Python 編程之網上刷題

這裡我們用網上評測系統來做做題。英文好的話，可以用 Codeforces 和 LeetCode。中文可以上計蒜客和力扣。這裡用 LeetCode。我這裡做了 10 道題。同時最後 1 題採用了多種方法，把程序效率從擊敗 10% 的提交優化到了擊敗 99%。

Polygon and Codeforces will be possibly unavailable in the period between Mar. 16, 16:00 (UTC) and Mar. 17, 08:00 (UTC) because of maintenance.

Codeforces Round #707 (Div.1, Div.2, based on Moscow Open Olympiad in Informatics, rated)

By ch_egor, 43 hours ago, translation,

Hello!

Right now happens the first tour of the Open Olympiad in Informatics, and tomorrow will be the second one. This contest is prepared by Moscow Olympiad Scientific Committee that you may know by Moscow Team Olympiad, Moscow Olympiad for Young Students and Metropolises Olympiad (rounds 327, 342, 345, 376, 401, 433, 441, 466, 469, 507, 516, 541, 545, 567, 583, 594, 622, 626, 657, 680, 704).

Open Olympiad consists of the most interesting and hard problems that are proposed by a wide community of authors, so we decided to conduct a Codeforces regular round based on it, which will happen on Saturday, March 13, 2021 at 17:05UTC+8 and will be based on **both** days of the Olympiad. Each division will have 6 problems and 2 and a half hours to solve them.

We kindly ask all the community members that are going to participate in the competition to show sportsmanship by not trying to cheat in any manner, in particular, by trying to figure out problem statements from the onsite participants. If you end up knowing some of the problems of Moscow Open Olympiad (by participating in it, from some of the onsite contestants or in any other way), please do not participate in the round. We also ask onsite contestants to not discuss problems in public. Failure to comply with any of the rules above may result in a disqualification.

Problems of this competition were prepared by Akulyat, KiKoS, wrg0ababd, Nebuchadnezzar, biection, alexX512 isaf27, ismagilov.code, DebNatkh, Siberian, NiceClock guided by cdkrot, vintage_Vlad_Makeev, GlebsHP, Zlobober, meshanya, ch_egor,

