0016. 最接近的三数之和

ITCharge 大约 2 分钟

- 标签：数组、双指针、排序
- 难度：中等

题目链接

- 0016. 最接近的三数之和 - 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：从 $nums$ 中选出三个整数，使它们的和与 $target$ 最接近。返回这三个数的和。假定每组输入只存在恰好一个解。

说明：

- $3 \leq nums.length \leq 1000$。
- $-1000 \leq nums[i] \leq 1000$。
- $-10^4 \leq target \leq 10^4$。

示例：

- 示例 1：

```py
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)。
```

- 示例 2：

```py
输入：nums = [0,0,0], target = 1
输出：0
```

解题思路

思路 1：对撞指针

直接暴力枚举三个数的时间复杂度是 $O(n^3)$。很明显的容易超时。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，使用 ans 记录最接近的三数之和。
- 遍历数组，对于数组元素 $nums[i]$，使用两个指针 $left$、$right$。$left$ 指向第 0 个元素位置，$right$ 指向第 $i - 1$ 个元素位置。
- 计算 $nums[i]$、$nums[left]$、$nums[right]$ 的和与 $target$ 的差值，将其与 ans 与 $target$ 的差值作比较。如果差值小，则更新 ans。
  - 如果 $nums[i] + nums[left] + nums[right] < target$，则说明 $left$ 小了，应该将 $left$ 右移，继续查找。
  - 如果 $nums[i] + nums[left] + nums[right] \geq target$，则说明 $right$ 太大了，应该将 $right$ 左移，然后继续判断。
- 当 $left == right$ 时，区间搜索完毕，继续遍历 $nums[i + 1]$。
- 最后输出 ans。

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 $O(n^2)$。

思路 1：代码

py

```
class Solution:
    def threeSumClosest(self, nums: List[int], target: int) -> int:
        nums.sort()
        res = float('inf')
        size = len(nums)
        for i in range(2, size):
            left = 0
            right = i - 1
            while left < right:
                total = nums[left] + nums[right] + nums[i]
                if abs(total - target) < abs(res - target):
                    res = total
                if total < target:
```

```
                left += 1
            else:
                right -= 1

        return res
```

思路 1：复杂度分析

- **时间复杂度**：$O(n^2)$，其中 n 为数组中元素的个数。
- **空间复杂度**：$O(\log n)$，排序需要 $\log n$ 的栈空间。