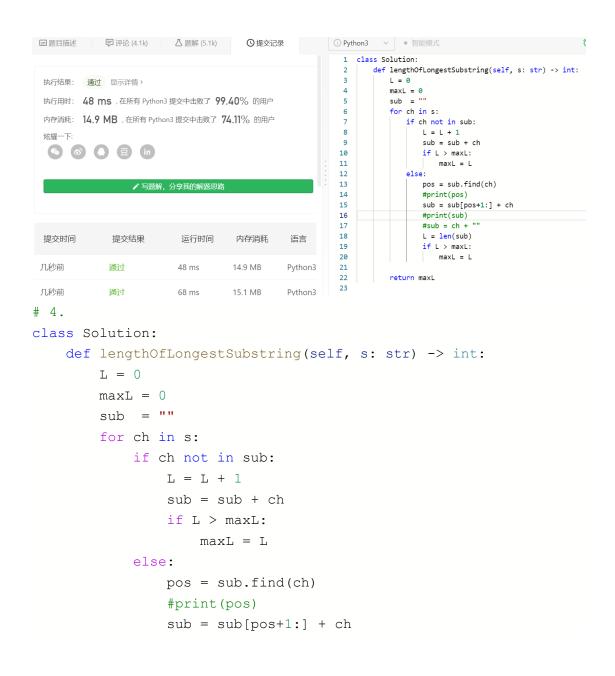


class Solution:

```
def addTwoNumbers(self, l1: ListNode, l2: ListNode) -> ListNo
de:
```

```
v1 = 0
t1 = 11
base = 1
while t1 is not None:
    v1 = v1 + base * t1.val
    t1 = t1.next
    base = base * 10
print(v1)
v2 = 0
t2 = 12
base = 1
while t2 is not None:
    v2 = v2 + base * t2.val
    t2 = t2.next
    base = base * 10
```

```
print(v2)
s = v1 + v2
print(s)
st = str(s)
L = len(st) - 1
t = ListNode(int(st[0]))
i = 1
while i<=L:
    ln = ListNode( int(st[i]), t )
    t = ln
    i = i + 1
return t</pre>
```



```
#print(sub)
#sub = ch + ""

L = len(sub)

if L > maxL:

    maxL = L
```

```
🖃 题目描述
           即评论 (2.5k)
                       △ 题解 (2.5k)
                                   ① 提交记录
                                                 ① Python3

    智能模式

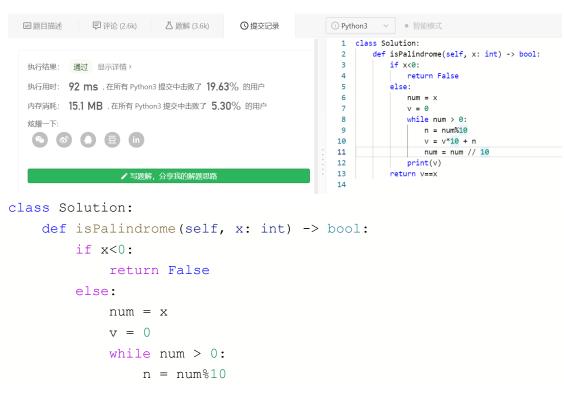
                                                                                             び 模拟面试 i {} 5 >.
                                                      class Solution:
    def findMedianSortedArrays(self, nums1: List[int], nums2: List[int]) -> float:
 执行结果: 通过 显示详情;
                                                            merge = nums1
for i in nums2:
                                                            for 1 in nums2:
    merge.append(i)
merge = sorted(merge)
L = len(merge)
if L % 2 == 1:
    return merge[ L//2 ]
 执行用时: 52 ms . 在所有 Python3 提交中击败了 70.69% 的用户
 内存消耗: 15.1 MB 在所有 Python3 提交中击败了 30.46% 的用户
 炫耀一下:
 00000
                                                            else:
                                                   11
12
13
14
15
                                                               return (merge[ L//2 ] + merge[ L//2 - 1])/2
class Solution1:
       def findMedianSortedArrays(self, nums1: list[int], nums2: lis
t[int]) -> float:
             merge = nums1
              for i in nums2:
                    merge.append(i)
             merge = sorted(merge)
             L = len (merge)
              if L % 2 == 1:
                    return merge[ L//2 ]
             else:
                    return (merge[ L//2 ] + merge[ L//2 - 1])/2
```



