

Python 代码实现

Hier verwenden wir ein Online-Bewertungssystem, um einige Aufgaben zu lösen. Wenn Ihr Englisch gut ist, könnt ihr Codeforces und LeetCode nutzen. Auf Chinesisch könnt ihr Jisuanke und LintCode verwenden. Hier nutzen wir LeetCode. Ich habe 10 Aufgaben gelöst. Bei der letzten Aufgabe habe ich mehrere Methoden angewendet und die Effizienz des Programms von 10% auf 99% der Einreichungen optimiert.

The screenshot shows the Codeforces homepage with the following details:

- Header:** CODEFORCES, Sponsored by Telegram. User lzjava is logged in.
- Navigation Bar:** HOME (highlighted), TOP, CONTESTS, GYM, PROBLEMSET, GROUPS, RATING, EDU, API, CALENDAR, HELP. A search bar is also present.
- Message Bar:** "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." with a close button.
- Content Area:**
 - Round Information:** "Codeforces Round #707 (Div.1, Div.2, based on Moscow Open Olympiad in Informatics, rated)"
 - Author:** By ch_egor, 43 hours ago, translation,
 - Text:** "Hello!"
 - Description:** "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 17:05 UTC+4 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."
 - Rules:** "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."
 - Preparation:** "Problems of this competition were prepared by Akulyat, KIKoS, wrg0ababd, Nebuchadnezzar, biexiong, alexX512_isaf27, ismaililov.code, DebNatkh, Siberian, NiceClock guided by cdkrot, vintaev, Vlad Makeev, GlebsHP, Zlobober, meshanva, ch_egor."
- User Profile:** lzjava (Rating: 1495, Contribution: 0). Includes links to Settings, Blog, Teams, Submissions, Favourites, Talks, and Contests.
- Ranking:** "Top rated" table with columns #, User, Rating.

Figure 1: cf

1480. Laufende Summe eines 1D-Arrays

Gegeben ein Array `nums`. Wir definieren die laufende Summe eines Arrays als `runningSum[i] = sum(nums[0] ... nums[i])`.

Gib die laufende Summe von `nums` zurück.

```
class Solution:

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

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

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

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

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

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

T1006 对齐输出
通过率: 28.4% 正确提交: 3536 总提交: 12111

Figure 2: jsk

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

Figure 3: leetcode

```

    s += num
    running.append(s)

return running

```

In diesem Python-Code wird eine Methode `runningSum` definiert, die eine Liste von Ganzzahlen (`nums`) als Eingabe nimmt und eine Liste zurückgibt, die die kumulative Summe der Elemente in `nums` enthält. Hier ist eine Übersetzung der Erklärung ins Deutsche:

```

class Solution:

    def runningSum(self, nums: [int]) -> [int]:
        running = [] # Eine leere Liste, um die kumulativen Summen zu speichern
        s = 0 # Eine Variable, um die aktuelle Summe zu speichern
        for num in nums: # Iteriere durch jedes Element in der Eingabeliste
            s += num # Addiere das aktuelle Element zur Summe
            running.append(s) # Füge die aktuelle Summe zur Liste der kumulativen Summen hinzu

        return running # Gib die Liste der kumulativen Summen zurück

```

Diese Methode berechnet die kumulative Summe der Elemente in der Liste `nums` und gibt eine neue Liste zurück, die diese Summen enthält.

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

Die erste Aufgabe ist bestanden.

1108. Entschärfen einer IP-Adresse

Gegeben eine gültige (IPv4) IP-Adresse, gib eine entschärzte Version dieser IP-Adresse zurück.

Eine *entschärzte IP-Adresse* ersetzt jeden Punkt `"."` durch `"[.]"`.

