欢迎大家参与本项目，贡献其他语言版本的代码，拥抱开源，让更多学习算法的小伙伴们收益！

> 需要哈希的地方都能找到map的身影

第454题.四数相加II

https://leetcode-cn.com/problems/4sum-ii/

给定四个包含整数的数组列表 A , B , C , D ,计算有多少个元组 (i, j, k, l) ， 使得 A[i] + B[j] + C[k] + D[l] = 0。 排列数

为了使问题简单化，所有的 A, B, C, D 具有相同的长度 N，且 0 ≤ N ≤ 500 。所有整数的范围在 -2^28 到 2^28 - 1 之间，最终结果不会超过 2^31 - 1 。

例如:

输入: A = [1, 2] B = [-2,-1] C = [-1, 2] D = [0, 2] 输出: 2 **解释:** 两个元组如下:

1. (0, 0, 0, 1) -> A[0] + B[0] + C[0] + D[1] = 1 + (-2) + (-1) + 2 = 0
2. (1, 1, 0, 0) -> A[1] + B[1] + C[0] + D[0] = 2 + (-1) + (-1) + 0 = 0

思路

本题咋眼一看好像和0015.三数之和，0018.四数之和差不多，其实差很多。

本题是使用哈希法的经典题目，而0015.三数之和，0018.四数之和并不合适使用哈希法，因为三数之和和四数之和这两道题目使用哈希法在不超时的情况下做到对结果去重是很困难的，很有多细节需要处理。

而这道题目是四个独立的数组，只要找到A[i] + B[j] + C[k] + D[l] = 0就可以，不用考虑有重复的四个元素相加等于0的情况，所以相对于题目18. 四数之和，题目15.三数之和，还是简单了不少！

如果本题想难度升级：就是给出一个数组（而不是四个数组），在这里找出四个元素相加等于0，答案中不可以包含重复的四元组，大家可以思考一下，后续的文章我也会讲到的。

本题解题步骤：

1. 首先定义 一个unordered_map，key放a和b两数之和，value 放a和b两数之和出现的次数。
2. 遍历大A和大B数组，统计两个数组元素之和，和出现的次数，放到map中。
3. 定义int变量count，用来统计a+b+c+d = 0 出现的次数。
4. 在遍历大C和大D数组，找到如果 0-(c+d) 在map中出现过的话，就用count把map中key对应的value也就是出现次数统计出来。
5. 最后返回统计值 count 就可以了

C++代码:

```
class Solution {
public:
    int fourSumCount(vector<int>& A, vector<int>& B, vector<int>& C, vector<int>& D) {
        unordered_map<int, int> umap; //key:a+b的数值，value:a+b数值出现的次数
        // 遍历大A和大B数组，统计两个数组元素之和，和出现的次数，放到map中
        for (int a : A) {
            for (int b : B) {
                umap[a + b]++;
            }
        }
        int count = 0; // 统计a+b+c+d = 0 出现的次数
        // 在遍历大C和大D数组，找到如果 0-(c+d) 在map中出现过的话，就把map中key对应的value也就是出现次数统计出来。
        for (int c : C) {
            for (int d : D) {
                if (umap.find(0 - (c + d)) != umap.end()) {
                    count += umap[0 - (c + d)];
                }
            }
        }
        return count;
    }
};
```

遍历数组的表达式

umap[key] = value

是出现次数统计出来。

其他语言版本

Java：

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

```
        Map<Integer, Integer> map = new HashMap<>();
        int temp;
        int res = 0;
        //统计两个数组中的元素之和，同时统计出现的次数，放入map
        for (int i : nums1) {
            for (int j : nums2) {
                temp = i + j;
                if (map.containsKey(temp)) {
                    map.put(temp, map.get(temp) + 1);
                } else {
                    map.put(temp, 1);
                }
            }
        }
        //统计剩余的两个元素的和，在map中找是否存在相加为0的情况，同时记录次数
        for (int i : nums3) {
            for (int j : nums4) {
                temp = i + j;
                if (map.containsKey(0 - temp)) {
                    res += map.get(0 - temp);
                }
            }
        }
        return res;
    }
}
```

Python：

```
class Solution(object):
    def fourSumCount(self, nums1, nums2, nums3, nums4):
        """
        :type nums1: List[int]
        :type nums2: List[int]
        :type nums3: List[int]
        :type nums4: List[int]
        :rtype: int
        """
        # use a dict to store the elements in nums1 and nums2 and their sum
        hashmap = dict()
        for n1 in nums1:
            for n2 in nums2:
                if n1 + n2 in hashmap:
                    hashmap[n1+n2] += 1
                else:
                    hashmap[n1+n2] = 1

        # if the -(a+b) exists in nums3 and nums4, we shall add the count
        count = 0
        for n3 in nums3:
            for n4 in nums4:
                key = - n3 - n4
                if key in hashmap:
                    count += hashmap[key]
        return count

# 下面这个写法更为简洁，但是表达的是同样的算法
```

```python
# class Solution:
#     def fourSumCount(self, nums1: List[int], nums2: List[int], nums3: List[int], nums4: L
#         from collections import defaultdict

#         hashmap = defaultdict(int)

#         for x1 in nums1:
#             for x2 in nums2:
#                 hashmap[x1+x2] += 1

#         count=0
#         for x3 in nums3:
#             for x4 in nums4:
#                 key = -x3-x4
#                 value = hashmap.get(key)

                # dict的get方法会返回None（key不存在）或者key对应的value
                # 所以如果value==0，就会继续执行or，count+0，否则就会直接加value
                # 这样就不用去写if判断了

#                 count += value or 0

#         return count
```

Go：

```go
func fourSumCount(nums1 []int, nums2 []int, nums3 []int, nums4 []int) int {
    m := make(map[int]int)
    count := 0
    for _, v1 := range nums1 {
        for _, v2 := range nums2 {
            m[v1+v2]++
        }
    }
    for _, v3 := range nums3 {
        for _, v4 := range nums4 {
            count += m[-v3-v4]
        }
    }
    return count
}
```

javaScript:

```javascript
/**
 * @param {number[]} nums1
 * @param {number[]} nums2
 * @param {number[]} nums3
 * @param {number[]} nums4
 * @return {number}
 */
var fourSumCount = function(nums1, nums2, nums3, nums4) {
    const twoSumMap = new Map();
    let count = 0;
```

```
    for(const n1 of nums1) {
        for(const n2 of nums2) {
            const sum = n1 + n2;
            twoSumMap.set(sum, (twoSumMap.get(sum) || 0) + 1)
        }
    }

    for(const n3 of nums3) {
        for(const n4 of nums4) {
            const sum = n3 + n4;
            count += (twoSumMap.get(0 - sum) || 0)
        }
    }

    return count;
};
```

- 作者微信：程序员Carl
- B站视频：代码随想录
- 知识星球：代码随想录