Codility_

Candidate Report: trainingXGMR8S-PDY

Check out Codility training tasks

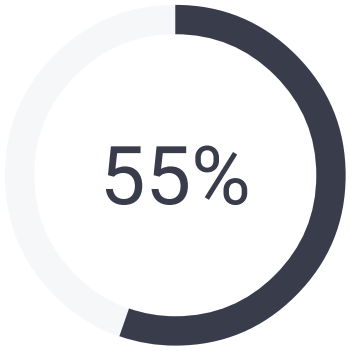
Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
MissingInteger C	32 min	55%

Total score



Tasks Details

Medium

1. MissingInteger

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

Task Score	Correctness	Performance
55%	80%	25%

Task description

This is a demo task.

Write a function:

```
int solution(int A[], int N);
```

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:	C
Total time used:	32 minutes
Effective time used:	32 minutes
Notes:	*not defined yet*

Task timeline

Code: 02:21:31 UTC, c, final, score: **55** show code in pop-up

```c
// you can write to stdout for debugging purposes, e.g.
// printf("this is a debug message\n");

int solution(int A[], int N) {
    // write your code in C99 (gcc 6.2.0)
    int ret = 1;
    int i;

```

```
9     if (N==1) return A[0]==1? 2: 1;
10
11    int rangemap[100000-1] = {0};
12    for (i=0; i<N; i++){
13        if (A[i] < ret)
14            continue;
15        rangemap[A[i]-1] |= 1;
16        if (A[i] == ret)
17            ret = A[i];
18    }
19    for (i=(ret-1); i<(100000-1); i++){
20        //printf("ret=%d, ",i+1);
21        if (!rangemap[i])
22            break;
23        ret = i+1+1;
24    }
25    return ret;
26 }
```

Analysis summary

The following issues have been detected: wrong answers, runtime errors.

For example, for the input `[-1000000, 1000000]` the solution terminated unexpectedly.

Analysis

expand all — Example tests

▶	example1 first example test	✓ OK
▶	example2 second example test	✓ OK
▶	example3 third example test	✓ OK

expand all — Correctness tests

▶	extreme_single a single element	✓ OK
▶	simple simple test	✓ OK
▼	extreme_min_max_value minimal and maximal values	✗ RUNTIME ERROR tested program terminated with exit code 1

1. 0.001 s **RUNTIME ERROR**, tested program terminated with exit code 1

   stderr:
   `Segmentation Fault`

2. 0.001 s **RUNTIME ERROR**, tested program terminated with exit code 1

   stderr:
   `Segmentation Fault`

▶	positive_only shuffled sequence of 0...100 and then 102...200	✓ OK
▶	negative_only shuffled sequence -100 ... -1	✓ OK

collapse all — Performance tests

▼	medium chaotic sequences length=10005 (with minus)	✗ RUNTIME ERROR tested program terminated with exit code 1

1. 0.001 s **OK**
2. 0.001 s **OK**

3. 0.001 s **RUNTIME ERROR**, tested program terminated with exit code 1

stderr:
```
Segmentation Fault
```

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

1. 0.004 s **OK**

▼ large_2 ✗ **WRONG ANSWER**
shuffled sequence 1, 2, ..., 100000 (without minus)
got 100000 expected 100001

1. 0.004 s **WRONG ANSWER**, got 100000 expected 100001
2. 0.008 s **OK**

▼ large_3 ✗ **RUNTIME ERROR**
chaotic + many -1, 1, 2, 3 (with minus)
tested program terminated with exit code 1

1. 0.004 s **RUNTIME ERROR**, tested program terminated with exit code 1

stderr:
```
Segmentation Fault
```