Q1

(a)

First, we use two *for* loops in order to go through $n(n-1)/2$ pairs $(A[k], A[M]), k < m$, and store $sum(A[k]^2, A[M]^2)$ for all $1 \leq k \leq m \leq n$, in a new array B. Eventually, array B has the size of $n(n-1)/2$.
Next, we sort the array B – we can do this in $O(N \log N) N = n(n-1)/2$ which is equal to $O(n^2 \log n)$ in worst case, for example, using Merge Sort. After that, traverse the sorted array and we can determine whether there exists such a number.

(b)

In this case, we take a similar approach as in (a), except using a hash map to check, each insertion and lookup takes $O(1)$ expected time.
Similar to (a), we replace the array B to a hash map where the key is the $sum(A[k]^2, A[M]^2)$ and the corresponding value is the time the sum appears.
After that, we check whether there exists a key with value 2 in $O(1)$, if so this key is the result, otherwise, there exists not such a number.
In conclusion, this algorithm runs in the expected time of $O(n^2)$.