Before contest
Codeforces Round #708 (Div. 2)
4 days

One person likes this. Sign Up to see

Rating: 1495
Contribution: 0

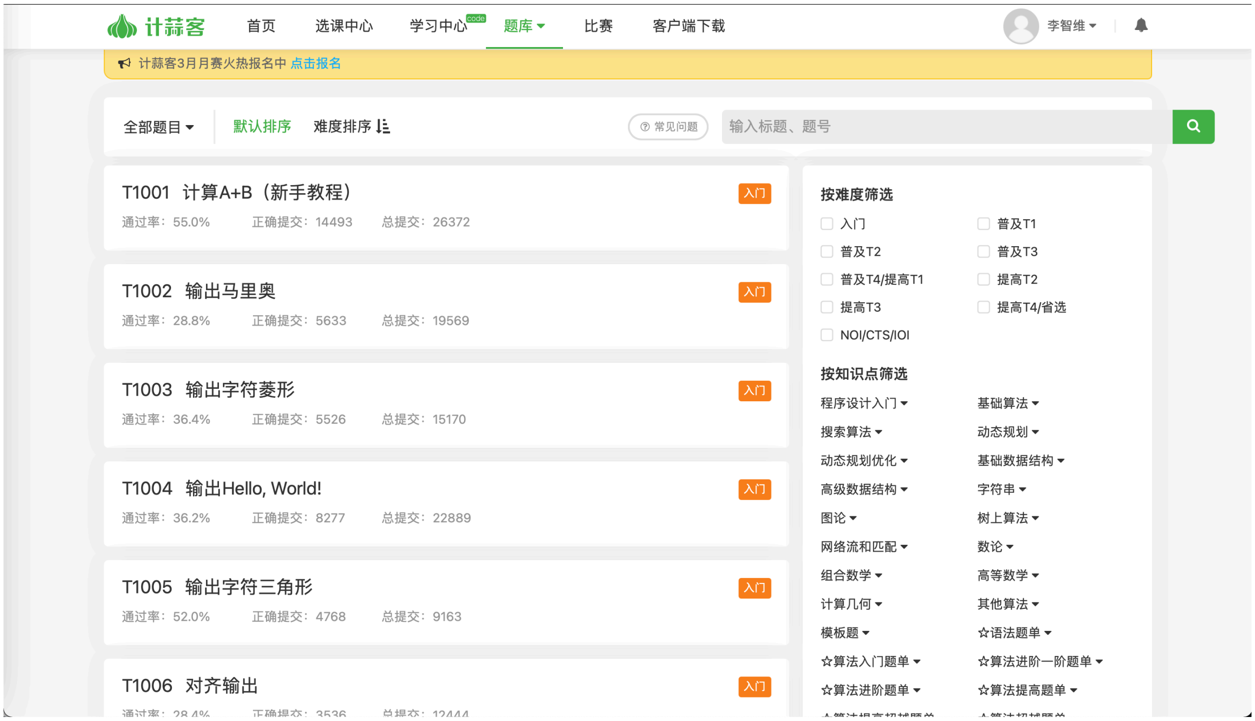
Figure 1: cf

1480. 一維陣列的運行總和

給定一個陣列 nums。我們定義一個陣列的運行總和為 runningSum[i] = sum(nums[0]…nums[i])。

返回 nums 的運行總和。

```
class Solution:
    def runningSum(self, nums: [int]) -> [int]:
      running = []
      s = 0
      for num in nums:
        s += num
        running.append(s)
```



计蒜客3月月赛火热报名中 点击报名

T1001 计算A+B（新手教程） 入门
通过率：55.0% 正确提交：14493 总提交：26372

T1002 输出马里奥 入门
通过率：28.8% 正确提交：5633 总提交：19569

T1003 输出字符菱形 入门
通过率：36.4% 正确提交：5526 总提交：15170

T1004 输出Hello, World! 入门
通过率：36.2% 正确提交：8277 总提交：22889

T1005 输出字符三角形 入门
通过率：52.0% 正确提交：4768 总提交：9163

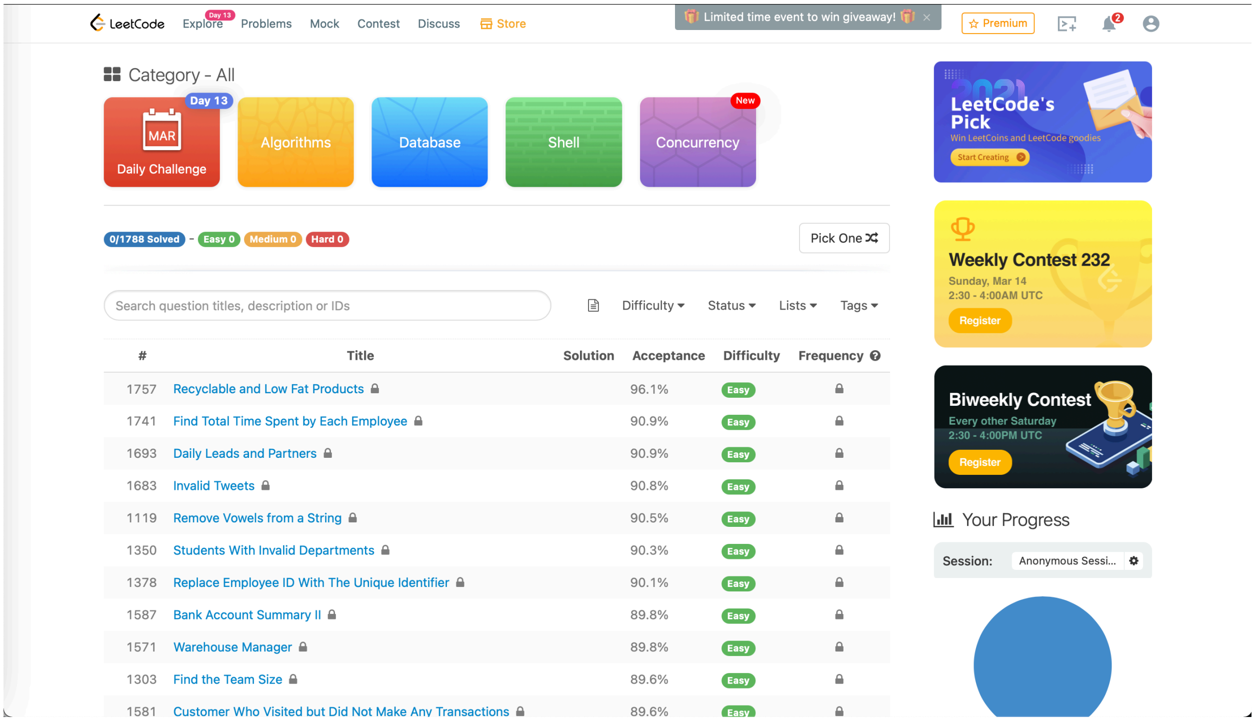
T1006 对齐输出 入门
通过率：28.4% 正确提交：3536 总提交：12444

