查找 2

1.给定一个整数数组 *nums* 和一个目标值 *target*，请你在该数组中找出和为目标值的那 **两个** 整数，

并返回他们的数组下标。你可以假设每种输入只会对应一个答案。但是，数组中同一个元素不能使用两遍。

示例:

给定 nums = [2, 7, 11, 15], target = 9

因为 nums[0] + nums[1] = 2 + 7 = 9

所以返回 [0, 1]

排序法

```
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        #排序法
        n=len(nums)
        index=list(range(n))
        index=sorted(index,key=lambda s:nums[s])
        low,high=0,n-1
        while low<high:
            if nums[index[low]]+nums[index[high]]==target:
                return [index[low],index[high]]
            elif nums[index[low]]+nums[index[high]]>target:
                high-=1
            else:
                low+=1
```

LeetCode15 题

题目描述

给出一个整型数组，寻找其中的所有不同的三元组(*a*,*b*,*c*)，使得 *a+b+c=0*

注意：答案中不可以包含重复的三元组。

如：*nums = [-1, 0, 1, 2, -1, -4]*,

结果为：*[[-1, 0, 1],[-1, -1, 2]]*

代码：

```
class Solution:
    def threeSum(self, nums: [int]) -> [[int]]:
    nums.sort()
    res = []
    for i in range(len(nums)-2):
```

```
        if nums[i] > 0: break
        if i > 0 and nums[i] == nums[i-1]: continue
            l,r = i+1, len(nums)-1
        while l < r:
            sum = nums[i] + nums[l] + nums[r]
            if sum == 0:
                res.append([nums[i],nums[l],nums[r]])
            l += 1
            r -= 1
            while l < r and nums[l] == nums[l-1]: l += 1
            while l < r and nums[r] == nums[r+1]: r -= 1
            elif sum < 0:
                l += 1
            else:
                r -= 1
        return res
```

LeetCode 第 18 题

题目描述

给出一个整形数组，寻找其中的所有不同的四元组(a,b,c,d)，使得 $a+b+c+d$ 等于一个给定的数字 *target*。

代码如下：

```
class Solution:
    def fourSum(self, nums: List[int], target: int) -> List[List[int]]:
        nums.sort()
        res = []
        if len(nums) < 4: return res
        if len(nums) == 4 and sum(nums) == target:
            res.append(nums)
            return res
        for i in range(len(nums)-3):
            if i > 0 and nums[i] == nums[i-1]: continue
        for j in range(i+1,len(nums)-2):
            if j > i+1 and nums[j] == nums[j-1]: continue
            l,r = j+1, len(nums)-1
        while l < r:
            sum_value = nums[i] + nums[j] + nums[l] + nums[r]
        if sum_value == target:
            res.append([nums[i],nums[j],nums[l],nums[r]])
```

```
            l += 1
            r -= 1
        while l < r and nums[l] == nums[l-1]: l += 1
        while l < r and nums[r] == nums[r+1]: r -= 1
        elif sum_value < target:
            l += 1
        else:
            r -= 1
            return res
```

LeetCode 第 16 题最接近三数之和

题目描述

给出一个整形数组，寻找其中的三个元素 a,b,c，使得 a+b+c 的值最接近另外一个给定的数字 target。

如：给定数组 nums = [-1，2，1，-4]，和 target = 1.

与 target 最接近的三个数的和为 2. (-1 + 2 + 1 = 2).

代码实现：

```
class Solution:
    def threeSumClosest(self, nums: List[int], target: int) -> int:
        nums.sort()
        diff = abs(nums[0]+nums[1]+nums[2]-target)
        res = nums[0] + nums[1] + nums[2]
        for i in range(len(nums)):
            l,r = i+1,len(nums)-1
            t = target - nums[i]
        while l < r:
            if nums[l] + nums[r] == t:
                return nums[i] + t
            else:
                if abs(nums[l]+nums[r]-t) < diff:
                    diff = abs(nums[l]+nums[r]-t)
                    res = nums[i]+nums[l]+nums[r]
                if nums[l]+nums[r] < t:
                    l += 1
                else:
                    r -= 1
                return res
```

LeetCode 49 Group Anagrams

题目描述

给出一个字符串数组，将其中所有可以通过颠倒字符顺序产生相同结果的单词进行分组。

示例：
输入：["eat", "tea", "tan", "ate", "nat", "bat"],
输出:[["ate","eat","tea"],["nat","tan"],["bat"]]
说明：
所有输入均为小写字母。
不考虑答案输出的顺序。

代码如下：

```
class Solution:
    def groupAnagrams(self, strs: List[str]) -> List[List[str]]:
        from collections import defaultdict
        strs_dict = defaultdict(list)
        res = []
        for str in strs:
            key = ''.join(sorted(list(str)))
            strs_dict[key] += str.split(',')
        for v in strs_dict.values():
            res.append(v)
        return res
```

LeetCode 447 Number of Boomerangs
题目描述

给出一个平面上的 n 个点，寻找存在多少个由这些点构成的三元组(*i,j,k*)，使得 *i,j* 两点的距离等于 *i,k* 两点的距离。 其中 n 最多为 *500*,且所有的点坐标的范围在[*-10000,10000*]之间。

代码实现：

```
class Solution:
    def numberOfBoomerangs(self, points: List[List[int]]) -> int:
        from collections import Counter
        def f(x1, y1):
            d = Counter((x2 - x1) ** 2 + (y2 - y1) ** 2 for x2, y2 in
            points)
            return sum(t * (t-1) for t in d.values())
    return sum(f(x1, y1) for x1, y1 in points)
```

LeetCode 149 Max Points on a Line

题目描述

给定一个二维平面，平面上有 n 个点，求最多有多少个点在同一条直线上。

示例 1:

输入: [[1,1],[2,2],[3,3]]

输出: 3

示例 2:

输入: [[1,1],[3,2],[5,3],[4,1],[2,3],[1,4]]

输出: 4

代码如下：

```
class Solution:
def maxPoints(self,points):
    if len(points) <= 1:
        return len(points)
    res = 0
    from collections import defaultdict
    for i in range(len(points)):
        record = defaultdict(int)
        samepoint = 0
    for j in range(len(points)):
        if points[i][0] == points[j][0] and points[i][1] == points[j][1]:
            samepoint += 1
    else:
        record[self.get_Slope(points,i,j)] += 1
        for v in record.values():
            res = max(res, v+samepoint)
            res = max(res, samepoint)
        return res
```