HOMEWORK 1 – Q1

MINGLANG XIE

z5228006

1. You are given an array *A* of *n* distinct integers.

   (a) You have to determine if there exists a number (not necessarily in *A*) which can be written as a sum of squares of two distinct numbers from *A* in two different ways (note: $m^2 + k^2$ and $k^2 + m^2$ counts as a single way) and which runs in time $n^2 log\ n$ in the **worst case** performance. Note that the brute force algorithm would examine all quadruples of elements in *A* and there are $\binom{n}{4} = O(n^4)$ such quadruples. (10 points)

   (b) Solve the same problem but with an algorithm which runs in the **expected time** of $O(n^2)$. (10 points)

Solution:

(a) First, we need to create a new empty array *T* to store the result of the calculation.

Second, for each element $A[i]$ we can calculate its sum of squares with all the other element $A[i+1:]$, and the worst case for this part is $O(n^2)$ ($\binom{n}{2} = \frac{n(n-1)}{2} = O(n^2)$).

Third, we have $\binom{n}{2} = O(n^2)$ such quadruples in array *T*, and we sort the array *T* using Merge Sort which run in $O(n^2 \log n^2)$ time, then we can use binary search to check that the array *T* has the same solution in $O(\log n^2)$ time. There is no special case because all elements in array *A* are different integers.

Hence, we take at most $O(\log n^2)$ time per calculation, in the worst case we must do $O(n^2)$ time calculations, which is giving an $O(n^2 \log n^2)$ algorithm which is equal to $O(n^2 \log n)$ algorithm because $O(n^2 \log n^2) = O(2n^2 \log n) = O(n^2 \log n)$.

	Time
for i in A:	n
for j in $A[i+1:]$:	n^2
calculation = $A[i]^2 + A[j]^2$	n^2
T.append (calculation)	n^2
Merge Sort (A)	$n^2 \log n^2$
for i in T:	
binary search(i)	$n^2 \log n^2$

$$O(n^2 \log n^2) = O(2n^2 \log n) = O(n^2 \log n)$$

(b) First, we sort the array A using Merge Sort which run in $O(n \log n)$ time.

Second, we create a new array T which have $A[-1]^2 + A[-2]^2$ element. Then, for each element $A[i]$ we can calculate its sum of squares with all the other element $A[i+1:]$, and the worst case for this part is $O(n^2)$, and we store $A[i]^2+A[j]^2$ into $t[A[i]^2+A[j]^2]$, and solution is the index. Therefore, when the second time we reach index $A[i]^2+A[j]^2$,it means there exists a number (not necessarily in *A*) which can be written as a sum of squares of two distinct numbers from *A* in two different ways. The algorithm only loops through the array twice, thus, it's a $O(n^2)$ algorithm.

	time
Merge Sort (A)	$n \log n$
L= $[\infty] * (A[-1]^2+A[-2]^2)$	
for i in A:	n
for j in A:	n^2
calculation = $A[i]^2+A[j]^2$	n^2
if $L[\text{calculation}]$ == calculation:	n^2
return true	
$L[\text{calculation}] = \text{calculation}$	n^2
	$O(n^2)$