532 K-diff Pairs in an Array

2018年4月13日 21:22

Given an array of integers and an integer **k**, you need to find the number of **unique** k-diff pairs in the array. Here a **k-diff** pair is defined as an integer pair (i, j), where **i** and **j** are both numbers in the array and their absolute difference is **k**.

Example 1:
Input: [3, 1, 4, 1, 5], k = 2
Output: 2
Explanation: There are two 2-diff pairs in the array, (1, 3) and (3, 5).
Although we have two 1s in the input, we should only return the number of **unique** pairs.
Example 2:
Input:[1, 2, 3, 4, 5], k = 1
Output: 4
Explanation: There are four 1-diff pairs in the array, (1, 2), (2, 3), (3, 4) and (4, 5).
Example 3:
Input: [1, 3, 1, 5, 4], k = 0
Output: 1
Explanation: There is one 0-diff pair in the array, (1, 1).
Note:

1. The pairs (i, j) and (j, i) count as the same pair.
2. The length of the array won't exceed 10,000.
3. All the integers in the given input belong to the range: [-1e7, 1e7].

来自 <https://leetcode.com/problems/k-diff-pairs-in-an-array/description/>

给定一个整数数组和一个整数 **k**, 你需要在数组里找到**不同的** k-diff 数对。这里将 **k-diff** 数对定义为一个整数对 (i, j), 其中 **i** 和 **j** 都是数组中的数字，且两数之差的绝对值是 **k**.
注意:

1. 数对 (i, j) 和数对 (j, i) 被算作同一数对。
2. 数组的长度不超过10,000。
3. 所有输入的整数的范围在 [-1e7, 1e7]。

Solution for Python3:

```python
class Solution1:
    def findPairs(self, nums, k):
        """
        :type nums: List[int]
        :type k: int
        :rtype: int
        """
        nums.sort()
        res = 0
        for i in range(0, len(nums)):
           if i > 0 and nums[i] == nums[i - 1]:
               continue
           for j in range(i + 1, len(nums)):
               t = nums[j] - k
               if t > nums[i]:
                   break
               if t == nums[i]:
                   res += 1
                   break
               if t < nums[i]:
                   continue
        return res

class Solution2:
    def findPairs(self, nums, k):
        """
        :type nums: List[int]
        :type k: int
        :rtype: int
        """
        return len(set(nums) & {n+k for n in nums}) if k > 0 else sum(v > 1 for v in collections.Counter(nums).values()) if k == 0 else 0
```

Solution for C++:

```cpp
class Solution {
public:
    int findPairs(vector<int>& nums, int k) {
        if (nums.empty() || nums.size() == 0 || k < 0)
            return 0;
        map<int, int> map;
        int cnt = 0;
        for (int i : nums)
            map[i] += 1;
        for (auto it = map.begin(); it != map.end(); it++) {
            if (k == 0) {
                if (it->second >= 2)
                    cnt++;
            } else {
                if (map.count(it->first + k))
                    cnt++;
            }
        }
        return cnt;
    }
};
```