Yuanzhi shang
Z5158334

# Question 1

(a) In the first loop, compute all possible pairs in array A and store in array B. That would be $\frac{n(n-1)}{2}$ combinations and the complexity would be O(n^2).

Pseudo code:

```
1.  // first step
2. for i in array A:
3.    j = i + 1
4.    for j in Array A:
5.       result = (i^2) + (j^2)
6.       store the result into array B    // O(n^2)
```

So far, all the possible combination results are stored in array B. All we need to do is to find two result that have same value. This would indicate that there are two pairs which have equal value of the sum of square.
Therefore, merge sort the array B(O(nlogn)) and then find if the value of element is equal to the value of its adjacent element.
Pseudo code:

```
1. // second step
2. merge_sort(B)   // O(n^2logn) as the input of the array B grow exponentially
3.
4. // thrid step
5. for i in array B:
6.    if B(i) = B(i+1):    // O(n^2)
7.       success
```

Therefore, the total time complexity in worst case would be the part of merge sort O(n^2logn).

(b) To make expected time complexity O(n^2). The computed result could be stored in Hashtable. As its query time would be O(1) if there exist an duplicated number. Thus, the expected time complexity would be the first step O(n^2)