# 题目

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 <= nums.length <= 10^3

-10^3<= nums[i]<= 10^3

-10^4<= target<= 10^4

类似题目：

167. 两数之和 II - 输入有序数组

15. 三数之和

18. 四数之和

611. 有效三角形的个数

# 分析

## 方法一：双指针

思路：

代码：

class Solution {

public:

int threeSumClosest(vector<int>& nums, int target) {

if (nums.size() < 3) return 0;

int res;

sort(nums.begin(), nums.end());

int left = 0, right = 0, length = nums.size();

res = nums[0] + nums[1] + nums[2];

for (int i = 0; i < length - 2; i++) { // 遍历第一个数

left = i + 1;

right = length - 1;

while (left < right) {

int sum = nums[i] + nums[left] + nums[right];

if (abs(target - sum) < abs(target - res)) {

res = sum;

}

if (sum == target) {

return res;

} else if (sum < target) {

left++;

} else {

right--;

}

}

}

return res;

}

};

或：

class Solution {

public:

int threeSumClosest(vector<int>& nums, int target) {

int res;

if(nums.size()<3)

return 0;

int len=nums.size(),left,right;

std::sort(nums.begin(),nums.end());

res = nums[0] + nums[1] + nums[2];

for(int i=0;i<len;i++) // 这里不需要遍历全部的数

{

left = i+1;

right = len-1;

while(left<right)

{

int sum = nums[i] + nums[left] + nums[right];

if(abs(target-sum)<abs(target-res))

res = sum;

if(sum==target)

return res;

else if(sum<target)

left++;

else

right--;

}

}

return res;

}

};