Matthew Soicher
z5160737

Question 1

(a)
Say our array is now A: $\{1, 2, 3, 4\}$ we create $\frac{n(n-1)}{2}$ pairs like so:

$(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)$ as an example. This is done because a pair $(1,2)$ is the same as $(2,1)$. We simultaneously compute the sum of the squares of each pair and store them into an array, B.

Then, we sort B using merge sort $(O(n \log n)$, and then inspect each element and see if it has a duplicate (if it exists, it would be adjacent as the array is sorted). Thus $O(n)$.

At worst case (if there are no two different, distinct pairs of elements to be found), the time complexity of the search is $O(n^2 \log n)$.

(b)

We use a Hash Table to store the sums of the pairs.
Run a nested for loop to run through the values in our given array, which runs in $O(n^2)$ time.

As we sum the results of each pair, we insert the result into the map, and while we do so, check if the Hash Table already contains this sum. If so, we return true from the function and thus have found two (different) pairs of distinct numbers in our array.

If not, we assign the sum to our hash table.

Any search in the hash table is $O(1)$ and does not alter our time complexity.

At worst case (if there are no two different, distinct pairs of elements to be found), the time complexity of the algorithm is $O(n^2)$