

[LintCode领扣题解 \(/problem\)](#) / 三数之和 · 3Sum

三数之和 · 3Sum

中文

[Works Applications \(/problem/?tags=works-applications\)](#)[Adobe \(/problem/?tags=adobe\)](#)[Bloomberg \(/problem/?tags=bloomberg\)](#)[微软 \(/problem/?tags=microsoft\)](#)[脸书 \(/problem/?tags=facebook\)](#)[亚马逊 \(/problem/?tags=amazon\)](#)[两根指针 \(/problem/?tags=two-pointers\)](#)[数组 \(/problem/?tags=array\)](#)[sort \(/problem/?tags=sort\)](#)

描述

给出一个有n个整数的数组S, 在S中找到三个整数a, b, c, 找到所有使得a + b + c = 0的三元组。

① <p>在三元组(a, b, c), 要求a <= b <= c。</p><p>结果不能包含重复的三元组。</p>

样例

例1:

输入: [2, 7, 11, 15]
输出: []

例2:

输入: [-1, 0, 1, 2, -1, -4]
输出: [[-1, 0, 1], [-1, -1, 2]]

在线评测地址: <https://www.lintcode.com/problem/3sum/> (<https://www.lintcode.com/problem/3sum/>)

收起题目描述 ^

语言类型

[ALL \(30\)](#)[java \(13\)](#)[python \(11\)](#)[cpp \(6\)](#)[上传题解](#)

令狐冲

更新于 11/27/2020, 11:24:20 PM

暴力枚举三个数复杂度为 $O(N^3)$ 先考虑2Sum的做法, 假设升序数列a, 对于一组解 a_i, a_j , 另一组解 a_k, a_l 必然满足 $i < k < j < l$ 或 $i < k < l < j$, 因此我们可以用两个指针, 初始时指向数列两端 指向数之和大于目标值时, 右指针向左移使得总和减小, 反之左指针向右移 由此可以用 $O(N)O(N)$ 的复杂度解决2Sum问题, 3Sum则枚举第一个数 $O(N^2)O(N^2)$ 使用有序数列的好处是, 在枚举和移动指针时值相等的数可以跳过, 省去去重部分

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param nums : Give an array numbers of n integer
     * @return : Find all unique triplets in the array which gives the sum of zero.
     */
}
```

```

    */
    public List<List<Integer>> threeSum(int[] nums) {
        List<List<Integer>> results = new ArrayList<>();

        if (nums == null || nums.length < 3) {
            return results;
        }

        Arrays.sort(nums);

        for (int i = 0; i < nums.length - 2; i++) {
            // skip duplicate triples with the same first numebr
            if (i > 0 && nums[i] == nums[i - 1]) {
                continue;
            }

            int left = i + 1, right = nums.length - 1;
            int target = -nums[i];

            twoSum(nums, left, right, target, results);
        }

        return results;
    }

    public void twoSum(int[] nums,
                      int left,
                      int right,
                      int target,
                      List<List<Integer>> results) {
        while (left < right) {
            if (nums[left] + nums[right] == target) {
                ArrayList<Integer> triple = new ArrayList<>();
                triple.add(-target);
                triple.add(nums[left]);
                triple.add(nums[right]);
                results.add(triple);

                left++;
                right--;
                // skip duplicate pairs with the same left
                while (left < right && nums[left] == nums[left - 1]) {
                    left++;
                }
                // skip duplicate pairs with the same right
                while (left < right && nums[right] == nums[right + 1]) {
                    right--;
                }
            } else if (nums[left] + nums[right] < target) {
                left++;
            } else {
                right--;
            }
        }
    }
}

// 九章硅谷求职算法集训营版本
public class Solution {
    /**
     * @param nums : Give an array numbers of n integer
     * @return : Find all unique triplets in the array which gives the sum of zero.
     */
    List<List<Integer>> results = new ArrayList<>();

    // This is soring results using 1st number as 1st key, 2nd number as 2nd key...
    class ListComparator<T> extends Comparable<T> implements Comparator<List<Integer>> {
        @Override

```

```
public int compare(List<Integer> a, List<Integer> b) {
    for (int i = 0; i < Math.min(a.size(), b.size()); i++) {
        int c = a.get(i).compareTo(b.get(i));
        if (c != 0) {
            return c;
        }
    }
    return Integer.compare(a.size(), b.size());
}

public List<List<Integer>> threeSum(int[] A) {
    if (A == null || A.length < 3) {
        return results;
    }

