高盛

8/2/19 21:57

☆ Weekly Stock Prices

In investment analysis, it is often helpful to determine the average of a value over some trailing number of days. This reduces the sensitivity of the value to short term volatility. For a given list of daily stock closing prices, determine the seven day trailing average stock price for every day where there are 7 days of data. Return a list of trailing averages cast as strings, rounded to two places after the decimal.

For example, there are $n = 14$ days of price history, indices from 0 to 13, $dailyPrice = \{1, 1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7\}$. The first day with 7 days worth of data is $dailyPrice[6]$, and the average of the first 7 days' prices is 1.00 as all of the values are 1. The next day, there are 6 days with a price of 1, and 1 day with a price of 7: $(1+1+1+1+1+1+7)/7 = 13/7 = 1.86$. The same analysis is performed for the remaining days, and the resulting string array is $answer = \{'1.00', '1.86', '2.71', '3.57', '4.43', '5.29', '6.14', '7.00'\}$.

Function Description

Complete the function *stringFormattedWeeklyPrices* in the editor below. It should return an array of $n - 6$ floats, each rounded to two decimals and cast as a string.

stringFormattedWeeklyPrices has the following parameter(s):

dailyPrice[dailyPrice[0]..dailyPrice[n-1]]: integer array of length n, such that $dailyPrice[i]$ represents the price of the i^{th} day

Constraints

- $7 \leq n \leq 10^5$
- $1 \leq dailyPrice[i] \leq 100$

▼ Sample Case 0

Sample Input For Custom Testing

```
8
7
8
8
11
9
7
5
6
```

Sample Output

```
7.86
7.71
```

Explanation

For this sample, the output array will contain two strings. The first one will be the average of the first 7 values from dailyPrice which is $(7+8+8+11+9+7+5) / 7 = 7.86$ when rounded to two decimal places. The second element of weeklyPrice is the average price over the span of the next 7 days, which is $(8+8+11+9+7+5+6) / 7 = 7.71$ when rounded to two decimal places.

▼ Sample Case 1

Sample Input For Custom Testing

```
9
5
5
5
5
5
5
5
6
6
```

Sample Output

```
5.00
5.14
5.29
```

Explanation

There are $n=9$ entries, and $dailyPrice = \{5, 5, 5, 5, 5, 5, 5, 6\}$. The average of the first 7 days is clearly 5.00 as the daily price for each of the first 7 days is 5. Remember that the weekly price is a string that always contains two decimal places. The average over the next 7 days is $(5*6 + 6) / 7 = 5.14$ when rounded to two decimal places, and the average over the last 7 days is $(5*5 + 6 + 6) / 7 = 5.29$ when rounded to two decimal places.

☆ A Strange Sorting Problem

Due to a bug in a trading program, the digits of the decimal system got reshuffled. For instance, each 0 changed to 2, each 1 got changed to 3 and each 2 got changed to 0. If the correct number is "021", the system shows "203". The users of the trading software care about the relative values of their positions. Before rolling out the fix for the underlying issue, the company decided to first issue a modified sorting patch that will show the relative order based on the correct values, sorted ascending.

Given the numbers that the program needs to sort and the mapping, i.e. the shuffled version of the decimal digits, return a list of the jumbled numbers sorted by their correct decimal values, ascending. If multiple mapped values are equal, the values returned should be in the original order they were presented.

For example, $mapping = \{3,5,4,6,2,7,9,8,0,1\}$ of fixed length of $m = 10$ and another array of numbers strings, $nums = \{'990','332','32'\}$ of length $n = 3$.

Map $'990'$ as follows:

- The first digit is $'9'$. In the $mapping$ array, 9 is at position 6 so the first digit of the mapped value is $'6'$.
- The second digit is $'9'$. Again, the value is at position 6, so the mapped value is now $'66'$.
- The third digit is $'0'$, found at position 8 of the $mapping$ array. The mapped value is $'668'$ or 668 as an integer.

Map $'332'$ as:

- The first and second digits are both $'3'$ which is found at index 0 in $mapping$. The mapped value is $'00'$.
- The third digit is $'2'$ found at index 4 of $mapping$, so the mapped value is $'004'$ or 4 as an integer.

The value $'32'$ maps to $'04'$, or integer 4, which equals the previous value.

Ordering by integer values yields $[4, 4, 668]$, and retaining order for $'332'$ and $'32'$ results in a return array of associated original values: $['332', '32', '990']$.

Function Description

Complete the function *strangeSort* in the editor below. The function must return an array of strings.

strangeSort has the following parameter(s):

mapping[mapping[0]....mapping[m-1]]: an array of integers, denoting the first 10 digits in the number system.

nums[nums[0]....nums[n-1]]: an array of strings, denoting the array that needs to be sorted.

Constraints

- $m = 10$
- $1 \leq n \leq 10^4$
- $1 \leq length\ of\ each\ nums[i] \leq 100$ (where $0 \leq i < n$)
- *mapping array will contain all the 10 digits*

▼ Sample Case 0

Sample Input For Custom Testing

```
10
2
1
4
8
6
3
0
9
7
5
8
12
02
4
023
65
83
224
50
```

Sample Output

```
4
224
12
83
65
02
50
023
```

Explanation

The conversion of the given numbers into the usual number system is as follows:

Input Numbers	Converted to the usual Number system
4	2
224	002
12	10
83	35
65	49
02	60
50	96
023	605

▼ Sample Case 1

Sample Input For Custom Testing

```
10
0
1
2
3
4
5
6
7
8
9
9
180
84
99
003
53
28
60
070
1
```

Sample Output

```
1
003
28
53
60
070
84
99
180
```

Explanation

The given number system is the usual decimal number system so the numbers map to their integer equivalents.