题目描述 | 评论 (2.2k) | 题解 (2.1k) | 提交记录

15. 三数之和

难度 中等 3084

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 $a + b + c = 0$？请你找出所有和为 `0` 且不重复的三元组。

注意：答案中不可以包含重复的三元组。

示例 1：

```
输入：nums = [-1,0,1,2,-1,-4]
输出：[[-1,-1,2],[-1,0,1]]
```

示例 2：

```
输入：nums = []
输出：[]
```


题目描述 | 评论 (2.2k) | 题解 (2.1k) | 提交记录

执行结果： 通过 显示详情 >

执行用时： 1596 ms ，在所有 Python3 提交中击败了 16.76% 的用户

内存消耗： 17.3 MB ，在所有 Python3 提交中击败了 85.16% 的用户

炫耀一下：

写题解，分享我的解题思路

```python
class Solution:
    def threeSum(self, nums: List[int]) -> List[List[int]]:
        L = len(nums)
        if L<1:
```

```
    return []
nums = sorted(nums)
re = []
#check = set()
# a<=b<=c
i = 0
while i<L-2:
    v1 = nums[i]
    if i>0 and v1 == nums[i-1]:
        i = i + 1
        continue
    j=i+1
    k = L-1
    while j<L-1:
        v2 = nums[j]
        #print(v1, v2)
        if j>i+1 and v2==nums[j-1]:
            j = j + 1
            continue
        while k>j:
            v3 = nums[k]
            if v3 < -(v1+v2):
                break
            if k < L-1 and v3 == nums[k+1]:
                k=k-1
                continue
            #print('===', v1, v2, v3)
            if v1+v2+v3 == 0:
                #print('---', i, j, k)
                #line = str(v1) + str(v2) + str(v3)
                re.append( [v1,v2,v3] )
            k = k - 1
        j = j + 1
    i = i + 1
#print(re)
return re
```

题目描述 | 评论 (1.1k) | 题解 (1.1k) | 提交记录

12. 整数转罗马数字

难度 中等 514

罗马数字包含以下七种字符： `I`， `V`， `X`， `L`，`C`，`D` 和 `M`。

字符	数值
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

例如， 罗马数字 2 写做 `II` ，即为两个并列的 1。12 写做 `XII` ，即为 `X` + `II` 。 27 写做 `XXVII`, 即为 `XX` + `V` + `II` 。

通常情况下，罗马数字中小的数字在大的数字的右边。但也存在特例，例如 4 不写做 `IIII`，而是 `IV`。数字 1 在数字 5 的左边，所表示的数等于大数 5 减小数 1 得到的数值 4 。同样地，数字 9 表示为 `IX`。这个特殊的规则只适用于以下六种情况：

题目描述 | 评论 (1.1k) | 题解 (1.1k) | 提交记录

执行结果： 通过 显示详情 >

执行用时： 56 ms , 在所有 Python3 提交中击败了 62.91% 的用户

内存消耗： 14.9 MB , 在所有 Python3 提交中击败了 24.68% 的用户

炫耀一下：

写题解，分享我的解题思路

```python
class Solution:
    def intToRoman(self, num: int) -> str:
        dic1 = { 1:'I', 5:'V', 10:'X', 50:'L', 100:'C', 500:'D',
1000:'M', 4:"IV", 9:"IX", 40:"XL", 90:"XC", 400:"CD", 900:"CM" }
        line = ""
```

```
        base = 1
        while num > 0:
            d = num%10
            n = d*base
            sub = self.roman(n, dic1)
            line = sub + line
            base = base * 10
            num = num//10

        return line

    def roman(self, n, dic1):
        if n  == 0:
            return ""
        R = ""
        if n < 4:
            R = n*str( dic1[1] )
        elif n == 4 or n == 5:
            R = str( dic1[n] )
        elif n < 9:
            R = str( dic1[5] ) + (n-5)*str( dic1[1] )
        elif n ==9 or n ==10:
            R = str( dic1[n] )
        #
        elif n<40:
            R = (n//10)*str( dic1[10] )
        elif n == 40 or n == 50:
            R = str( dic1[n] )
        elif n < 90:
            R = str( dic1[50] ) + (n//10-5)*str( dic1[10] )
        elif n ==90 or n ==100:
            R = str( dic1[n] )
        #
        elif n<400:
            R = (n//100)*str( dic1[100] )
        elif n == 400 or n == 500:
            R = str( dic1[n] )
        elif n < 900:
            R = str( dic1[500] ) + (n//100-5)*str( dic1[100] )
        elif n ==900 or n ==1000:
            R = str( dic1[n] )
        #
        else:
            R = (n//1000)*str( dic1[1000] )
```

```
        return R
```

