**Yuan Gao z5239220 Q1**

You are given an array A of n distinct integers.

1. 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: and counts as a single way) and which runs in time in the worst case performance. Note that the brute force algorithm would examine all quadruples of elements in A and there are such quadruples.
2. Solve the same problem but with an algorithm which runs in the expected time of .

Step1: We use to pick two elements(A[i], A[j]) form A and append the squares of two distinct numbers in squareSum[] array, which costs

Step2: Sort squareSum[] array, which costs

Step3: Use binary search to find whether there are two equal numbers in

squareSum[] array,which costs

Therefore, the total time complexity is

There is cpp code:

1. /\*This is the binary search fuction\*/
2. **int** binSearch(vector<**int**> num, **int** left, **int** right, **int** key){
3. **if**(left > right) **return** -1;
4. **int** mid = left + (right - left) / 2;
5. **if**(num[mid] == key){
6. **return** mid;
7. }
8. **if**(num[mid] > key){
9. **return** binSearch(num, left, mid - 1, key);
10. }**else**{
11. **return** binSearch(num, mid + 1, left, key);
12. }
13. }
15. /\*This is the judgement function to check that
16. whether there are two equal numbers in squareSum[] array\*/
17. **bool** judgeSquareSum(vector<**int**> num){
18. /\*Set a vextor to contain squares of two distinct numbers\*/
19. vector<**int**> squareSum;
20. /\*Step1: append sqSum in squareSum array\*/
21. **for**(**int** i = 0; i < num.size() - 1; i++){
22. **for**(**int** j = i + 1; j < num.size(); j++){
23. **int** sqSum = num[i] \* num[i] + num[j] \* num[j];
24. squareSum.push\_back(sqSum);
25. }
26. }
27. /\*Step2: Sort squareSum[] array\*/
28. sort(squareSum.begin(),squareSum.end());
29. /\*Step3: Use binary search to find whether
30. there are two equal numbers in squareSum[] array\*/
31. **for**(**int** i = 0; i < squareSum.size() - 1; i++){
32. **if**(binSearch(squareSum, i + 1, squareSum.size(), squareSum[i]) != -1){
33. **return** **true**;
34. }
35. }
36. **return** **false**;
37. }

We need to construct hashmap to make sure the search time complexity is

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

This is python3 code:

1. **def** judgeSquareSum\_fi(num):
2. #Set a dic to contain squares of two distinct numbers
3. numDic = {}
4. #Return value
5. count = bool(0)
7. **for** i **in** range(0, len(num) - 1):
8. **for** j **in** range(i, len(num)):
9. #Get squares of two distinct numbers sqSum
10. sqSum = num[i] \* num[i] + num[j] \* num[j]
11. #Search whether sqSUM already exist in numDic
12. #if yes returen true
13. #else append it in the numDic
14. **if** str(sqSum) **in** numDic.keys():
15. count = bool(1)
16. **break**
17. **else**:
18. numDic[str(sqSum)] = [1]
19. **return** count