```

class Solution:

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

```

```

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]))

```

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

Figure 4: ac

In diesem Code wird eine Methode `defangIPaddr` definiert, die eine IP-Adresse als Zeichenkette (`address`) entgegennimmt und diese so modifiziert, dass alle Punkte (.) durch [.] ersetzt werden. Dies wird häufig verwendet, um IP-Adressen in einem Format darzustellen, das in URLs oder anderen Kontexten sicherer ist.

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

1431. Kinder mit der größten Anzahl an Süßigkeiten

> Gegeben ist das Array `candies` und die Ganzzahl `extraCandies`, wobei `candies[i]` die Anzahl der Bonbons für Kind i ist.

>

> Überprüfe für jedes Kind, ob es eine Möglichkeit gibt, die `extraCandies` so unter den Kindern zu verteilen, dass jedes Kind mindestens so viele Bonbons erhält wie Kind i .

```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. Reichster Kundenvermögen

> Ihnen wird ein `m x n` Integer-Gitter `accounts` gegeben, wobei `accounts[i][j]` den Geldbetrag darstellt.  
>  
> Der **Wohlstand** eines Kunden ist die Summe des Geldes, das er auf all seinen Bankkonten hat. Der reichste Kunde hat also die höchste Summe.

```

```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

```

Der obige Code bleibt unverändert, da es sich um einen Python-Code handelt, der nicht übersetzt werden muss. Der Code definiert eine Methode `maximumWealth`, die das maximale Vermögen in einer Liste von Konten berechnet.

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

1470. Das Array mischen

Gegeben ist das Array `nums`, das aus $2n$ Elementen in der Form $[x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n]$ besteht.

Gib das Array in der Form $[x_1, y_1, x_2, y_2, \dots, x_n, y_n]$ zurück.

```
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
```

In diesem Code wird eine Methode `shuffle` definiert, die eine Liste von Zahlen `nums` und eine Ganzzahl `n` als Eingabe nimmt. Die Methode teilt die Liste `nums` in zwei Teile auf: `ns1` enthält die ersten `n` Elemente, und `ns2` enthält die restlichen Elemente. Anschließend wird eine neue Liste `ns` erstellt, in der die Elemente aus `ns1` und `ns2` abwechselnd eingefügt werden. Das Ergebnis ist eine gemischte Liste, die zurückgegeben wird.

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

1512. Anzahl der guten Paare

```
> Gegeben ein Array von Ganzzahlen `nums`.
>
> Ein Paar `(i,j)` wird als *gut* bezeichnet, wenn `nums[i]` == `nums[j]` und `i` < `j` ist.
>
> Gib die Anzahl der *guten* Paare zurück.
```

```
```python
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

```

### **Übersetzung:**

```

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

```

### **Erklärung:**

Diese Python-Klasse `Solution` enthält eine Methode `numIdenticalPairs`, die die Anzahl der identischen Paare in einer Liste von Ganzzahlen (`nums`) zählt. Ein identisches Paar liegt vor, wenn zwei Elemente in der Liste den gleichen Wert haben und das erste Element vor dem zweiten Element erscheint.

- `j` ist ein Index, der bei 1 beginnt und sich durch die Liste bewegt.
- `n` ist die Länge der Liste `nums`.
- `p` ist ein Zähler für die Anzahl der identischen Paare.
- Die `while`-Schleife läuft, solange `j` kleiner als `n` ist.
- Die innere `for`-Schleife durchläuft alle Elemente vor dem aktuellen `j`-Index.
- Wenn ein Element vor `j` denselben Wert wie das Element bei `j` hat, wird der Zähler `p` erhöht.
- Schließlich wird der Zähler `p` zurückgegeben, der die Anzahl der identischen Paare darstellt.

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

## 771. Juwelen und Steine

> Ihnen werden die Zeichenketten `jewels` gegeben, die die Arten von Steinen repräsentieren, die Edelsteine sind.  
>  
> Groß- und Kleinschreibung wird beachtet, daher wird `"a"` als eine andere Art von Stein betrachtet als `A`.

```python

```
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
```

(Der Code bleibt auf Englisch, da es sich um eine Programmiersprache handelt und die Variablennamen sowie die Syntax nicht übersetzt werden sollten.)

