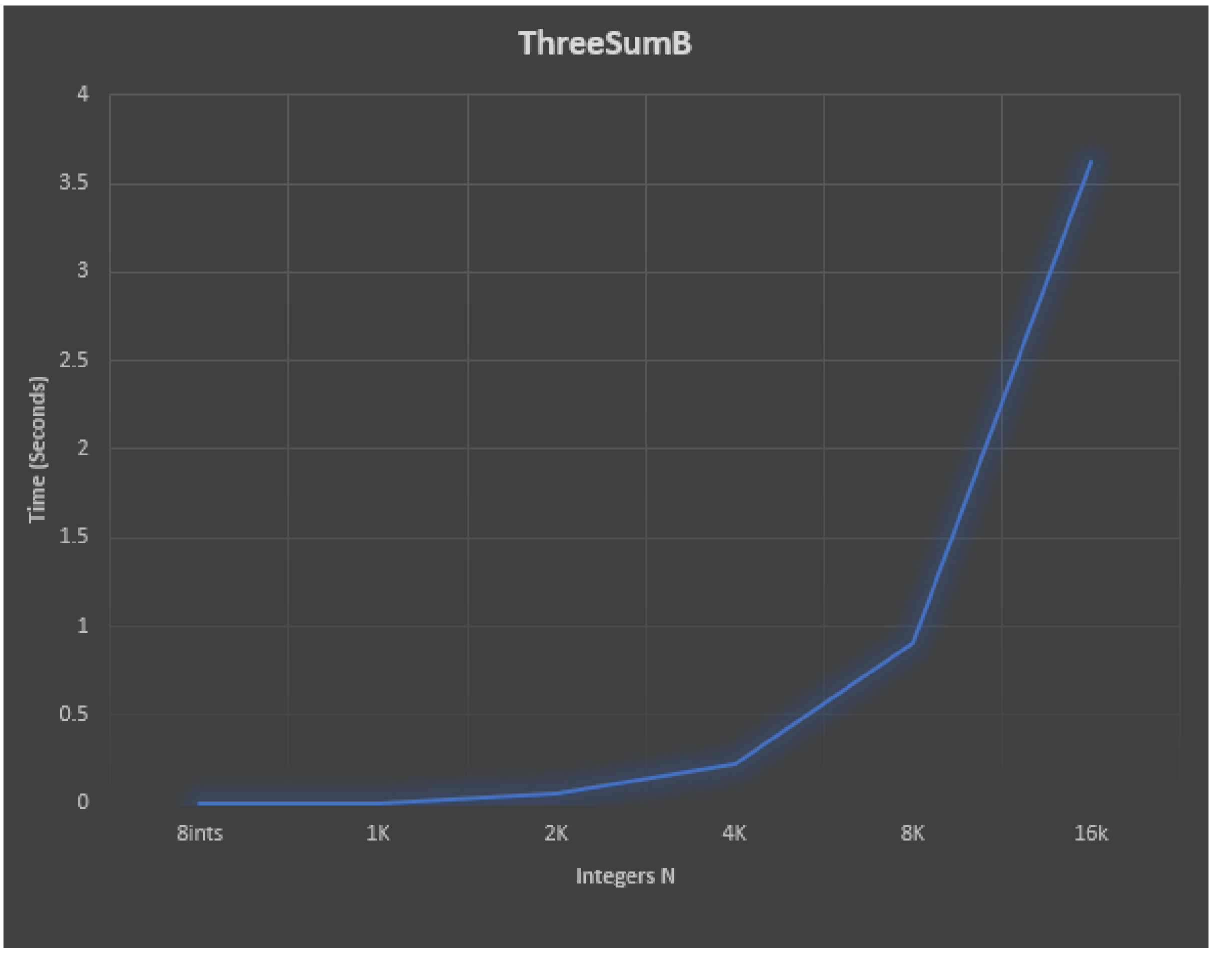
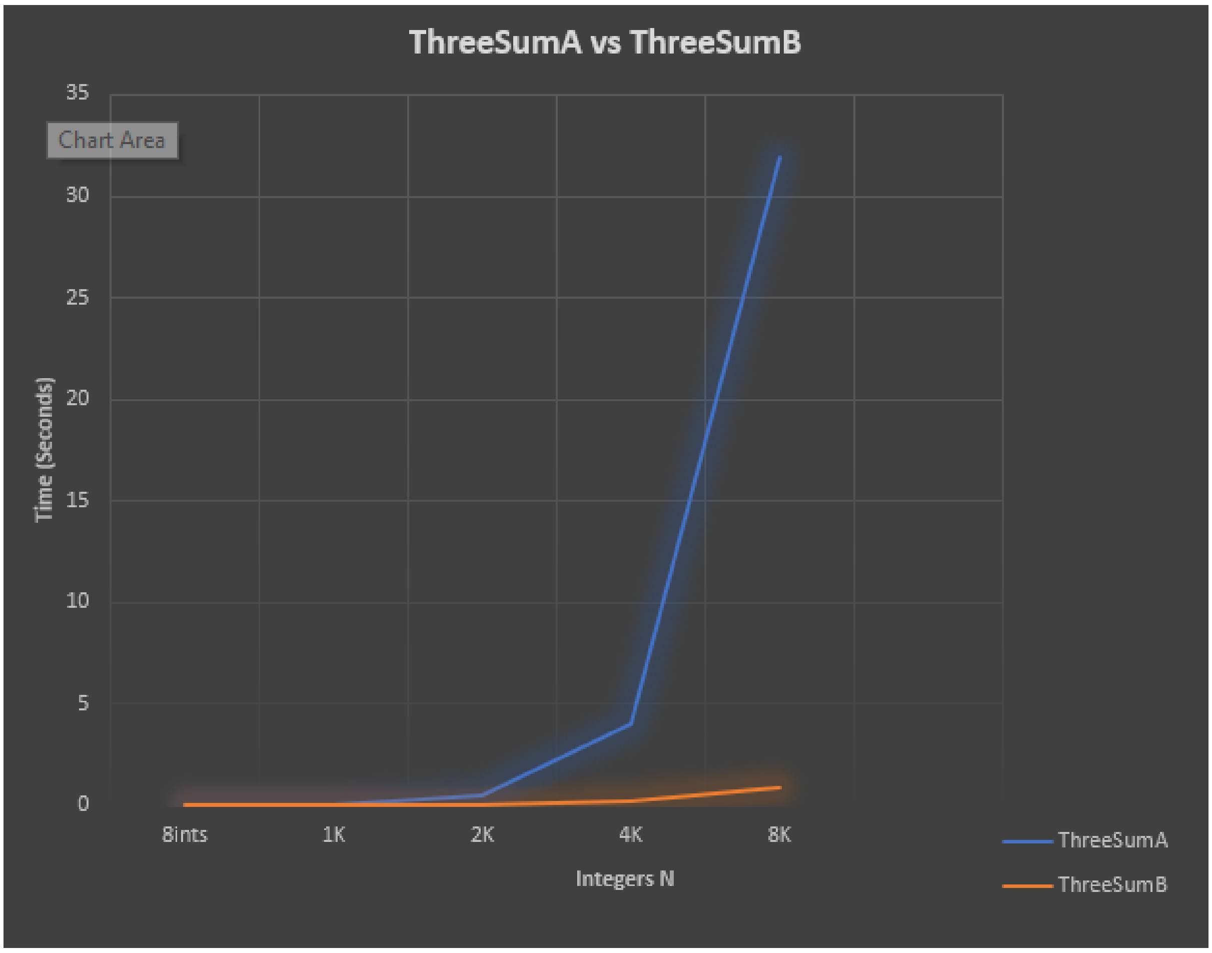
Assignment 1

