0018. 四数之和

ITCharge 大约 2 分钟

- 标签：数组、双指针、排序
- 难度：中等

题目链接

- 0018. 四数之和 - 力扣

题目大意

描述：给定一个整数数组 $nums$ 和一个目标值 $target$。

要求：找出所有满足以下条件切不重复的四元组。

1. $0 \leq a, b, c, d < n$。
2. a、b、c 和 d 互不相同。
3. $nums[a] + nums[b] + nums[c] +$ $\imath s[d] == target$。

说明：

- $1 \leq nums.length \leq 200$。
- $-10^9 \leq nums[i] \leq 10^9$。
- $-10^9 \leq target \leq 10^9$。

示例：

- 示例 1：

```py
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：

```py
输入：nums = [2,2,2,2,2], target = 8
输出：[[2,2,2,2]]
```

解题思路

思路 1：排序 + 双指针

和 0015. 三数之和　解法类似。

直接三重遍历查找 a、b、c、d 的时间复杂度是：$O(n^4)$。我们可以通过一些操作来降低复杂度。

1. 先将数组进行排序，以保证按顺序查找 a、b、c、d 时，元素值为升序，从而保证所找到的四个元素是不重复的。同时也方便下一步使用双指针减少一重遍历。这一步的时间复杂度为：$O(n \times \log n)$。
2. 两重循环遍历元素 a、b，对于每个 a 元素，从 a 元素的下一个位置开始遍历元素 b。对于元素 a、b，使用双指针 $left$，$right$ 来查找 c、d。$left$ 指向 b 元素的下一个位置，$right$ 指向末尾位置。先将 $left$ 右移、$right$ 左移去除重复元素，再进行下边的判断。
    1. 如果 $nums[a] + nums[b] + nums[left] + nums[right] == target$，则得到一个解，将其加入答案数组中，并继续将 $left$ 右移，$right$ 左移；
    2. 如果 $nums[a] + nums[b] + nu\cdots[left] + nums[right] > target$，说明 $nums[right]$ 值太大，将 $right$ 向左移；
    3. 如果 $nums[a] + nums[b] + nums[left] + nums[right] < target$，说明 $nums[left]$ 值太小，将 $left$ 右移。

思路 1：代码

py

```
class Solution:
    def fourSum(self, nums: List[int], target: int) -> List[List[int]]:
        n = len(nums)
        nums.sort()
        ans = []

        for i in range(n):
            if i > 0 and nums[i] == nums[i-1]:
                continue
            for j in range(i+1, n):
                if j > i+1 and nums[j] == nums[j-1]:
                    continue
                left = j + 1
```

```
                right = n - 1
                while left < right:
                    while left < right and left > j + 1 and nums[left] ==
nums[left - 1]:
                        left += 1
                    while left < right and right < n - 1 and nums[right + 1] ==
nums[right]:
                        right -= 1
                    if left < right and nums[i] + nums[j] + nums[left] +
nums[right] == target:
                        ans.append([nums[i], nums[j], nums[left], nums[right]])
                        left += 1
                        right -= 1
                    elif nums[i] + nums[j] + nums[left] + nums[right] > target:
                        right -= 1
                    else:
                        left += 1
        return ans
```

思路 1：复杂度分析

- **时间复杂度**：$O(n^3)$，其中 n 为数组中元素个数。
- **空间复杂度**：$O(\log n)$，排序额外空间为 $\log n$。