按难度筛选

- □ 入门
- □ 普及T1
- □ 普及T2
- □ 普及T3
- □ 普及T4/提高T1
- □ 提高T2
- □ 提高T3
- □ 提高T4/省选
- □ NOI/CTS/IOI

按知识点筛选

Figure 2: jsk



Category - All

Weekly Contest 232
Sunday, Mar 14
2:30 - 4:00AM UTC
Register

Biweekly Contest
Every other Saturday
2:30 - 4:00PM UTC
Register

0/1788 Solved - Easy 0 Medium 0 Hard 0

#	Title	Solution	Acceptance	Difficulty	Frequency
1757	Recyclable and Low Fat Products 🔒		96.1%	Easy	🔒
1741	Find Total Time Spent by Each Employee 🔒		90.9%	Easy	🔒
1693	Daily Leads and Partners 🔒		90.9%	Easy	🔒
1683	Invalid Tweets 🔒		90.8%	Easy	🔒
1119	Remove Vowels from a String 🔒		90.5%	Easy	🔒
1350	Students With Invalid Departments 🔒		90.3%	Easy	🔒
1378	Replace Employee ID With The Unique Identifier 🔒		90.1%	Easy	🔒
1587	Bank Account Summary II 🔒		89.8%	Easy	🔒
1571	Warehouse Manager 🔒		89.8%	Easy	🔒
1303	Find the Team Size 🔒		89.6%	Easy	🔒
1581	Customer Who Visited but Did Not Make Any Transactions 🔒		89.6%	Easy	🔒

