0259. 较小的三数之和

ITCharge 大约 2 分钟

- 标签：数组、双指针、二分查找、排序
- 难度：中等

题目链接

- 0259. 较小的三数之和 - 力扣

题目大意

描述：给定一个长度为 n 的整数数组和一个目标值 $target$。

要求：寻找能够使条件 $nums[i] + nums[j] + nums[k] < target$ 成立的三元组 (i, j, k) 的个数（$0 <= i < j < k < n$）。

说明：

- 最好在 $O(n^2)$ 的时间复杂度内解决问题。
- $n == nums.length$。
- $0 \leq n \leq 3500$。
- $-100 \leq nums[i] \leq 100$。
- $-100 \leq target \leq 100$。

示例：

- 示例 1：

```py
输入：nums = [-2,0,1,3], target = 2
输出：2
解释：因为一共有两个三元组满足累加和小于 2:
     [-2,0,1]
     [-2,0,3]
```

- 示例 2：

```py
输入：nums = [], target = 0
输出：0
```

解题思路

思路 1：排序 + 双指针

三元组直接枚举的时间复杂度是 $O(n^3)$，明显不符合题目要求。那么可以考虑使用双指针减少循环内的时间复杂度。具体做法如下：

- 先对数组进行从小到大排序。
- 遍历数组，对于数组元素 $nums[i]$，使用两个指针 $left$、$right$。$left$ 指向第 $i+1$ 个元素位置，$right$ 指向数组的最后一个元素位置。
- 在区间 $[left, right]$ 中查找满足 $nums[i] + nums[left] + nums[right] < target$的方案数。
- 计算 $nums[i]$、$nums[left]$、$nums[right]$ 的和，将其与 $target$ 比较。
  - 如果 $nums[i] + nums[left] + nums[right] < target$，则说明 i、$left$、$right$ 作为三元组满足题目要求，同时说明区间 $[left, right]$ 中的元素作为 $right$ 都满足条件，此时将 $left$ 右移，继续判断。
  - 如果 $nums[i] + nums[left] + nums[right] \geq target$，则说明 $right$ 太大了，应该缩小 $right$，然后继续判断。
- 当 $left == right$ 时，区间搜索完毕，继续遍历 $nums[i+1]$。

这种思路使用了两重循环，其中内层循环当 $left == right$ 时循环结束，时间复杂度为 $O(n)$，外层循环时间复杂度也是 $O(n)$。所以算法的整体时间复杂度为 $O(n^2)$，符合题目要求。

思路 1：代码

```py
class Solution:
    def threeSumSmaller(self, nums: List[int], target: int) -> int:
        nums.sort()
        size = len(nums)
        res = 0
        for i in range(size):
            left, right = i + 1, size - 1
            while left < right:
```

```
            total = nums[i] + nums[left] + nums[right]
            if total < target:
                res += (right - left)
                left += 1
            else:
                right -= 1
        return res
```

思路 1：复杂度分析

- **时间复杂度**：$O(n^2)$。
- **空间复杂度**：$O(\log n)$。