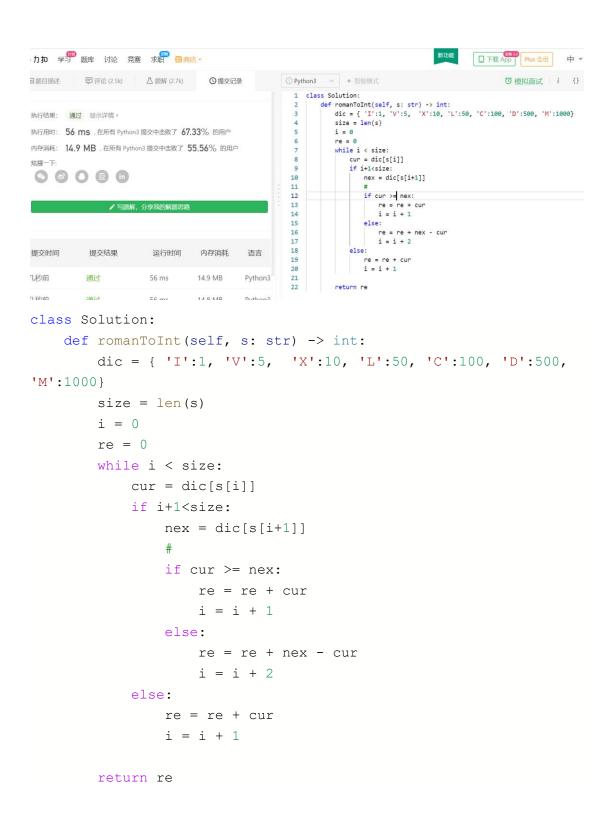
```
class Solution:
    def longestPalindrome(self, s: str) -> str:
        if len(s) <= 1:
            return s
        if len(s) == 2 and s[0] == s[1]:
            return s
        L = len(s)
        target = ""
        for i in range(L):
            print(i)
            left, right, sub = self.expand(s, i, i)
            if len(sub) >= len(target):
                target = sub
            left, right, sub = self.expand(s, i, i+1)
            if len(sub) >= len(target):
                target = sub
            #print(target)
        return target
    def expand(self, s, left, right):
        while left>=0 and right<len(s) and s[left] == s[right]:</pre>
            left = left - 1
            right = right + 1
        #print('--', left+1, right-1, s[left+1:right])
        return left, right, s[left+1:right]
```



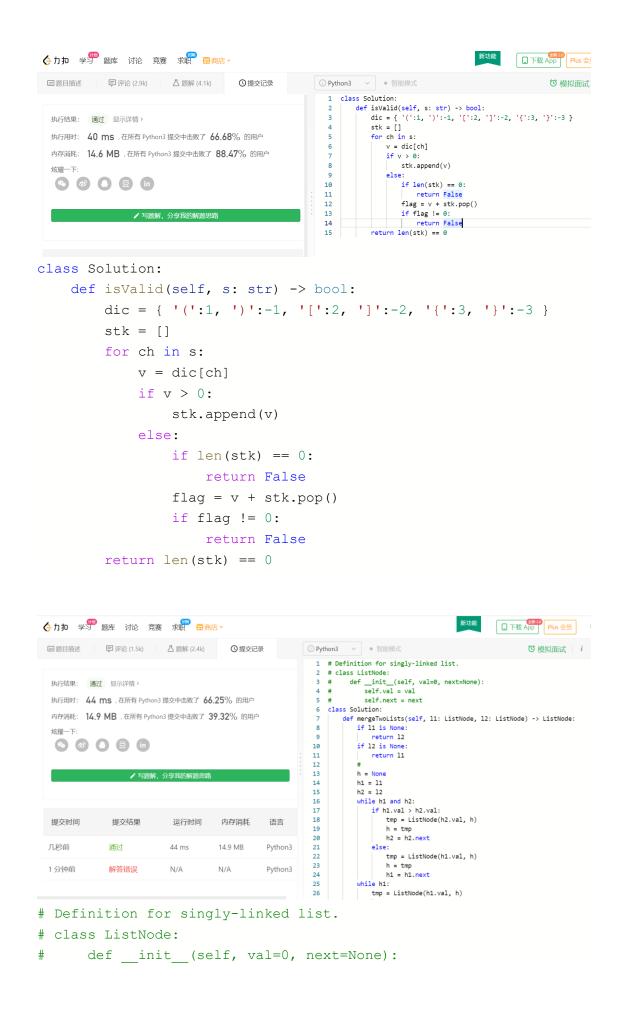
```
class Solution:
    def reverse(self, x: int) -> int:
        if x>0:
            st = str(x)
            base = 1
            v = 0
            for i in st:
                v = v + base*int(i)
                base = base * 10
            if v > math.pow(2, 31)-1:
                return 0
            return v
        else:
            x = -x
            st = str(x)
            base = 1
            v = 0
            for i in st:
                v = v + base*int(i)
                base = base * 10
            v = -v
            if v < -math.pow(2, 31):
                return 0
            return v
```



```
v = v*10 + n
num = num // 10
print(v)
return v==x
```