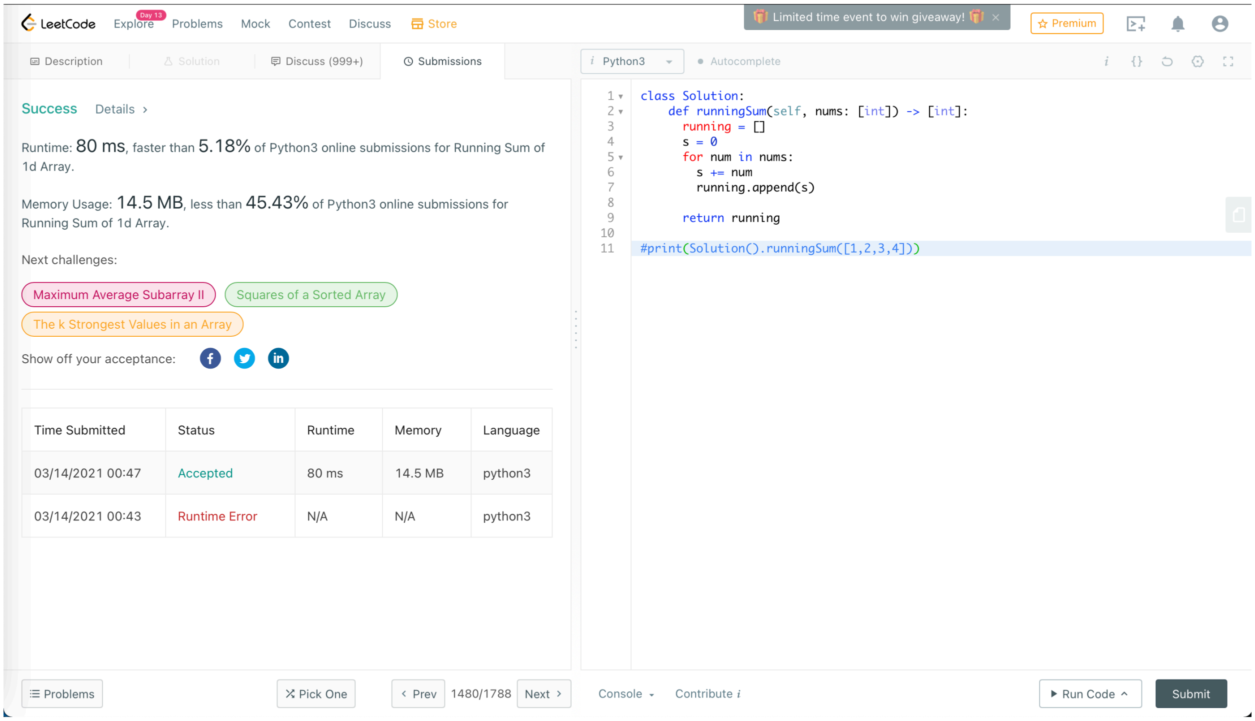
Your Progress

Session: Anonymous Sessi...

Figure 3: leetcode

```
        return running

#print(Solution().runningSum([1,2,3,4]))
```



Success Details ›

Runtime: 80 ms, faster than 5.18% of Python3 online submissions for Running Sum of 1d Array.

Memory Usage: 14.5 MB, less than 45.43% of Python3 online submissions for Running Sum of 1d Array.

Next challenges:

Maximum Average Subarray II   Squares of a Sorted Array   The k Strongest Values in an Array

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
03/14/2021 00:47	Accepted	80 ms	14.5 MB	python3
03/14/2021 00:43	Runtime Error	N/A	N/A	python3

```
class Solution:
    def runningSum(self, nums: [int]) -> [int]:
        running = []
        s = 0
        for num in nums:
            s += num
            running.append(s)

        return running

#print(Solution().runningSum([1,2,3,4]))
```

Figure 4: ac

第一題通過。

1108. 替換 IP 地址中的句點

給定一個有效的（IPv4） IP 地址 `address`，返回該 IP 地址的替換版本。

一個替換的 IP 地址將每個句點"."替換為"[.]"。

```
class Solution:
    def defangIPaddr(self, address: str) -> str:
        return address.replace('.', '[.]')

# print(Solution().defangIPaddr('1.1.1.1'))
```

1431. 擁有最多糖果的孩子

給定陣列 candies 和整數 extraCandies，其中 candies[i] 表示第 i 個孩子擁有的糖果數量。

對於每個孩子，檢查是否存在一種分配 extraCandies 的方法，使得他或她可以擁有最多的糖果。注意，多個孩子可以擁有最多的糖果。

```python
class Solution:
    def kidsWithCandies(self, candies: [int], extraCandies: int) -> [bool]:
        max = 0
        for candy in candies:
          if candy > max:
            max = candy
        greatests = []
        for candy in candies:
          if candy + extraCandies >= max:
            greatests.append(True)
          else:
            greatests.append(False)
        return greatests

# print(Solution().kidsWithCandies([2,3,5,1,3], 3))
```