    Arrays.sort(A);
    // enumerate c, the maximum element
    int i;
    int n = A.length;
    for (i = 2; i < n; ++i) {
        // c is always the last in a bunch of duplicates
        if (i + 1 < n && A[i + 1] == A[i]) {
            continue;
        }

        // want to find all pairs of A[j]+A[k]=-A[i], such that
        // j < k < i
        twoSum(A, i - 1, -A[i]);
    }

    Collections.sort(results, new ListComparator<>());

    return results;
}

// find all unique pairs such that A[i]+A[j]=S, and i < j <= right
private void twoSum(int[] A, int right, int S) {
    int i, j;
    j = right;
    for (i = 0; i <= right; ++i) {
        // A[i] must be the first in its duplicates
        if (i > 0 && A[i] == A[i - 1]) {
            continue;
        }

        while (j > i && A[j] > S - A[i]) {
            --j;
        }

        if (i >= j) {
            break;
        }

        if (A[i] + A[j] == S) {
            List<Integer> t = new ArrayList<>();
            t.add(A[i]);
            t.add(A[j]);
            t.add(-S); // t.add(A[right+1])
            results.add(t);
        }
    }
}
```

👍 获赞 9

💬 5 条评论

你的口袋题库

2000+ 算法真题、国内外名企题库免费开放



九章算法APP

令狐冲

更新于 12/4/2020, 5:46:17 PM

假设 $a \leq b \leq c$

for 循环 a , 找 $b + c = -a$ 即可调用 two sum 的算法来解决。

python

java

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param numbers: Give an array numbers of n integer
    @return: Find all unique triplets in the array which gives the sum of zero.
    """
    def threeSum(self, nums):
        nums = sorted(nums)

        results = []
        for i in range(len(nums)):
            if i > 0 and nums[i] == nums[i - 1]:
                continue
            self.find_two_sum(nums, i + 1, len(nums) - 1, -nums[i], results)

        return results

    def find_two_sum(self, nums, left, right, target, results):
        last_pair = None
        while left < right:
            if nums[left] + nums[right] == target:
                if (nums[left], nums[right]) != last_pair:
                    results.append([-target, nums[left], nums[right]])
                    last_pair = (nums[left], nums[right])
                    right -= 1
                    left += 1
            elif nums[left] + nums[right] > target:
                right -= 1
            else:
                left += 1
```

👍 获赞 5

💬 添加评论

令狐冲

更新于 9/29/2020, 3:39:32 AM

暴力枚举三个数复杂度为 $O(N^3)$ 先考虑2Sum的做法, 假设升序数列 a , 对于一组解 a_i, a_j , 另一组解 a_k, a_l 必然满足 $i < k \ j > l$ 或 $i > k \ j < l$, 因此我们可以用两个指针, 初始时指向数列两端 指向数之和大于目标值时, 右指针向左移使得总和减小, 反之左指针向右移 由此可以用 $O(N)$ 的复杂度解决2Sum问题, 3Sum则枚举第一个数 $O(N^2)$ 使用有序数列的好处是, 在枚举和移动指针时值相等的数可以跳过, 省去去重部分

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param numbers: Give an array numbers of n integer
    @return: Find all unique triplets in the array which gives the sum of zero.
    """
    def threeSum(self, nums):
        nums = sorted(nums)
        results = []
        length = len(nums)
        for i in range(0, length - 2):
            if i > 0 and nums[i] == nums[i - 1]:
                continue
            self.find_two_sum(nums, i + 1, length - 1, -nums[i], results)
        return results