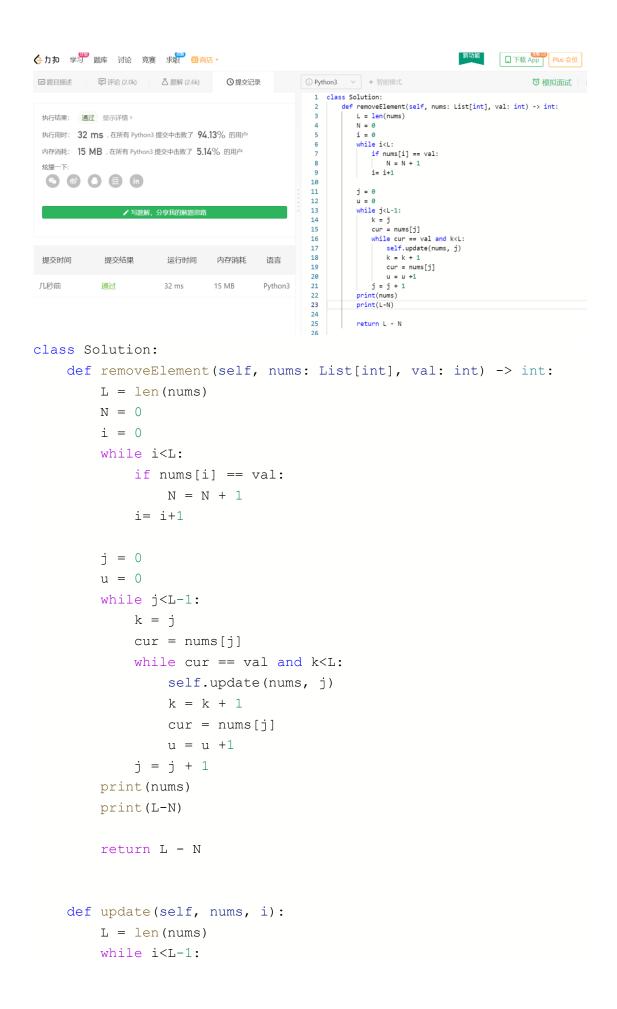




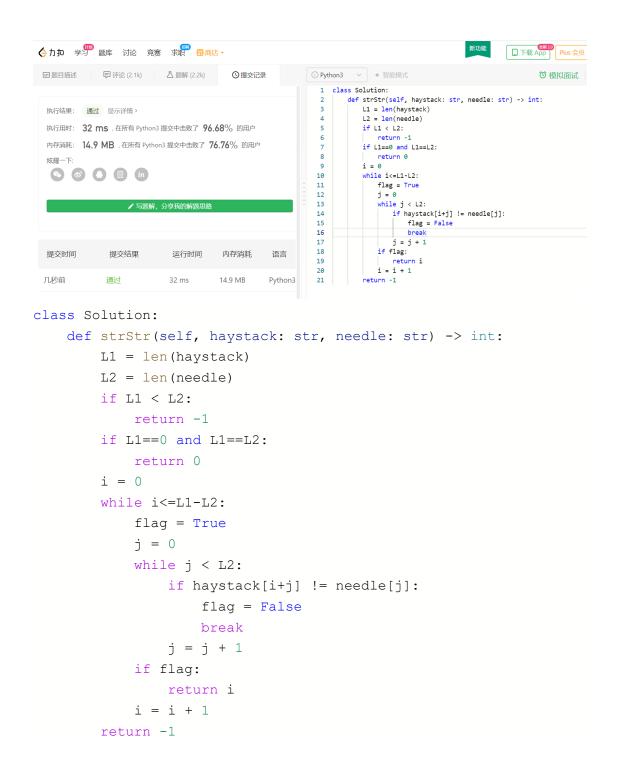


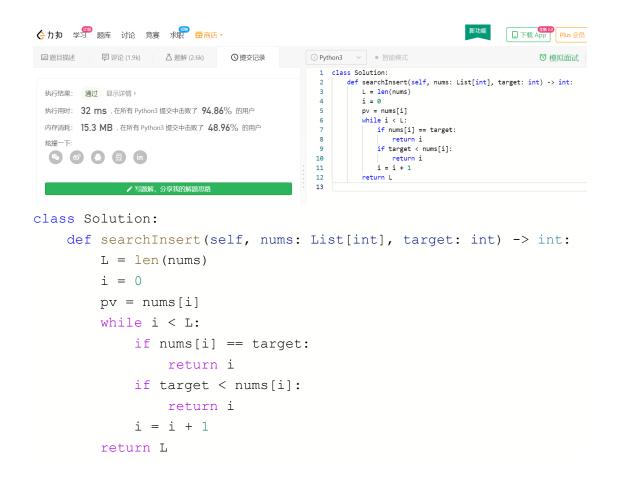
```
self.val = val
          self.next = next
class Solution:
    def mergeTwoLists(self, 11: ListNode, 12: ListNode) -> ListNo
de:
        if 11 is None:
           return 12
        if 12 is None:
           return 11
        h = None
        h1 = 11
        h2 = 12
        while h1 and h2:
            if h1.val > h2.val:
                tmp = ListNode(h2.val, h)
                h = tmp
                h2 = h2.next
