Yuan Gao z5239220 Q1

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: $m^2 + n^2$ and $n^2 + m^2$ counts as a single way) and which runs in time $n^2 logn$ 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.

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

a)

Step1: We use $\binom{n}{2}$ to pick two elements(A[i], A[j]) form A and append the squares of two distinct numbers $sqSum = A[i]^2 + A[j]^2$ in squareSum[] array, which costs $\frac{n(n-1)}{2} = O(n^2)$

Step2: Sort squareSum[] array, which costs $O(n^2 logn)$

Step3: Use binary search to find whether there are two equal numbers in squareSum[] array,which costs $O(n^2 logn)$

Therefore, the total time complexity is $O(n^2 logn)$

There is cpp code:

```cpp
/*This is the binary search fuction*/
int binSearch(vector<int> num, int left, int right, int key){
    if(left > right) return -1;
    int mid = left + (right - left) / 2;
    if(num[mid] == key){
        return mid;
    }
    if(num[mid] > key){
        return binSearch(num, left, mid - 1, key);
    }else{
        return binSearch(num, mid + 1, left, key);
    }
}

/*This is the judgement function to check that
whether there are two equal numbers in squareSum[] array*/
bool judgeSquareSum(vector<int> num){
    /*Set a vextor to contain squares of two distinct numbers*/
    vector<int> squareSum;
    /*Step1: append sqSum in squareSum array*/
    for(int i = 0; i < num.size() - 1; i++){
```

```
        for(int j = i + 1; j < num.size(); j++){
            int sqSum = num[i] * num[i] + num[j] * num[j];
            squareSum.push_back(sqSum);
        }
    }
    /*Step2: Sort squareSum[] array*/
    sort(squareSum.begin(),squareSum.end());
    /*Step3: Use binary search to find whether
    there are two equal numbers in squareSum[] array*/
    for(int i = 0; i < squareSum.size() - 1; i++){
        if(binSearch(squareSum, i + 1, squareSum.size(), squareSum[i]) != -1){
            return true;
        }
    }
    return false;
}
```

b)

We need to construct hashmap to make sure the search time complexity is $O(1)$
We can use the dictionary of python3.
In https://docs.python.org/3.8/library/stdtypes.html#dict (Mapping Types — dict), the document describes the data structure of the dictionary
Therefore, we use python dictionary can reduce time complexity
The time complexity is $O(n^2)$
This is python3 code:

```
def judgeSquareSum_fi(num):
    #Set a dic to contain squares of two distinct numbers
    numDic = {}
    #Return value
    count = bool(0)

    for i in range(0, len(num) - 1):
        for j in range(i, len(num)):
            #Get squares of two distinct numbers sqSum
            sqSum = num[i] * num[i] + num[j] * num[j]
            #Search whether sqSUM already exist in numDic
            #if yes returen true
            #else append it in the numDic
            if str(sqSum) in numDic.keys():
                count = bool(1)
                break
            else:
                numDic[str(sqSum)] = [1]
    return count
```