Codility_

Candidate Report: Anonymous

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MissingInteger ⚠ Python	29 min	100%

Total score

100%

Tasks Details

Medium

1. MissingInteger

Find the smallest positive integer that does not occur in a given sequence.

Task Score	Correctness	Performance
100%	100%	100%

Task description

This is a demo task.

Write a function:

```
def solution(A)
```

that, given an array A of N integers, returns the smallest positive integer (greater than 0) that does not occur in A.

For example, given A = [1, 3, 6, 4, 1, 2], the function should return 5.

Given A = [1, 2, 3], the function should return 4.

Given A = [−1, −3], the function should return 1.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- each element of array A is an integer within the range [−1,000,000..1,000,000].

Solution

Programming language used:	Python
Total time used:	29 minutes
Effective time used:	29 minutes
Notes:	*not defined yet*

Task timeline

11:57:38 12:26:28

Code: 12:26:28 UTC, py, final, score: **100**

show code in pop-up

```
# you can write to stdout for debugging purposes,
# print("this is a debug message")


# <釐清問題>
#   - 找到最小的正整數, 且該正整數未出現於陣列中
#   e.g.
#   A = [1, 3, 6, 4, 1, 2], the function should re

# <其它限制>
# Given A = [1, 2, 3], the function should return
# Given A = [-1, -3], the function should return 1
# N is an integer within the range [1..100,000];
# each element of array A is an integer within the

def solution(A):
    return findMinDisappearInteger(A)


def findMinDisappearInteger(array) -> int:
    non_duplicated_array = list(set(array))

    sorted_array = sorted(non_duplicated_array)

    cnt = 1
    for element in sorted_array:
        if element < 1:
            continue
        else:
            if cnt == element:
                cnt += 1
            else:
                break
    return cnt
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N) or O(N * log(N))**

collapse all **Example tests**

▼ example1 ✓ OK
first example test

1. 0.036 s OK

▼ example2 ✓ OK
second example test

1. 0.040 s OK

▼ example3 ✓ OK
third example test

1. 0.040 s OK

collapse all **Correctness tests**

▼ extreme_single ✓ OK
a single element

1. 0.036 s **OK**
2. 0.036 s **OK**
3. 0.036 s **OK**
4. 0.036 s **OK**

▼ simple ✓ OK
simple test

1. 0.036 s **OK**
2. 0.040 s **OK**
3. 0.036 s **OK**

▼ extreme_min_max_value ✓ OK
minimal and maximal values

1. 0.040 s **OK**
2. 0.036 s **OK**

▼ positive_only ✓ OK
shuffled sequence of 0...100 and then 102...200

1. 0.040 s **OK**
2. 0.040 s **OK**

▼ negative_only ✓ OK
shuffled sequence -100 ... -1

1. 0.036 s **OK**

collapse all **Performance tests**

▼ medium ✓ OK
chaotic sequences length=10005 (with minus)

1. 0.048 s **OK**
2. 0.048 s **OK**
3. 0.052 s **OK**

▼ large_1 ✓ OK
chaotic + sequence 1, 2, ..., 40000 (without minus)

1. 0.136 s **OK**

▼ large_2 ✓ OK
shuffled sequence 1, 2, ..., 100000 (without minus)

1. 0.164 s **OK**
2. 0.148 s **OK**

▼ large_3 ✓ OK
chaotic + many -1, 1, 2, 3 (with minus)

1. 0.140 s **OK**