

宫水三叶的刷题日记

n 数之和

Author : 宫水三叶

Date : 2021/10/07

QQ Group: 703311589

WeChat : oaoaya

宫水三叶

刷题日记

公众号: 宫水三叶的刷题日记

噔噔噔噔，这是公众号「[宫水三叶的刷题日记](#)」的原创专题「n 数之和」合集。

本合集更新时间为 2021-10-07，大概每 2-4 周会集中更新一次。关注公众号，后台回复「n 数之和」即可获取最新下载链接。

💡下面介绍使用本合集的最佳使用实践：

学习算法：

1. 打开在线目录（[Github 版](#) & [Gitee 版](#)）；
2. 从侧边栏的类别目录找到「n 数之和」；
3. 按照「推荐指数」从大到小进行刷题，「推荐指数」相同，则按照「难度」从易到难进行刷题；
4. 拿到题号之后，回到本合集进行检索。

维持熟练度：

1. 按照本合集「从上往下」进行刷题。

学习过程中遇到任何困难，欢迎加入「每日一题打卡 QQ 群：703311589」进行交流   

题目描述

这是 LeetCode 上的 [15. 三数之和](#)，难度为 中等。

Tag：「双指针」、「排序」、「n 数之和」

给你一个包含 n 个整数的数组 nums，判断 nums 中是否存在三个元素 a，b，c，使得 $a + b + c = 0$ ？

请你找出所有和为 0 且不重复的三元组。

注意：答案中不可以包含重复的三元组。

示例 1：

刷题日记

公众号：宫水三叶的刷题日记

输入：nums = [-1,0,1,2,-1,-4]
输出：[[-1,-1,2],[-1,0,1]]

示例 2：

输入：nums = []
输出：[]

示例 3：

输入：nums = [0]
输出：[]

提示：

- $0 \leq \text{nums.length} \leq 3000$
- $-10^5 \leq \text{nums}[i] \leq 10^5$

排序 + 双指针

对数组进行排序，使用三个指针 `i`、`j` 和 `k` 分别代表要找的三个数。

1. 通过枚举 `i` 确定第一个数，另外两个指针 `j`，`k` 分别从左边 `i + 1` 和右边 `n - 1` 往中间移动，找到满足 `nums[i] + nums[j] + nums[k] == 0` 的所有组合。
2. `j` 和 `k` 指针的移动逻辑，分情况讨论 `sum = nums[i] + nums[j] + nums[k]`：
 - `sum > 0`：`k` 左移，使 `sum` 变小
 - `sum < 0`：`j` 右移，使 `sum` 变大
 - `sum = 0`：找到符合要求的答案，存起来

由于题目要求答案不能包含重复的三元组，所以在确定第一个数和第二个数的时候，要跳过数值一样的下标（在三数之和确定的情况下，确保第一个数和第二个数不会重复，即可保证三元组不重复）。

代码：

刷题日记

公众号：宫水三叶的刷题日记

```

class Solution {
    public List<List<Integer>> threeSum(int[] nums) {
        Arrays.sort(nums);
        int n = nums.length;
        List<List<Integer>> ans = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            if (i > 0 && nums[i] == nums[i - 1]) continue;
            int j = i + 1, k = n - 1;
            while (j < k) {
                while (j > i + 1 && j < n && nums[j] == nums[j - 1]) j++;
                if (j >= k) break;
                int sum = nums[i] + nums[j] + nums[k];
                if (sum == 0) {
                    ans.add(Arrays.asList(nums[i], nums[j], nums[k]));
                    j++;
                } else if (sum > 0) {
                    k--;
                } else if (sum < 0) {
                    j++;
                }
            }
        }
        return ans;
    }
}

```

- 时间复杂度：排序的复杂度为 $O(\log N)$ ，对于每个 i 而言，最坏的情况 j 和 k 都要扫描一遍数组的剩余部分，复杂度为 $O(n^2)$ 。整体复杂度为 $O(n^2)$
- 空间复杂度： $O(n^2)$

**🔍更多精彩内容，欢迎关注：[公众号](#) / [Github](#) / [LeetCode](#) / [知乎](#) **

题目描述

这是 LeetCode 上的 **16. 最接近的三数之和**，难度为 **中等**。

Tag：「双指针」、「排序」、「n 数之和」

给定一个包括 n 个整数的数组 $nums$ 和一个目标值 $target$ 。

找出 $nums$ 中的三个整数，使得它们的和与 $target$ 最接近。

返回这三个数的和。

假定每组输入只存在唯一答案。

示例：

输入：nums = [-1,2,1,-4], target = 1

输出：2

解释：与 target 最接近的和是 2 ($-1 + 2 + 1 = 2$)。

提示：

- $3 \leq \text{nums.length} \leq 10^3$
- $-10^3 \leq \text{nums}[i] \leq 10^3$
- $-10^4 \leq \text{target} \leq 10^4$

排序 + 双指针

这道题的思路和「15. 三数之和（中等）」区别不大。

对数组进行排序，使用三个指针 `i`、`j` 和 `k` 分别代表要找的三个数。

1. 通过枚举 `i` 确定第一个数，另外两个指针 `j`，`k` 分别从左边 `i + 1` 和右边 `n - 1` 往中间移动，找到满足 `nums[i] + nums[j] + nums[k]` 最接近 `target` 的唯一解。
2. `j` 和 `k` 指针的移动逻辑，分情况讨论 `sum = nums[i] + nums[j] + nums[k]`：
 - `sum > target`：`k` 左移，使 `sum` 变小
 - `sum < target`：`j` 右移，使 `sum` 变大
 - `sum = target`：找到最符合要求的答案，直接返回