    def find_two_sum(self, nums, left, right, target, results):
        while left < right:
            if nums[left] + nums[right] == target:
                results.append([-target, nums[left], nums[right]])
                right -= 1
                left += 1
                while left < right and nums[left] == nums[left - 1]:
                    left += 1
                while left < right and nums[right] == nums[right + 1]:
                    right -= 1
            elif nums[left] + nums[right] > target:
                right -= 1
            else:
                left += 1

```

👍 获赞 2

💬 7 条评论



令狐冲

更新于 6/9/2020, 7:04:30 AM

暴力枚举三个数复杂度为 $O(N^3)$ 先考虑2Sum的做法, 假设升序数列a, 对于一组解 a_i, a_j , 另一组解 a_k, a_l 必然满足 $i < k < j < l$ 或 $i > k > j > l$, 因此我们可以用两个指针, 初始时指向数列两端 指向数之和大于目标值时, 右指针向左移使得总和减小, 反之左指针向右移 由此可以用 $O(N)O(N)$ 的复杂度解决2Sum问题, 3Sum则枚举第一个数 $O(N^2)O(N^2)$ 使用有序数列的好处是, 在枚举和移动指针时值相等的数可以跳过, 省去去重部分

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
class Solution {
public:
    /**
     * @param numbers : Give an array numbers of n integer
     * @return : Find all unique triplets in the array which gives the sum of zero.
     */
    vector<vector<int> > threeSum(vector<int> &nums) {
        vector<vector<int> > result;

        sort(nums.begin(), nums.end());
        for (int i = 0; i < nums.size(); i++) {
            if (i > 0 && nums[i] == nums[i - 1]) {
                continue;
            }
            // two sum;
            int start = i + 1, end = nums.size() - 1;
            int target = -nums[i];
            while (start < end) {
                if (start > i + 1 && nums[start - 1] == nums[start]) {
                    start++;
                    continue;
                }
                if (nums[start] + nums[end] < target) {
                    start++;
                } else if (nums[start] + nums[end] > target) {
                    end--;
                } else {
                    vector<int> triple;
                    triple.push_back(nums[i]);
                    triple.push_back(nums[start]);
                    triple.push_back(nums[end]);
                    result.push_back(triple);
                    start++;
                }
            }
        }

        return result;
    }
};
```

👍 获赞 0 💬 添加评论



九章用户5TM4C3

更新于 6/9/2020, 7:03:46 AM

这一题提交之后beat 100% submission 所以传出来供大家参考一下。用传统的三根指针来做，关键就是用前后去重的方法，来增加代码运行效率。前后去重分别体现在：line 22, line 42-48

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution {
    public List<List<Integer>> threeSum(int[] nums) {
        List<List<Integer>> results = new ArrayList<>();

        if (nums == null && nums.length < 3){
            return results;
        }

        Arrays.sort(nums);

        for (int i = 0; i < nums.length - 2; i++){
            if(nums[i]>0)break;
            if(i>0 && nums[i]==nums[i-1])continue;

            int left = i + 1;
            int right = nums.length - 1;

            while (left < right){
                int sum = nums[i] + nums[left] + nums[right];
                if (sum == 0){
                    List<Integer> list = new ArrayList<>();
                    list.add(nums[i]);
                    list.add(nums[left]);
                    list.add(nums[right]);

                    results.add(new ArrayList<>(list));

                    left++;
                    right--;

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

                    while (right > left && nums[right] == nums[right + 1]){
                        right--;
                    }
                }else if(sum < 0){
                    left++;
                }else{
                    right--;
                }
            }
        }
        return results;
    }
}

```

(/) 课程 (/course/) 旗舰课 (premium-course/) 1对1私教 (/1on1/) 免费课 (free-course/) 下载 APP 邀请有礼 刷题打卡 成功案例 消息 更多... 我的课程 (/accounts/profile/) (/accounts/)

👍 获赞 6 💬 添加评论



Nepenthes Athene

更新于 6/9/2020, 7:03:46 AM

Python 3

我倒是觉得, 这题就掌握两个思想: 1. 将3Sum退化成2Sum: for 循环每次固定好第一个数, 剩下的问题不就是基础的2 Sum问题吗。2. 去重: 数组sort的以后, 第一层的去重基本是模板了。因为这题比较特殊是3Sum, 意味着只要第二层不重复, 第三层也不会重复。所以在第二层也就是while循环那里套用第一层的去重方法就好了。

时间复杂度 $O(n^2)$ 空间 $O(1)$

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    def threeSum(self, nums):
        """
        :type nums: List[int]
        :rtype: List[List[int]]
        """
        nums.sort()
        ans = []
        for i in range(len(nums)):
            if i > 0 and nums[i] == nums[i-1]:
                continue
            if nums[i] > 0:
                break

            target = 0 - nums[i]
            start, end = i+1, len(nums)-1
            while start < end:
                if start > i+1 and nums[start] == nums[start-1]:
                    start += 1
                    continue
                cur_sum = nums[start] + nums[end]
                if cur_sum == target:
                    ans.append([nums[i], nums[start], nums[end]])
                    start += 1
                elif cur_sum < target:
                    start += 1
                else:
                    end -= 1
            return ans
```