1672. 最富有客戶的資產

給定一個 m x n 的整數網格 accounts，其中 accounts[i][j] 是第 i 個客戶在第 j 個銀行中的金額。返回最富有客戶的資產。

客戶的資產是他們在所有銀行帳戶中的金額總和。最富有的客戶是擁有最大資產的客戶。

```python
class Solution:
    def maximumWealth(self, accounts: [[int]]) -> int:
        max = 0
        for account in accounts:
          s = sum(account)
          if max < s:
            max = s
        return max

#print(Solution().maximumWealth([[1,2,3],[3,2,1]]))
```

1470. 重新排列陣列

給定一個由 2n 個元素組成的陣列 nums，形式為 [x1,x2,...,xn,y1,y2,...,yn]。

返回陣列的形式為 [x1,y1,x2,y2,...,xn,yn]。

```
class Solution:
  def shuffle(self, nums: [int], n: int) -> [int]:
    ns1 = nums[:n]
    ns2 = nums[n:]
    ns = []
    for i in range(n):
      ns.append(ns1[i])
      ns.append(ns2[i])
    return ns

# print(Solution().shuffle([2,5,1,3,4,7], 3))
```

1512. 好數對的數量

給定一個整數陣列 nums。

如果 nums[i]==nums[j] 且 i<j，則稱 (i,j) 為好數對。

返回好數對的數量。

```
class Solution:
    def numIdenticalPairs(self, nums: [int]) -> int:
        j = 1
        n = len(nums)
        p = 0
        while j < n:
          for i in range(j):
            if nums[i] == nums[j]:
              p += 1
          j+=1
        return p

# print(Solution().numIdenticalPairs([1,2,3,1,1,3]))
```

771. 寶石與石頭

給定字符串 jewels 表示寶石的類型，和 stones 表示你擁有的石頭。stones 中的每個字符代表你擁有的一種石頭。你想知道你擁有的石頭中有多少是寶石。

字母區分大小寫，所以"a" 被認為是不同於"A" 的石頭類型。

```
class Solution:
    def numJewelsInStones(self, jewels: str, stones: str) -> int:
        n = 0
        for i in range(len(jewels)):
          js = jewels[i:i+1]
          n += stones.count(js)
        return n

# print(Solution().numJewelsInStones("aA", "aAAbbbb"))
```

1603. 設計停車系統

為停車場設計一個停車系統。停車場有三種停車位：大、中、小，每種大小有固定數量的停車位。

實現 ParkingSystem 類：

- ParkingSystem(int big, int medium, int small) 初始化 ParkingSystem 類的對象。每種停車位的數量作為構造函數的一部分給出。
- bool addCar(int carType) 檢查是否有 carType 類型的停車位可供汽車進入停車場。carType 可以是三種類型：大、中、小，分別用 1、2 和 3 表示。**汽車只能停在與其 carType 匹配的停車位**。如果沒有可用的停車位，返回 false，否則將汽車停在該大小的停車位並返回 true。

```
class ParkingSystem:
    slots = [0, 0, 0]

    def __init__(self, big: int, medium: int, small: int):
      self.slots[0] = big
      self.slots[1] = medium
      self.slots[2] = small

    def addCar(self, carType: int) -> bool:
      if self.slots[carType - 1] > 0:
        self.slots[carType - 1] -=1
        return True
      else:
        return False

# parkingSystem = ParkingSystem(1, 1, 0)
# print(parkingSystem.addCar(1))
# print(parkingSystem.addCar(2))
# print(parkingSystem.addCar(3))
# print(parkingSystem.addCar(1))
```

1773. 統計匹配規則的項目數量

給定一個陣列 items，其中每個 items[i] = [typei, colori, namei] 描述了第 i 個項目的類型、顏色和名稱。你還給定一個由兩個字符串 ruleKey 和 ruleValue 表示的規則。

如果滿足以下條件之一，則第 i 個項目被認為匹配該規則：

- ruleKey == "type" 且 ruleValue == typei。
- ruleKey == "color" 且 ruleValue == colori。
- ruleKey == "name" 且 ruleValue == namei。