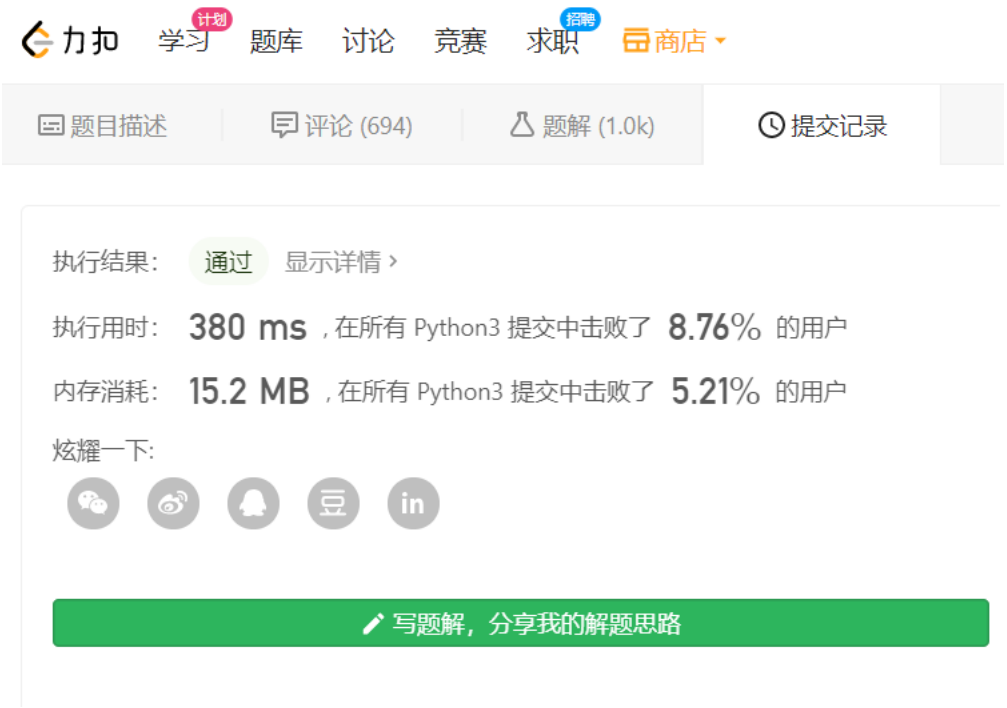
16. 最接近的三数之和

难度 中等 721

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) 。
```




```
class Solution:
    def threeSumClosest(self, nums: List[int], target: int) -> int:
        L = len(nums)
        if L<3:
            return 0
        nums = sorted(nums)
        N = nums[0]
```

```
        # a<=b<=c
        i = 0
        j = 0
        pv = nums[0]+nums[1]+nums[2]
        op = pv
        print(nums)
        while i<L-2:
            j = i + 1
            k = L - 1
            v1 = nums[i]
            while j<L-1:
                v2 = nums[j]
                while k>j:
                    v3 = nums[k]
                    s = v1+v2+v3
                    if self.update(target, op, s):
                        op = s
                        if op == target:
                            return op
                    if s < target:
                        break
                    k = k-1
                j = j + 1
            i = i+1
        print(op)
        return op

    def update(self, target, op, pv):
        d1 = abs(op-target)
        d2 = abs(pv-target)
        print("===", op, pv)
        if d1 <= d2:
            return False
        return True
```