# 题目

给你一个由n个整数组成的数组nums，和一个目标值target。请你找出并返回满足下述全部条件且不重复的四元组[nums[a], nums[b], nums[c], nums[d]]：

0 <= a, b, c, d< n

a、b、c和d互不相同

nums[a] + nums[b] + nums[c] + nums[d] == target

你可以按任意顺序返回答案。

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 = [2,2,2,2,2], target = 8

输出：[[2,2,2,2]]

提示：

1 <= nums.length <= 200

-109 <= nums[i] <= 109

-109 <= target <= 109

类似题目：

167. 两数之和 II - 输入有序数组

15. 三数之和

16. 最接近的三数之和

611. 有效三角形的个数

# 分析

## 方法一：排序+双指针

思路：

代码：

class Solution {

public:

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

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

vector<vector<int>> ans;

int n = nums.size();

for (int i = 0; i < n - 3; i++) {

if (i > 0 && nums[i] == nums[i - 1]) continue;

for (int j = i + 1; j < n - 2; j++) {

if (j > i + 1 && nums[j] == nums[j - 1]) continue;

int left = j + 1, right = n - 1;

while (left < right) {

long long sum = static\_cast<long long>(nums[i]) + static\_cast<long long>(nums[j]) + static\_cast<long long>(nums[left]) + static\_cast<long long>(nums[right]);

if (sum == target) {

ans.push\_back({nums[i], nums[j], nums[left], nums[right]});

while (left < right && nums[left] == nums[left + 1]) left++;

while (left < right && nums[right] == nums[right - 1]) right--;

left++;

right--;

} else if (sum < target) {

left++;

} else {

right--;

}

}

}

}

return ans;

}

};

在上述代码基础上继续优化：

class Solution {

public:

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

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

vector<vector<int>> ans;

int n = nums.size();

for (int a = 0; a < n - 3; a++) { // 枚举第一个数

long long x = nums[a]; // 使用 long long 避免溢出

if (a > 0 && x == nums[a - 1]) continue; // 跳过重复数字

if (x + nums[a + 1] + nums[a + 2] + nums[a + 3] > target) break; // 优化一

if (x + nums[n - 3] + nums[n - 2] + nums[n - 1] < target) continue; // 优化二

for (int b = a + 1; b < n - 2; b++) { // 枚举第二个数

long long y = nums[b];

if (b > a + 1 && y == nums[b - 1]) continue; // 跳过重复数字

if (x + y + nums[b + 1] + nums[b + 2] > target) break; // 优化一

if (x + y + nums[n - 2] + nums[n - 1] < target) continue; // 优化二

int c = b + 1, d = n - 1;

while (c < d) { // 双指针枚举第三个数和第四个数

long long s = x + y + nums[c] + nums[d]; // 四数之和

if (s > target) d--;

else if (s < target) c++;

else { // s == target

ans.push\_back({(int) x, (int) y, nums[c], nums[d]});

for (c++; c < d && nums[c] == nums[c - 1]; c++); // 跳过重复数字

for (d--; d > c && nums[d] == nums[d + 1]; d--); // 跳过重复数字

}

}

}

}

return ans;

}

};