返回匹配給定規則的項目數量。

```
class Solution:
    def countMatches(self, items: [[str]], ruleKey: str, ruleValue: str) -> int:
      i = 0
      if ruleKey == "type":
        i = 0
      elif ruleKey == "color":
        i = 1
      else:
        i = 2
      n = 0
      for item in items:
        if item[i] == ruleValue:
          n +=1
      return n

# print(Solution().countMatches([["phone","blue","pixel"],["computer","silver","lenovo"],["phone","gold
```

1365. 有多少小於當前數字的數字

給定陣列 `nums`，對於每個 `nums[i]`，找出陣列中有多少數字小於它。也就是說，對於每個 `nums[i]`，你需要計算滿足 `j != i` **且** `nums[j] < nums[i]` 的有效 `j` 的數量。

以陣列形式返回答案。

輸入: nums = [8,1,2,2,3]

輸出: [4,0,1,1,3]

解釋:

對於nums[0]=8，有四個數字小於它（1，2，2和3）。

對於nums[1]=1，沒有數字小於它。

對於nums[2]=2，有一個數字小於它（1）。

對於nums[3]=2，有一個數字小於它（1）。

對於nums[4]=3，有三個數字小於它（1，2和2）。

```
class Solution:
    def smallerNumbersThanCurrent(self, nums: [int]) -> [int]:
        ns = []
        l = len(nums)
        for i in range(l):
          n = 0
          for j in range(l):
            if i != j:
              if nums[j] < nums[i]:
                n += 1
          ns.append(n)
        return ns

# print(Solution().smallerNumbersThanCurrent([8,1,2,2,3]))
```

用時 528ms，擊敗了 11.81% 的程序。優化一下。

```
class Solution:
    def smallerNumbersThanCurrent(self, nums: [int]) -> [int]:
        l = len(nums)

        sort_nums = nums.copy()

        ins = list(range(l))
        for i in range(l):
          for j in range(i+1, l):
            if sort_nums[i] > sort_nums[j]:
              a = sort_nums[i]
              sort_nums[i] = sort_nums[j]
              sort_nums[j] = a

              a = ins[i]
              ins[i] = ins[j]
              ins[j] = a

        smalls = [0]
        for i in range(1, l):
          if sort_nums[i-1] == sort_nums[i]:
            smalls.append(smalls[i-1])
          else:
            smalls.append(i)

        # print(sort_nums)
        # print(smalls)

        r_is = list(range(l))
        for i in ins:
          r_is[ins[i]] = i

        ns = []
        for i in range(l):
          ns.append(smalls[r_is[i]])
        return ns

# print(Solution().smallerNumbersThanCurrent([8,1,2,2,3]))
```

這會測試用時 284ms，比剛剛用時 528ms 少。

用寫系統的函數簡寫一下。

```
class Solution:
    def smallerNumbersThanCurrent(self, nums: [int]) -> [int]:
        sort_nums = nums.copy()
        sort_nums.sort()

        ns = []
        for num in nums:
          ns.append(sort_nums.index(num))
        return ns

# print(Solution().smallerNumbersThanCurrent([8,1,2,2,3]))
```

這會只需用時 64ms，擊敗了 71% 的提交。

```
class Solution:
    def smallerNumbersThanCurrent(self, nums: [int]) -> [int]:
        l = len(nums)
        ns = [0] * l
        for i in range(l):
          for j in range(i+1, l):
            if nums[i] > nums[j]:
              ns[i] +=1
            elif nums[i] < nums[j]:
              ns[j] +=1
            else:
              pass
        return ns

# print(Solution().smallerNumbersThanCurrent([8,1,2,2,3]))
```

又想出來一種解法。用時 400ms。

```
class Solution:
    def smallerNumbersThanCurrent(self, nums: [int]) -> [int]:
        ss = sorted((e,i) for i,e in enumerate(nums))

        l = len(nums)
        smalls = [0]
        for i in range(1, l):
          (e0, j0) = ss[i-1]
          (e1, j1) = ss[i]
          if e0 == e1:
            smalls.append(smalls[i-1])
          else:
            smalls.append(i)

        ns = [0]*l
        for i in range(l):
          (e,j) = ss[i]
          ns[j] = smalls[i]
        return ns

# print(Solution().smallerNumbersThanCurrent([8,1,2,2,3]))
```

> Runtime: 52 ms, faster than 91.45% of Python3 online submissions forHow Many Numbers Are Smaller Than the Current Number.
>
> Memory Usage: 14.6 MB, less than 15.18% of Python3 online submissions for How Many Numbers Are Smaller Than the Current Number.