为了更快找到答案，对于相同的 `i`，可以直接跳过下标。

代码：

宫水三叶
の
刷题日记

公众号：宫水三叶的刷题日记

```

class Solution {
    public int threeSumClosest(int[] nums, int t) {
        Arrays.sort(nums);
        int ans = nums[0] + nums[1] + nums[2];
        int n = nums.length;
        for (int i = 0; i < n; i++) {
            if (i > 0 && nums[i] == nums[i - 1]) continue;
            int j = i + 1, k = n - 1;
            while (j < k) {
                int sum = nums[i] + nums[j] + nums[k];
                if (Math.abs(sum - t) < Math.abs(ans - t)) ans = sum;
                if (ans == t) {
                    return t;
                } else if (sum > t) {
                    k--;
                } else if (sum < t) {
                    j++;
                }
            }
        }
        return ans;
    }
}

```

- 时间复杂度：排序的复杂度为 $O(\log N)$ ，对于每个 i 而言，最坏的情况 j 和 k 都要扫描一遍数组的剩余部分，复杂度为 $O(n^2)$ 。整体复杂度为 $O(n^2)$
- 空间复杂度： $O(n^2)$

更多精彩内容，欢迎关注：[公众号](#) / [Github](#) / [LeetCode](#) / [知乎](#) **

题目描述

这是 LeetCode 上的 **18. 四数之和**，难度为 **中等**。

Tag：「双指针」、「排序」、「n 数之和」

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 1：

输入：nums = [1,0,-1,0,-2,2], target = 0

输出：[[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

示例 2：

输入：nums = [], target = 0

输出：[]

提示：

- $0 \leq \text{nums.length} \leq 200$
- $-10^9 \leq \text{nums}[i] \leq 10^9$
- $-10^9 \leq \text{target} \leq 10^9$

排序 + 双指针

这道题的思路和「15. 三数之和（中等）」、「16. 最接近的三数之和（中等）」类似。

对数组进行排序，使用四个指针 `i`、`j`、`k` 和 `p` 分别代表要找的四个数。

1. 通过枚举 `i` 确定第一个数，枚举 `j` 确定第二个数，另外两个指针 `k` 和 `p` 分别从左边 `j + 1` 和右边 `n - 1` 往中间移动，找到满足 `nums[i] + nums[j] + nums[k] + nums[p] == t` 的所有组合。
2. `k` 和 `p` 指针的移动逻辑，分情况讨论
`sum = nums[i] + nums[j] + nums[k] + nums[p]`：
 - `sum > target`：`p` 左移，使 `sum` 变小
 - `sum < target`：`k` 右移，使 `sum` 变大
 - `sum = target`：将组合加入结果集，`k` 右移继续进行检查

题目要求不能包含重复元素，所以我们要对 `i`、`j` 和 `k` 进行去重，去重逻辑是对于相同的数，只使用第一个。

代码：

刷题日记

公众号：宫水三叶的刷题日记

```

class Solution {
    public List<List<Integer>> fourSum(int[] nums, int t) {
        Arrays.sort(nums);
        int n = nums.length;
        List<List<Integer>> ans = new ArrayList<>();
        for (int i = 0; i < n; i++) { // 确定第一个数
            if (i > 0 && nums[i] == nums[i - 1]) continue; // 对第一个数进行去重（相同的数只取第一个）
            for (int j = i + 1; j < n; j++) { // 确定第二个数
                if (j > i + 1 && nums[j] == nums[j - 1]) continue; // 对第二个数进行去重（相同的数只取第一个）
                // 确定第三个数和第四个数
                int k = j + 1, p = n - 1;
                while (k < p) {


                    // 对第三个数进行去重（相同的数只取第一个）
                    while (k > j + 1 && k < n && nums[k] == nums[k - 1]) k++;
                    // 如果 k 跳过相同元素之后的位置超过了 p，本次循环结束
                    if (k >= p) break;

                    int sum = nums[i] + nums[j] + nums[k] + nums[p];
                    if (sum == t) {
                        ans.add(Arrays.asList(nums[i], nums[j], nums[k], nums[p]));
                        k++;
                    } else if (sum > t) {
                        p--;
                    } else if (sum < t) {
                        k++;
                    }
                }
            }
        }
        return ans;
    }
}

```

- 时间复杂度： i 和 j 是直接枚举确定，复杂度为 $O(n^2)$ ，当确定下来 i 和 j 之后，通过双指针确定 k 和 p ，也就是对于每一组 i 和 j 而言复杂度为 $O(n)$ 。总的复杂度为 $O(n^3)$
- 空间复杂度： $O(n)$

更多精彩内容，欢迎关注：[公众号](#) / [Github](#) / [LeetCode](#) / [知乎](#)

 **更新 Tips**：本专题更新时间为 2021-10-07，大概每 2-4 周 集中更新一次。

最新专题合集资料下载，可关注公众号「[宫水三叶的刷题日记](#)」，后台回复「n 数之和」获取下

公众号: 宫水三叶的刷题日记

载链接。

觉得专题不错，可以请作者吃糖 🍬🍬🍬：



“给作者手机充个电”

YOLO 的赞赏码

版权声明：任何形式的转载请保留出处 [Wiki](#)。