            else:
                tmp = ListNode(h1.val, h)
                h = tmp
                h1 = h1.next
        while h1:
            tmp = ListNode(h1.val, h)
            h = tmp
            h1 = h1.next
        while h2:
            tmp = ListNode(h2.val, h)
            h = tmp
            h2 = h2.next
        hh = None
        while h:
            tmp = ListNode(h.val, hh)
            hh = tmp
            h = h.next
        return hh
```

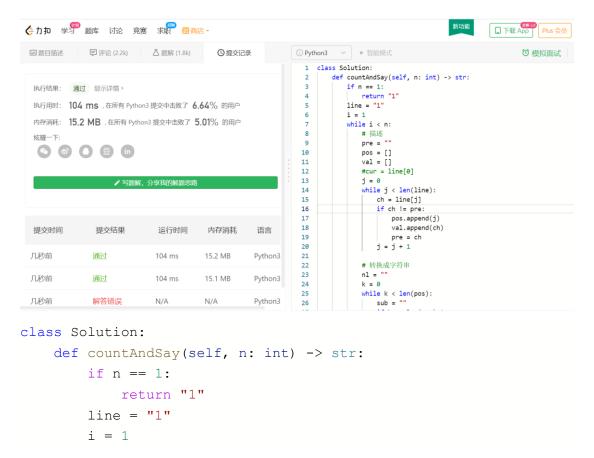




```
nums[i] = nums[i+1]
i = i + 1
```





