Part A:

Given: An array A of distinct numbers with size of n.

Aim: Determine if there exists two pairs of two numbers that have the same value of a sum of squares of two distinct numbers.

Step 1:

In order to save each sum values of all possible cases that pairs with two distinct values in array A (there are totally n distinct numbers and we want to find two numbers combination so that we will totally have $\frac{n!}{2!(n-2)!}$ cases), then create an Array B that have size $\frac{n(n-1)}{2}$ (deduced by $\frac{n(n-1)(n-2)!}{2(n-2)!}$). This step will have the time complexity of O(n^2).

Step 2:

Applying the Merge Sort algorithm (time complexity of O($n\log(n)$)) to sort the array B. This step will have the time complexity of O($n^2\log(n)$), because the size of array B is n^2, so that the size of array that algorithm has to sort is $\frac{n(n-1)}{2}$, then the time complexity will be O($n^2\log(n)$).

Step 3:

Go through the sorted array B, find two numbers that are equal by just looking besides each of them, If the next number after aim number is greater, then aim number will move to the next number and keep checking next number of aim number until finding the equal number and return true. If none of numbers in array B are equal then the return will be false . This step have the time complexity of O(n), However the size of whole array B is $\frac{n(n-1)}{2}$, so that the time complexity is O(n^2)

Conclusion:

The algorithm will proceed an array with size n and determine if there are equal sum of squares of two distinct numbers.

The final time complexity is 2*O(n^2)+O($n^2\log(n)$), which is O($n^2\log(n)$).

Part B:

Given: An array A of distinct numbers with size of n.

Aim: Determine if there exists two pairs of two numbers that have the same value of a sum of squares of two distinct numbers.

Step 1: Create a hash table base on array A, Then implementing Merge Sort (O($n\log(n)$)), then find then A[max]. Hash table will not bigger than 2*(A[max])^2

Step 2: Calculating the sum square of all combinations in array A and put the result into hash table. If the place in hash table already exists a pair of sum square values then it means $a^2+b^2=c^2+d^2$ ($\forall$ (a,b,c,d is an element of array n and a,b,c,d are not equal to each others)). This step will have time complexity of O(n^2), because the array that will be gone through have size $\frac{n(n-1)}{2}$.

Conclusion:

The hash table can solve the problem in time complexity of O(n^2).