終於成功了！這個方法又更快了，打敗了 91.45% 的提交。

繼續精簡一下。

```
class Solution:
    def smallerNumbersThanCurrent(self, nums: [int]) -> [int]:
        ss = sorted((e,i) for i,e in enumerate(nums))

        l = len(nums)
        smalls = [0]
        ns = [0]*l
        for i in range(1, l):
          (e0, j0) = ss[i-1]
          (e1, j1) = ss[i]
          if e0 == e1:
            smalls.append(smalls[i-1])
          else:
            smalls.append(i)

          ns[j1] = smalls[i]
        return ns

# print(Solution().smallerNumbersThanCurrent([8,1,2,2,3]))
```

繼續。

```
class Solution:
    def smallerNumbersThanCurrent(self, nums: [int]) -> [int]:
        ss = sorted((e,i) for i,e in enumerate(nums))

        l = len(nums)
        last = 0
        ns = [0]*l
        for i in range(1, l):
          (e0, j0) = ss[i-1]
          (e1, j1) = ss[i]
          if e0 == e1:
            pass
```

```python
          else:
            last = i

          ns[j1] = last
        return ns

# print(Solution().smallerNumbersThanCurrent([8,1,2,2,3]))
```

這時我們跑到了 40ms，擊敗了 99.81% 程序。

> Runtime: 40 ms, faster than 99.81% of Python3 online submissions forHow Many Numbers Are Smaller Than the Current Number.
>
> Memory Usage: 14.4 MB, less than 15.18% of Python3 online submissions for How Many Numbers Are Smaller Than the Current Number.

再來一種解法。

```python
class Solution:
    def smallerNumbersThanCurrent(self, nums: [int]) -> [int]:
      l = len(nums)
      n = [0] * 101
      max_num = 0
      for num in nums:
        n[num] += 1
        if num > max_num:
          max_num = num

      sm = [0] * (max_num + 1)
      sum = 0
      for i in range(max_num+1):
        sm[i] = sum
        sum += n[i]

      ns = [0] * l
      for i in range(l):
        ns[i] = sm[nums[i]]
      return ns

# print(Solution().smallerNumbersThanCurrent([8,1,2,2,3]))
```

來個稍微複雜的。

```
class Solution:
    def smallerNumbersThanCurrent(self, nums: [int]) -> [int]:
      l = len(nums)
      n = [0] * 101
      max_num = 0
      for num in nums:
        n[num] += 1
        if num > max_num:
          max_num = num

      short_n = []
      short_num = [] * l
      zn = [0] * 101
      j = 0
      for i in range(max_num+1):
        if n[i] > 0:
          zn[i] = j
          short_n.append(n[i])
          short_num.append(num)
          j+=1

      sm = [0] * j
      sum = 0
      for i in range(j):
        sm[i] = sum
        sum += short_n[i]

      ns = [0] * l
      for i in range(l):
        ns[i] = sm[zn[nums[i]]]
      return ns
```

```
# print(Solution().smallerNumbersThanCurrent([8,1,2,2,3]))
```

“ ‘python class Solution:

```
def smallerNumbersThanCurrent(self, nums: [int]) -> [int]:
  max_num =max(nums)

  n = [0] * (max_num + 1)
  for num in nums:
    n[num] += 1

  sorted_ls = []
  for i in range(max_num + 1):
    if n[i] > 0:
      sorted_ls.append(i)

  sm = [0] * (max_num + 1)
  sum = 0
  for i in range(len(sorted_ls)):
    v = sorted_ls[i]
    sm[v] = sum
    sum += n[v]

  ns = []
```