1. 两数之和+454. 四数相加 II

Monday, July 18, 2022 9:27 AM

https://leetcode.cn/problems/two-sum/

简单

给定一个整数数组 nums 和一个整数目标值 target，请你在该数组中找出 和为目标值 target 的那 两个 整数，并返回它们的数组下标。

你可以假设每种输入只会对应一个答案。但是，数组中同一个元素在答案里不能重复出现。

你可以按任意顺序返回答案。

示例 1：

输入：nums = [2,7,11,15], target = 9

输出：[0,1]

解释：因为 nums[0] + nums[1] == 9 ，返回 [0, 1] 。

示例 2：

输入：nums = [3,2,4], target = 6

输出：[1,2]

示例 3：

输入：nums = [3,3], target = 6

输出：[0,1]

思路：

解法一、暴力，双for循环

解法二、单for循环遍历数组，target减去当前数组数

在map中时，返回其map下标，将此两个下标对应数存入结果。

若 不在map中，则将当前数及其下标存入map。

```
public int[] twoSum(int[] nums, int target) {
    int[] res = new int[2]; // 题目要求：结果为两个数
    if(nums == null || nums.length == 0){
        return res;
    }
    Map<Integer, Integer> map = new HashMap<>(); // 存每个数及其对应数组下标
    for(int i = 0; i < nums.length; i++){
        int temp = target - nums[i];
        if(map.containsKey(temp)){ //数组中有 则找到了，存入结果
            res[1] = i;
            res[0] = map.get(temp);
        }
        map.put(nums[i], i); // 放进去的是 数和对应下标
    }
    return res;
}
```

454. 四数相加 II

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

中等

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

示例 1：

输入：nums1 = [1,2], nums2 = [-2,-1], nums3 = [-1,2], nums4 = [0,2]

输出：2

解释：

两个元组如下：

1. (0, 0, 0, 1) -> nums1[0] + nums2[0] + nums3[0] + nums4[1] = 1 + (-2) + (-1) + 2 = 0
2. (1, 1, 0, 0) -> nums1[1] + nums2[1] + nums3[0] + nums4[0] = 2 + (-1) + (-1) + 0 = 0

思路：

首先，求的是不同的组合数。不管数是否相同。

然后，我们遍历前两个数组，将其每种组合之和存入map，key为和，value为出现此和的次数。

接着，遍历后两个数组，遍历其每种组合之和，在map中找相加为0的key，返回其value，最终结果加上该value。

code：

```
class Solution {
    public int fourSumCount(int[] nums1, int[] nums2, int[] nums3, int[] nums4) {
        int res=0; //结果有几种
        int temp;
        HashMap<Integer,Integer> map=new HashMap<>();
        for(int i:nums1){ // 存前两个数组 组合之和 出现的次数
            for(int j:nums2){
                temp=i+j;
                if(map.containsKey(temp)){
                    map.put(temp,map.get(temp)+1);
                }else{
                    map.put(temp,1);
                }
            }
        }
        for(int i: nums3){
            for(int j:nums4){
                temp=i+j;
                if(map.containsKey(-temp)){
```

```
                    res+=map.get(-temp);
                }
            }
        }
        return res;
    }
}
```