Qn Link : <https://leetcode.com/problems/4sum-ii/description/>

Given four integer arrays nums1, nums2, nums3, and nums4 all of length n, return the number of tuples (i, j, k, l) such that:

* 0 <= i, j, k, l < n
* nums1[i] + nums2[j] + nums3[k] + nums4[l] == 0

Brute Force :

Iterate over all possibilities of 4 array 🡪 Tc : O ( n ^ 4)

Better Approch :

Store the last array elements in a hashmap and reteive them in O (1) Tc

Optimal approach :

Observation :

* Split the eqn , a[i] + b[j] + c[k] + d[l] = 0
* Change the above eqn in the format that , a[i] + b[j] = - ( c[k] + d[l] )

Step 1 : So we put all the possible sum of LHS along with it count it the map

Step 2: The we run another 2 for loops , to generate the possible combination sum in the C and D array.

Step 3 : Get the count by – ( c[i] + d[i])

Step 4 : Don’t directly put the sum

class Solution {

    public int fourSumCount(int[] nums1, int[] nums2, int[] nums3, int[] nums4) {

        Map<Integer , Integer> map = new HashMap<>();

        int n = nums1.length;

        int sum = 0;

        for(int i = 0 ; i < n ; i++){

            for(int j = 0 ; j < n  ; j++){

                sum = nums1[i] + nums2[j];

                map.put(sum , map.getOrDefault(sum, 0) + 1);

            }

        }

        int count = 0;

        for(int i = 0 ; i < n ; i++){

            for(int j = 0 ; j < n  ; j++){

                sum = nums3[i] + nums4[j];

                count += map.getOrDefault(-sum, 0);

            }

        }

        return count;

    }

}