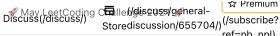
(/) Explore Problems(/problemset/all/)

Interview Contest







User Accepted:

Total Accepted:

Total Submissions:

User Tried:

Difficulty:





0

0

0

0

Medium

5761. Finding Pairs With a Certain Sum

My Submissions (/contest/weekly-contest-241/problems/finding-pairs-with-a-certain-sum/submissions/)

Back to Contest (/contest/weekly-contest-241/)

You are given two integer arrays nums1 and nums2. You are tasked to implement a data structure that supports queries of two types:

- 1. Add a positive integer to an element of a given index in the array nums2.
- 2. Count the number of pairs (i, j) such that nums1[i] + nums2[j] equals a given value (0) <= i < nums1.length and 0 <= j < nums2.length).

Implement the FindSumPairs class:

- FindSumPairs(int[] nums1, int[] nums2) Initializes the FindSumPairs object with two integer arrays nums1 and nums2.
- void add(int index, int val) Adds val to nums2[index], i.e., apply nums2[index] += val.
- int count(int tot) Returns the number of pairs (i, j) such that nums1[i] + nums2[j] == tot.

Example 1:

```
Input
["FindSumPairs", "count", "add", "count", "count", "add", "add", "count"]
[[[1, 1, 2, 2, 2, 3], [1, 4, 5, 2, 5, 4]], [7], [3, 2], [8], [4], [0, 1], [1, 1], [7]]
Output
[null, 8, null, 2, 1, null, null, 11]
Explanation
FindSumPairs findSumPairs = new FindSumPairs([1, 1, 2, 2, 2, 3], [1, 4, 5, 2, 5, 4]);
findSumPairs.count(7); // return 8; pairs (2,2), (3,2), (4,2), (2,4), (3,4), (4,4) make 2 + 5 and pairs (5,1),
findSumPairs.add(3, 2); // now nums2 = [1,4,5,4,5,4]
findSumPairs.count(8); // return 2; pairs (5,2), (5,4) make 3 + 5
findSumPairs.count(4); // return 1; pair (5,0) makes 3 + 1
findSumPairs.add(0, 1); // now nums2 = [2,4,5,4,5,4]
findSumPairs.add(1, 1); // now nums2 = [2,5,5,4,5,4]
findSumPairs.count(7); // return 11; pairs (2,1), (2,2), (2,4), (3,1), (3,2), (3,4), (4,1), (4,2), (4,4) make
```

Constraints:

```
• 1 <= nums1.length <= 1000
```

- 1 <= nums2.length <= 10⁵
- $1 \le nums1[i] \le 10^9$
- $1 \le nums2[i] \le 10^5$
- 0 <= index < nums2.length
- 1 <= val <= 10⁵
- $1 \le tot \le 10^9$
- At most 1000 calls are made to add and count each.

```
JavaScript
                                                                                                              {f c}
1 - /**
     * @param {number[]} nums1
2
     * @param {number[]} nums2
3
4
     */
5 var FindSumPairs = function(nums1, nums2) {
```

```
6
 7
    };
 8
9 - /**
     * @param {number} index
10
     * @param {number} val
11
12
     * @return {void}
13
14 ▼ FindSumPairs.prototype.add = function(index, val) {
15
16
17
18 🗸 /**
     * @param {number} tot
19
20
     * @return {number}
21
22 ▼ FindSumPairs.prototype.count = function(tot) {
23
24
    };
25
26 ▼ /**
     * Your FindSumPairs object will be instantiated and called as such:
27
     * var obj = new FindSumPairs(nums1, nums2)
28
     * obj.add(index,val)
29
     * var param_2 = obj.count(tot)
30
     */
31
```

☐ Custom Testcase

Use Example Testcases





Copyright © 2021 LeetCode

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

