第454题.四数相加II

力扣题目链接(opens new window)

给你四个整数数组 `nums1`、`nums2`、`nums3` 和 `nums4` ，数组长度都是 `n` ，请你计算有多少个元组 `(i, j, k, l)` 能满足：

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

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

本题乍眼一看好像和0015.三数之和 (opens new window)，0018.四数之和 (opens new window)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 (opens new window)，0018.四数之和 (opens new window)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 **（对应python中的字典dict）**，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. 再遍历大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

(版本一) 使用字典

```
class Solution:
    def fourSumCount(self, nums1: List[int], nums2: List[int], nums3: List[int], nums4: List[int]) -> int:
        result = dict()
        for i in nums1:
            for j in nums2:
                if i+j in result:
                    result[i+j] += 1
                else:
                    result[i+j] = 1

        count = 0
        for k in nums3:
            for l in nums4:
                key = -k-l
                if key in result:
                    count += result[key]

        return count
```

(版本二) 使用字典

```
class Solution(object):
    def fourSumCount(self, nums1, nums2, nums3, nums4):
        # 使用字典存储nums1和nums2中的元素及其和
        hashmap = dict()
        for n1 in nums1:
            for n2 in nums2:
                hashmap[n1+n2] = hashmap.get(n1+n2, 0) + 1

        # 如果 -(n1+n2) 存在于nums3和nums4, 存入结果
        count = 0
        for n3 in nums3:
            for n4 in nums4:
                key = - n3 - n4
                if key in hashmap:
                    count += hashmap[key]
        return count
```

(版本三) 使用 defaultdict

```python
from collections import defaultdict
class Solution:
    def fourSumCount(self, nums1: list, nums2: list, nums3: list, nums4: list) ->
int:
        rec, cnt = defaultdict(lambda : 0), 0
        for i in nums1:
            for j in nums2:
                rec[i+j] += 1
        for i in nums3:
            for j in nums4:
                cnt += rec.get(-(i+j), 0)
        return cnt
```

> **Tip**
>
> **一、`defaultdict` 基础知识**
>
> `defaultdict` 是 Python 标准库 `collections` 模块中的一个字典（`dict`）的子类。它的主要特点是：当你访问一个不存在的键时，它不会抛出 `KeyError`，而是自动创建这个键，并赋予一个默认值。
>
> 1. 创建方式
>
> ```python
> from collections import defaultdict
>
> # 语法：defaultdict(默认值工厂函数)
> d = defaultdict(int)        # 默认值为 0（因为 int() 返回 0）
> d = defaultdict(list)       # 默认值为 []（因为 list() 返回空列表）
> d = defaultdict(set)        # 默认值为 set()
> d = defaultdict(lambda: 0)  # 用 lambda 自定义默认值，等价于 defaultdict(int)
> ```
>
> 2. 与普通 `dict` 的区别
>
> ```python
> d1 = {}
> print(d1['a'])  # KeyError!
>
> d2 = defaultdict(int)
> print(d2['a'])  # 输出 0，不会报错
> ```
>
> 3. 为什么用 `defaultdict(lambda: 0)`？
>
> `lambda: 0` 表示：每次遇到新键，就调用这个函数，返回 `0`。 这等价于 `defaultdict(int)`，因为 `int()` 默认返回 `0`。
> 所以你代码中的 `defaultdict(lambda: 0)` 可以简写为 `defaultdict(int)`，效果完全一样。