1. 题目
2. 题干

给你一个包含 n 个整数的数组 nums，判断 nums 中是否存在三个元素 a，b，c ，使得 a + b + c = 0 ？请你找出所有满足条件且不重复的三元组。

注意：答案中不可以包含重复的三元组

1. 示例

给定数组 nums = [-1, 0, 1, 2, -1, -4]，

满足要求的三元组集合为：

[

[-1, 0, 1],

[-1, -1, 2]

]

1. 题解
2. 思路

题目要我们找出所有不重复的组合数。如何才能找到不重复的组合？

在有序的情况下，按照顺序来找一定是不重复的。

因此，首先我们要对这些数进行排序，再来进行查找。

一共要找三个数字，这三个数字的和为一确定值。从第一个数开始枚举，通过三层for循环，就可以将所有的情况都枚举出来。

但是，这样效率不太高，我们可以想办法优化一下。当第一个数位置确定时，若有第一个数 加上 第二个数 和 第三个数的和为0，则第二个数和第三个数的和是确定的，即第一个数的相反数。那么当第一个数确定后，我们继续找其他的组合时，若第二个数+1，则第三个数-1即可，可以看到，这第二个数和第三个数是一种对称关系。只要两者的和大于第一个数的相反数，则第三个数继续左移一位。

若第二个数和第三个数位置重合，那么直接退出到第一个循环中。因为，若第三个数移动到第二个数的位置，则第三个数无法继续左移了，这是第三个数最小的位置，依然不满足条件。之后，第二个数一定是右移为更大的数，而第三个数无法左移到更小的位置，则更加无法满足条件了。

1. 代码实现
2. C++

class Solution {

public:

vector<vector<int>> threeSum(vector<int>& nums) {

int n = nums.size();

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

vector<vector<int>> ans;

//三层循环

for (int first = 0; first < n; ++first) {

//需要和上一次的枚举数不相同

if (first > 0 & nums[first] == nums[first - 1])

continue;

//c对应的指针初始指向数组的最右端

int third = n - 1;

int target = -nums[first];

//枚举第二个数

for (int second = first + 1; second < n; ++second) {

if (second > first + 1 && nums[second] == nums[second - 1])

continue;

//左移third

while (second < third && nums[second] + nums[third]>target)

--third;

//third左移到second处，直接break;

if (second == third)

break;

//满足条件

if (nums[second] + nums[third] == target)

ans.push\_back({ nums[first }, nums[second], nums[third]);

}

}

return ans;

}

};

1. Java

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
public class Solution {  
 public List<List<Integer>> threeSum(int[] nums){  
 int n = nums.length;  
 Arrays.*sort*(nums);  
 List<List<Integer>> ans = new ArrayList<List<Integer>>();  
 //枚举first  
 for(int first=0; first < n; ++first){  
 if(first>0 && nums[first]==nums[first-1])  
 continue;  
 int third= n-1;  
 int target=-nums[first];  
 //枚举second  
 for(int second=first+1; second <n; ++second){  
 if(second > first+1 && nums[second] == nums[second-1])  
 continue;  
 while(second < third && nums[second] + nums[third] > target)  
 --third;  
 if(second==third)  
 break;  
 if(nums[second] + nums[third] == target){  
 ArrayList<Integer> list = new ArrayList<>();  
 list.add(nums[first]);  
 list.add(nums[second]);  
 list.add(nums[third]);  
 ans.add(list);  
 }  
 }  
 }  
 return ans;  
 }  
   
}