```
print(Solution().numJewelsInStones("aA", "aAAabbBb"))
```

1603. Parkplatzsystem entwerfen

> Entwerfen Sie ein Parksystem für einen Parkplatz. Der Parkplatz verfügt über drei Arten von Parkplätzen.
>
> Implementieren Sie die Klasse `ParkingSystem`:
>
> - `ParkingSystem(int big, int medium, int small)` Initialisiert ein Objekt der Klasse `ParkingSystem`.
> - `bool addCar(int carType)` Überprüft, ob ein Parkplatz des Typs `carType` für das Auto verfügbar ist.

```python

```
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

```

(Übersetzung nicht notwendig, da es sich um einen Codeblock handelt, der in der Regel nicht übersetzt wird.)

**parkingSystem = ParkingSystem(1, 1, 0)**

**print(parkingSystem.addCar(1))**

**print(parkingSystem.addCar(2))**

**print(parkingSystem.addCar(3))**

**print(parkingSystem.addCar(1))**

## 1773. Artikel zählen, die einer Regel entsprechen

```

> Gegeben ist ein Array `items`, wobei jedes `items[i] = [typei, colori, namei]` den Typ, die Farbe und
>
> Das `i-te` Element passt zur Regel, wenn **eine** der folgenden Bedingungen erfüllt ist:
>
> - `ruleKey == "type"` und `ruleValue == typei`.
> - `ruleKey == "color"` und `ruleValue == colori`.
> - `ruleKey == "name"` und `ruleValue == namei`.

```

```
>
> Gib *die Anzahl der Elemente zurück, die der gegebenen Regel entsprechen*.
```

```
```python  
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
```

Übersetzung:

```
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
```

Erklärung:

Diese Python-Klasse `Solution` enthält eine Methode `countMatches`, die eine Liste von `items`, einen `ruleKey` und einen `ruleValue` als Eingabe erhält. Die Methode zählt, wie viele Elemente in der Liste `items` den angegebenen `ruleKey` und `ruleValue` entsprechen.

- `i` wird verwendet, um den Index zu bestimmen, der dem `ruleKey` entspricht:
 - Wenn `ruleKey` "type" ist, wird `i` auf 0 gesetzt.
 - Wenn `ruleKey` "color" ist, wird `i` auf 1 gesetzt.
 - Andernfalls wird `i` auf 2 gesetzt (für "name").
- `n` ist ein Zähler, der die Anzahl der übereinstimmenden Elemente speichert.
- Die `for`-Schleife durchläuft jedes Element in `items` und überprüft, ob der Wert an der Position `i` mit `ruleValue` übereinstimmt. Wenn ja, wird der Zähler `n` erhöht.
- Schließlich gibt die Methode den Wert von `n` zurück, der die Anzahl der übereinstimmenden Elemente darstellt.

```
print(Solution().countMatches([[“phone”, “blue”, “pixel”],[“computer”, “color”, “silver”)])
```

```
## 1365. Wie viele Zahlen sind kleiner als die aktuelle Zahl
```

```
> Gegeben das Array `nums` , finde für jedes `nums[i]` heraus, wie viele Zahlen im Array kleiner sind als dies.
>
> Gib die Antwort in einem Array zurück.

> ````
> Eingabe: nums = [8,1,2,2,3]
> Ausgabe: [4,0,1,1,3]
> Erklärung:
> Für nums[0]=8 gibt es vier kleinere Zahlen als diese (1, 2, 2 und 3).
> Für nums[1]=1 gibt es keine kleinere Zahl als diese.
> Für nums[2]=2 gibt es eine kleinere Zahl als diese (1).
> Für nums[3]=2 gibt es eine kleinere Zahl als diese (1).
> Für nums[4]=3 gibt es drei kleinere Zahlen als diese (1, 2 und 2).
> ````
```

```

```python
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]))**

Verwendete Zeit: 528ms, schlägt 11,81% der Programme. Optimieren Sie es.

```

```python
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)

```

(Der Code wurde nicht übersetzt, da es sich um eine Programmiersprache handelt, die in der Regel nicht übersetzt wird.)

```

# 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

```

Übersetzung:

```

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

```

Hinweis: Der Code wurde nicht übersetzt, da Code in der Regel in der Originalsprache belassen wird, um die Funktionalität und Lesbarkeit zu gewährleisten.

