Question 1

a)

Compute the $A[k^2] + A[m^2]$ for all $1 \leq k < m < n$ and put them into Array B.
$\binom{n}{2} = \frac{n(n-1)}{2}$. Therefore, Array B has $\frac{n(n-1)}{2}$ integers.
Then we sort the array – using merge sort.
The time complexity is $\frac{n(n-1)}{2} * \log\left(\frac{n(n-1)}{2}\right) \rightarrow n^2 * \log(n^2) \rightarrow 2n^2 log(n)$
So the worst case is $O(n^2 \log(n))$

As array B is linear growth, then we only find that the array B exists 2 same integers equal to the number, using brute force to search. In worst case it takes O(n^2).

Hence this algorithm is also $O(n^2 \log(n))$.

b)

Like the part(a), we compute $A[k^2] + A[m^2]$ and put them into array B.
Therefore, array B has $\frac{n(n-1)}{2}$ integers.
But in this question, we use a hash table to check if it has 2 same numbers in the B and equal to the number. Each insertion and lookup 's average time complexity is O(1).

We hash all elements and go through array B. If the number equal to the one of the integers in array B. We only check if it has as least 2 copies of the integer in hash table.

It takes $\frac{n(n-1)}{2} * 1$, the time complexity is O(n^2).