Algorithm	Input	Time	Number of triples?
ThreeSumA	8ints.txt	0.0	4
~~	1Kints.txt	0.074	70
~~	2Kints.txt	0.524	528
~~	4Kints.txt	4.034	4039
~~	8Kints.txt	31.921	32074
~~	16Kints.txt	255.825	255181



Algorithm	Input	Time	Number of triples?
ThreeSumB	8ints.txt	0.001	4
~~	1Kints.txt	0.017	70
~~	2Kints.txt	0.061	528
~~	4Kints.txt	0.218	4039
~~	8Kints.txt	0.901	32074
~~	16Kints.txt	3.616	255181





1. As seen from above we can tell that ThreeSumB runs much more efficiently than ThreeSumA, if we add the values for 16k ints to this graph, ThreeSumB will only be a line as ThreeSumB grows so much slower than ThreeSumA

2. We can tell that for our ThreeSumA the program has a time complexity of O(n^3) as it is a more primitive implementation of ThreeSum which requires 3 for loops hence O(n^3).
   However ThreeSumB implements the algorithm first using the built in arrays.sort() method which uses a version of timSort (A sorting algorithm based on both insertion sort and MergeSort) which has a time complexity O(nlogn). The algorithm then finds the first 2 elements then does a binary search (O(logN)) to find the element in the sorted array which is equal to the negative of the sum of the first 2 integers. If it finds this val then count is incremented. We can see that we have a time complexity of O(n^2) due to the first 2 loops.