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

Dieser Test dauerte `284ms`, was weniger ist als die vorherigen `528ms`.

Verwenden Sie die Funktionen des Schreibsystems, um es abzukürzen.

```
```python
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
```

In diesem Code wird eine Methode `smallerNumbersThanCurrent` definiert, die eine Liste von Zahlen `nums` als Eingabe nimmt und eine Liste zurückgibt, die für jede Zahl in `nums` die Anzahl der kleineren Zahlen in der ursprünglichen Liste enthält.

1. Zuerst wird eine Kopie der Eingabeliste `nums` erstellt und sortiert.
2. Dann wird für jede Zahl in der ursprünglichen Liste `nums` der Index dieser Zahl in der sortierten Liste ermittelt, was der Anzahl der kleineren Zahlen entspricht.
3. Diese Indizes werden in einer neuen Liste `ns` gesammelt und zurückgegeben.

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

Dieser Vorgang dauert nur `64ms` und schlägt `71%` der Einreichungen.

```
```python
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
    
```

Der obige Code bleibt unverändert, da es sich um einen Python-Codeblock handelt, der nicht übersetzt werden sollte. Der Code definiert eine Methode `smallerNumbersThanCurrent`, die für jedes Element in der Liste `nums` die Anzahl der Elemente zählt, die kleiner als das aktuelle Element sind. Das Ergebnis wird in einer neuen Liste `ns` gespeichert und zurückgegeben.

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

Ich habe noch eine weitere Lösung gefunden. Die Laufzeit beträgt `400ms`.

```

```python
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

```

Übersetzung:

```

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

```

Der Code bleibt auf Englisch, da es sich um eine Programmiersprache handelt, die international verwendet wird. Die Variablennamen und die Struktur des Codes werden nicht übersetzt.

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

```

> Laufzeit: 52 ms, schneller als 91,45 % der Python3-Online-Einreichungen für "How Many Numbers Are Smaller Than the Current Number"
>
> Speicherverbrauch: 14,6 MB, weniger als 15,18 % der Python3-Online-Einreichungen für "How Many Numbers Are Smaller Than the Current Number"

```

Endlich geschafft! Diese Methode ist noch schneller und schlägt `91,45 %` der Einreichungen.

Weiter vereinfachen.

```

```python
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]))

Fortsetzung folgt.

```

```python
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
 else:
 last = i

ns[j1] = last
return ns

```

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

Zu diesem Zeitpunkt haben wir eine Laufzeit von `40ms` erreicht und `99,81%` der Programme übertroffen.

> Laufzeit: 40 ms, schneller als 99,81 % der Python3-Online-Einreichungen für "How Many Numbers Are Smaller Than the Current Number"

>

> Speicherverbrauch: 14,4 MB, weniger als 15,18 % der Python3-Online-Einreichungen für "How Many Numbers

Hier ist noch eine weitere Lösung.

```
```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
```

Dieser Codeausschnitt initialisiert ein Array `sm` mit einer Länge von `max_num + 1`, wobei jedes Element zunächst auf 0 gesetzt wird. Anschließend wird eine Schleife durchlaufen, die das Array `sm` mit kumulativen Summen füllt. Die Variable `sum` wird dabei verwendet, um die laufende Summe der Elemente des Arrays `n` zu speichern.

Danach wird ein weiteres Array `ns` mit einer Länge von `l` initialisiert, ebenfalls mit allen Elementen auf 0 gesetzt. In einer weiteren Schleife wird jedes Element von `ns` mit dem entsprechenden Wert aus dem Array `sm` belegt, basierend auf den Werten aus dem Array `nums`.

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

Hier ist ein etwas komplexeres Beispiel.

```
```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

 short_n = []
 short_num = [] * 1
 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 = []
        for i in range(len(nums)):
            ns.append(sm[nums[i]])
        return ns

# print(Solution().smallerNumbersThanCurrent([72,48,32,16,10,59,83,38,1,4,68,7,67,16,5,35,99,15,55,11,20]))
```

Übung

- Die Schüler bearbeiten einige Aufgaben ähnlich wie oben.