邀请有礼
vitation/sha



👍 获赞 6

💬 1 条评论



Qiao

更新于 6/27/2020, 1:33:22 AM

$a + b + c = 0$ 可以转换为 $target = -a$, 找到满足 $b + c = target$ 的数字。时间复杂度为 $O(n^2)$, 空间复杂度为 $O(n^2)$ 。


```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
#Python3.0
class Solution:
    def threeSum(self, nums):
        """
        :type nums: List[int]
        :rtype: List[List[int]]
        """
        if not nums or len(nums) < 3:
            return []
        result = []
        for i in range(len(nums)-2):
            target = - nums[i]
            dict = {}
            for j in range(i+1, len(nums)):
                if target-nums[j] in dict:
                    res = sorted([nums[i], nums[j], target-nums[j]])
                    if res not in result:
                        result.append(res)
                else:
                    dict[nums[j]] = j
        return result
```

👍 获赞 5

💬 添加评论



九章用户BHJVP5

更新于 7/8/2020, 11:14:30 PM

dfs 排序去重, 纯当练习来做 纯当练习来做 纯当练习来做

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
public List<List<Integer>> threeSum(int[] numbers) {
    Arrays.sort(numbers);
    List<List<Integer>> results = new ArrayList<>();
    dfs(numbers, 0, new ArrayList<>(), results);
    return results;
}

private void dfs(int[] numbers,
                 int startIndex,
                 ArrayList<Integer> buffer,
                 List<List<Integer>> results) {
    if (buffer.size() == 3 &&
        buffer.get(0) + buffer.get(1) + buffer.get(2) == 0) {
        results.add(new ArrayList<>(buffer));
    }

    if (buffer.size() == 3) {
        return;
    }

    for (int i = startIndex; i < numbers.length; i++) {
        if (i > startIndex && numbers[i] == numbers[i - 1]) {
            continue;
        }

        buffer.add(numbers[i]);
        dfs(numbers, i + 1, buffer, results);
        buffer.remove(buffer.size() - 1);
    }
}
```

👍 获赞 4

💬 1 条评论

**S同学**

更新于 6/9/2020, 7:03:50 AM

和九章答案解法思想类似, 时间复杂度 $O(n^2)$, 空间复杂度 $O(n)$

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
public class Solution {
    /**
     * @param numbers: Give an array numbers of n integer
     * @return: Find all unique triplets in the array which gives the sum of zero.
     */
    public List<List<Integer>> threeSum(int[] numbers) {
        // write your code here
        List<List<Integer>> result = new ArrayList<>();
        Arrays.sort(numbers);

        for (int i = 0; i < numbers.length; i++) {
            if (i > 0 && numbers[i] == numbers[i - 1]) {
                continue;
            }

            int start = i + 1, end = numbers.length - 1;
            int target = -numbers[i];
            twoSum(numbers, target, result, start, end);
        }

        return result;
    }

    public void twoSum(int[] numbers, int target, List<List<Integer>> result, int start, int end) {
        int begin = start;
        while (start < end) {
            if (start > begin && numbers[start] == numbers[start - 1]) {
                start++;
                continue;
            }

            if (numbers[start] + numbers[end] == target) {
                List<Integer> list = new ArrayList<>();

                list.add(-target);
                list.add(numbers[start]);
                list.add(numbers[end]);

                result.add(list);

                start++;
                end--;
                continue;
            }

            if (numbers[start] + numbers[end] < target) {
                start++;
            } else {
                end--;
            }
        }
    }
}
```

👍 获赞 2 💬 添加评论

**Tin**

更新于 6/9/2020, 7:03:49 AM

题号是57, 算是经典老题了。面试时碰到, 敢说自己能代码写的漂亮吗?