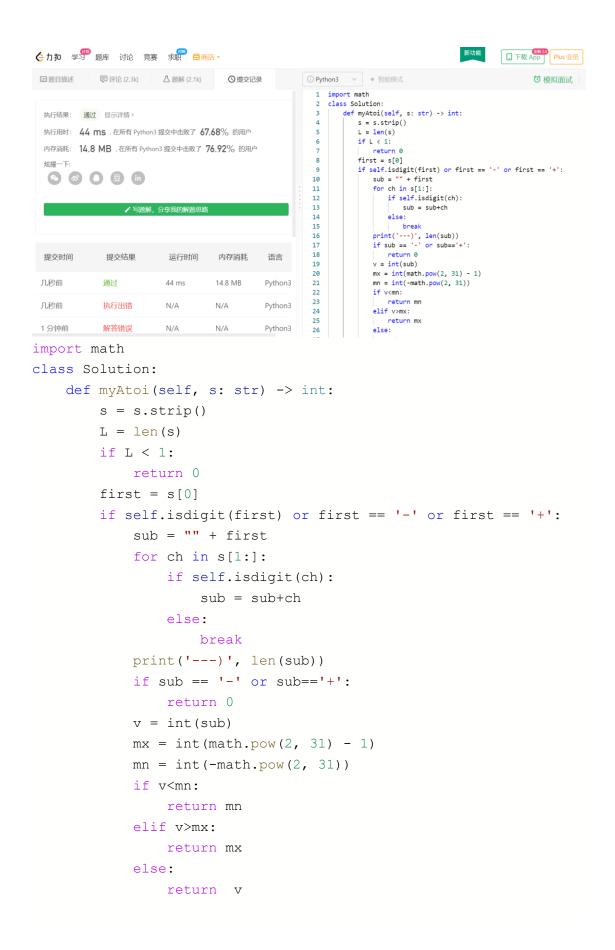


```
while i < n:
    # 描述
    pre = ""
    pos = []
    val = []
    #cur = line[0]
    j = 0
    while j < len(line):</pre>
        ch = line[j]
        if ch != pre:
            pos.append(j)
            val.append(ch)
            pre = ch
        j = j + 1
    # 转换成字符串
    nl = ""
    k = 0
    while k < len(pos):
        sub = ""
        if k == len(pos)-1:
            sub = line[ pos[k]: ]
        else:
            sub = line[pos[k]:pos[k+1]]
        nl = nl + str(len(sub))+str(val[k])
        k = k + 1
    line = nl
    i = i + 1
return line
```



class Solution:

```
def convert(self, s: str, numRows: int) -> str:
    if numRows <= 1:</pre>
        return s
    padding = ' '
    nr = numRows - 1
    rd = numRows * 2 - 2
    ns = ""
    i = 0
    L = len(s)
    col = 0
    while i < L:
        ch = s[i]
        n1 = i%rd
        if n1 < numRows:</pre>
             ns = ns + ch
            col = 0
        else:
            col = col + 1
             col = col % nr
             af = col
            bf = nr - af
             j = 0
            while j<bf:</pre>
                ns = ns + padding
                 j = j + 1
            ns = ns+ch
             j = 0
            while j<af:</pre>
                ns = ns + padding
                 j = j + 1
        i = i + 1
    print(ns)
    line = ""
    i = 0
    while i<numRows:</pre>
        step = numRows
        j = i
        while j<len(ns):</pre>
            if ns[j] != ' ':
                line = line + ns[j]
             j = j + step
        i = i + 1
    return line
```



```
else:
    return 0

def isdigit(self, ch):
    if ch>='0' and ch<='9':
        return True
    else:
        return False</pre>
```



```
def findIndex(self, height, h, L):
    i = 0
    j = L - 1
    idx01 = -1
    idx02 = -1
    while i<L:
       if height[i] >= h:
            idx01 = i
            break
        i = i+1
    print(idx01)
    while j>i:
        if height[j] >= h:
            idx02 = j
            break
        j = j-1
    if idx01 >=0 and idx02>idx01:
       return idx01, idx02
    return -1, -1
def area(self, height, i, j):
    h1 = height[i]
    h2 = height[j]
    h = 0
    if h1<h2:</pre>
       h = h1
    else:
        h = h2
    b = j - i
    return b*h
```