PDF下载 代码随想录 刷题 微信群 B站 代码随想录 知识星球 代码随想录

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！

> 一样的道理，能解决四数之和 那么五数之和、六数之和、N数之和呢？

第18题. 四数之和

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

题意：给定一个包含 n 个整数的数组 nums 和一个目标值 target，判断 nums 中是否存在四个元素 a，b，c 和 d ，使得 a + b + c + d 的值与 target 相等？找出所有满足条件且不重复的四元组。

注意：

答案中不可以包含重复的四元组。

示例： 给定数组 nums = [1, 0, -1, 0, -2, 2]，和 target = 0。 满足要求的四元组集合为： [[-1, 0, 0, 1], [-2, -1, 1, 2], [-2, 0, 0, 2]]

思路

四数之和，和15.三数之和是一个思路，都是使用双指针法, 基本解法就是在15.三数之和 的基础上再套一层for循环。

但是有一些细节需要注意，例如： 不要判断 `nums[k] > target` 就返回了，三数之和 可以通过 `nums[i] > 0` 就返回了，因为 0 已经是确定的数了，四数之和这道题目 target是任意值。（大家亲自写代码就能感受出来）

15.三数之和的双指针解法是一层for循环num[i]为确定值，然后循环内有left和right下表作为双指针，找到nums[i] + nums[left] + nums[right] == 0。

四数之和的双指针解法是两层for循环nums[k] + nums[i]为确定值，依然是循环内有left和right下表作为双指针，找出nums[k] + nums[i] + nums[left] + nums[right] == target的情况，三数之和的时间复杂度是O(n^2)，四数之和的时间复杂度是O(n^3) 。

那么一样的道理，五数之和、六数之和等等都采用这种解法。

对于15.三数之和双指针法就是将原本暴力O(n^3)的解法，降为O(n^2)的解法，四数之和的双指针解法就是将原本暴力O(n^4)的解法，降为O(n^3)的解法。

之前我们讲过哈希表的经典题目：454.四数相加II，相对于本题简单很多，因为本题是要求在一个集合中找出四个数相加等于target，同时四元组不能重复。

而454.四数相加II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，所以相对于本题还是简单了不少！

我们来回顾一下，几道题目使用了双指针法。

双指针法将时间复杂度O(n^2)的解法优化为 O(n)的解法。也就是降一个数量级，题目如下：

- 27.移除元素
- 15.三数之和
- 18.四数之和

操作链表：

- 206.反转链表
- 19.删除链表的倒数第N个节点
- 面试题 02.07. 链表相交
- 142题.环形链表II

双指针法在字符串题目中还有很多应用，后面还会介绍到。

C++代码

```
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        vector<vector<int>> result;
        sort(nums.begin(), nums.end());
        for (int k = 0; k < nums.size(); k++) {
            // 这种剪枝是错误的，这道题目target 是任意值
            // if (nums[k] > target) {
            //     return result;
            // }
            // 去重
            if (k > 0 && nums[k] == nums[k - 1]) {
                continue;
            }
            for (int i = k + 1; i < nums.size(); i++) {
```

```
                // 正确去重方法
                if (i > k + 1 && nums[i] == nums[i - 1]) {
                    continue;
                }
                int left = i + 1;
                int right = nums.size() - 1;
                while (right > left) {
                    if (nums[k] + nums[i] + nums[left] + nums[right] > target) {
                        right--;
                    } else if (nums[k] + nums[i] + nums[left] + nums[right] < target) {
                        left++;
                    } else {
                        result.push_back(vector<int>{nums[k], nums[i], nums[left], nums[rig
                        // 去重逻辑应该放在找到一个四元组之后
                        while (right > left && nums[right] == nums[right - 1]) right--;
                        while (right > left && nums[left] == nums[left + 1]) left++;

                        // 找到答案时，双指针同时收缩
                        right--;
                        left++;
                    }
                }

            }
        }
        return result;
    }

};
```

其他语言版本

Java：

```
class Solution {
    public List<List<Integer>> fourSum(int[] nums, int target) {
        List<List<Integer>> result = new ArrayList<>();
        Arrays.sort(nums);

        for (int i = 0; i < nums.length; i++) {

            if (i > 0 && nums[i - 1] == nums[i]) {
                continue;
            }

            for (int j = i + 1; j < nums.length; j++) {

                if (j > i + 1 && nums[j - 1] == nums[j]) {
                    continue;
                }

                int left = j + 1;
                int right = nums.length - 1;
                while (right > left) {
                    int sum = nums[i] + nums[j] + nums[left] + nums[right];
```

```
                    if (sum > target) {
                        right--;
                    } else if (sum < target) {
                        left++;
                    } else {
                        result.add(Arrays.asList(nums[i], nums[j], nums[left], nums[right]));

                        while (right > left && nums[right] == nums[right - 1]) right--;
                        while (right > left && nums[left] == nums[left + 1]) left++;

                        left++;
                        right--;
                    }
                }
            }
        }
        return result;
    }
}
```

Python:

```
class Solution(object):
    def fourSum(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: List[List[int]]
        """
        # use a dict to store value:showtimes
        hashmap = dict()
        for n in nums:
            if n in hashmap:
                hashmap[n] += 1
            else:
                hashmap[n] = 1

        # good thing about using python is you can use set to drop duplicates.
        ans = set()
        for i in range(len(nums)):
            for j in range(i + 1, len(nums)):
                for k in range(j + 1, len(nums)):
                    val = target - (nums[i] + nums[j] + nums[k])
                    if val in hashmap:
                        # make sure no duplicates.
                        count = (nums[i] == val) + (nums[j] == val) + (nums[k] == val)
                        if hashmap[val] > count:
                            ans.add(tuple(sorted([nums[i], nums[j], nums[k], val])))
                    else:
                        continue
        return ans
```

Go:

javaScript:

```
/**
 * @param {number[]} nums
 * @param {number} target
 * @return {number[][]}
 */
var fourSum = function(nums, target) {
    const len = nums.length;
    if(len < 4) return [];
    nums.sort((a, b) => a - b);
    const res = [];
    for(let i = 0; i < len - 3; i++) {
        // 去重i
        if(i > 0 && nums[i] === nums[i - 1]) continue;
        for(let j = i + 1; j < len - 2; j++) {
            // 去重j
            if(j > i + 1 && nums[j] === nums[j - 1]) continue;
            let l = j + 1, r = len - 1;
            while(l < r) {
                const sum = nums[i] + nums[j] + nums[l] + nums[r];
                if(sum < target) { l++; continue}
                if(sum > target) { r--; continue}
                res.push([nums[i], nums[j], nums[l], nums[r]]);
                while(l < r && nums[l] === nums[++l]);
                while(l < r && nums[r] === nums[--r]);
            }
        }
    }
    return res;
};
```

- 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