代码整齐对仗的要点是:

1. 锁定最后(第三个数)一个数, 并倒序, 这样, 第一第二个数, 求 2sum 时, 循环的初始就比较自然。
2. 做和判定时, 相等放在 else 条件里, 这样大于和小于就对称了。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:

    def threeSum(self, nums):

        nums.sort()

        output = []

        for k in reversed(range(2, len(nums))):
            if k+1 < len(nums) and nums[k] == nums[k+1]:
                continue
            i, j, target = 0, k-1, -nums[k]
            while i < j:
                if nums[i] + nums[j] < target:
                    i += 1
                elif nums[i] + nums[j] > target:
                    j -= 1
                else:
                    output.append((nums[i], nums[j], nums[k]))
                    i += 1
                    while i < j and nums[i] == nums[i-1]:
                        i += 1
                    j -= 1
                    while i < j and nums[j] == nums[j+1]:
                        j -= 1
            return output
```

👍 获赞 2 💬 添加评论

**九章用户RBODW7**

更新于 6/9/2020, 7:03:55 AM

用了上课讲的方法, 找 $b+c=-a$ ($a \leq b \leq c$)。 $O(n^2)$ 的时间复杂度, $O(1)$ 的空间复杂度。先排序

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param numbers: Give an array numbers of n integer
    @return: Find all unique triplets in the array which gives the sum of zero.
    """
    def threeSum(self, numbers):
        # for each a, find b+c = -a
        if not numbers or len(numbers) < 3:
            return []

        result = []
        minimum = 0
        numbers.sort()

        while minimum < len(numbers) - 2:
            self.twoSum(numbers[minimum + 1:], -numbers[minimum], result)
            minimum += 1
            while minimum < len(numbers) - 2 and numbers[minimum] == numbers[minimum - 1]:
                minimum += 1

        return result

    def twoSum(self, numbers, target, pairs):
        left, right = 0, len(numbers) - 1
        while left < right:
            if numbers[left] + numbers[right] == target:
                pairs.append([-target, numbers[left], numbers[right]])
                left += 1
                right -= 1
                while left < right and numbers[left] == numbers[left - 1]:
                    left += 1
                while left < right and numbers[right] == numbers[right + 1]:
                    right -= 1
            if numbers[left] + numbers[right] < target:
                left += 1
            if numbers[left] + numbers[right] > target:
                right -= 1

        return
```

👍 获赞 1

💬 添加评论

**Eric**

更新于 6/9/2020, 7:03:52 AM

经典的三指针, 固定i, j和k相向指针移动, 注意去重复, 就是Ai () == Ai-1 ()去掉, 还有收集到一对i,j,k之后, Aj () == Aj-1 () Ak () == Ak+1 ()两种情况去掉;

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param numbers: Give an array numbers of n integer
     * @return: Find all unique triplets in the array which gives the sum of zero.
     */
    public List<List<Integer>> threeSum(int[] numbers) {
        List<List<Integer>> lists = new ArrayList<List<Integer>>();
        if(numbers == null || numbers.length == 0) {
            return lists;
        }

        Arrays.sort(numbers);
        for(int i = 0; i < numbers.length - 2; i++) {
            if(i > 0 && numbers[i] == numbers[i-1]){
                continue;
            }
            int j = i + 1;
            int k = numbers.length - 1;
            while(j < k) {
                int sum = numbers[i] + numbers[j] + numbers[k];
                if(sum == 0) {
                    List<Integer> list = new ArrayList<Integer>();
                    list.add(numbers[i]);
                    list.add(numbers[j]);
                    list.add(numbers[k]);
                    lists.add(list);
                    j++;
                    k--;

                    while(j < k && numbers[j] == numbers[j-1]) {
                        j++;
                    }

                    while(j < k && numbers[k] == numbers[k+1]) {
                        k--;
                    }
                } else if(sum > 0) {
                    k--;
                } else {
                    // sum < 0;
                    j++;
                }
            }
        }
        return lists;
    }
}
```

👍 获赞 1

💬 添加评论

加载更多题解

进阶课程

视频+互动	直播+互动	直播+互动	互动课
<div>九章算法班 2021 版</div> <div>8周时间精通 57 个核心高频考点，9招击破 FLAG、BATJ 算法面试。22....</div>	<div>系统架构设计 System Design 2021 版</div> <div>成为百万架构师必上。30 课时带你快速掌握 18大系统架构设计知识点与面...</div>	<div>九章算法面试高频题冲刺班</div> <div>每期更新 15% 题目，考前押题，一举拿下FLAG & BATJ Offer</div>	<div>面向对象设计 OOD</div> <div>应届生及亚马逊面试必考，IT求职必备基础</div>