c0d1l1ty

...ck out Codility training tasks

Demo ticket

Session

ID: demo2VRR9T-DXZ
Time limit: 120 min.

Status: closed

Created on: 2014-08-28 02:36 UTC
Started on: 2014-08-28 02:37 UTC
Finished on: 2014-08-28 03:09 UTC

Tasks in test

1 | {} Dominator

Correctness

100%

Performance

100%

Task score

100%

Test score

❓

100%

100 out of 100 points

EASY

1. Dominator

Find an index of an array such that its value occurs at more than half of indices in the array.

score: 100 of 100

Task description

A zero-indexed array A consisting of N integers is given. The *dominator* of array A is the value that occurs in more than half of the elements of A.
For example, consider array A such that

```
A[0] = 3    A[1] = 4    A[2] =  3
A[3] = 2    A[4] = 3    A[5] = -1
A[6] = 3    A[7] = 3
```

The dominator of A is 3 because it occurs in 5 out of 8 elements of A (namely in those with indices 0, 2, 4, 6 and 7) and 5 is more than a half of 8.
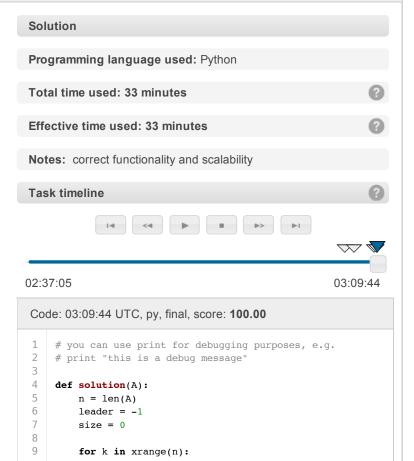Write a function

```
def solution(A)
```

that, given a zero-indexed array A consisting of N integers, returns index of any element of array A in which the dominator of A occurs. The function should return −1 if array A does not have a dominator.
Assume that:

- N is an integer within the range [0..100,000];
- each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].

For example, given array A such that

```
A[0] = 3    A[1] = 4    A[2] =  3
A[3] = 2    A[4] = 3    A[5] = -1
A[6] = 3    A[7] = 3
```

the function may return 0, 2, 4, 6 or 7, as explained above.

Solution

Programming language used: Python

Total time used: 33 minutes ❓

Effective time used: 33 minutes ❓

Notes: correct functionality and scalability

Task timeline ❓

|◄ ◄◄ ► ■ ►► ►|

02:37:05   03:09:44

Code: 03:09:44 UTC, py, final, score: **100.00**

```python
1    # you can use print for debugging purposes, e.g.
2    # print "this is a debug message"
3
4    def solution(A):
5        n = len(A)
6        leader = -1
7        size = 0
8
9        for k in xrange(n):
```

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(1), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

```
10          if size == 0:
11              value = k
12              size += 1
13          else:
14              if A[k] == A[value]:
15                  size += 1
16              else:
17                  size -= 1
18
19      if size == 0:
20          return -1
21      else:
22          candidate = value
23
24      count = 0
25      for k in xrange(n):
26          if A[k] == A[candidate]:
27              count += 1
28
29      if count > n // 2:
30          return candidate
31      else:
32          return -1
```

Analysis

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

test	time	result
Example tests		
example example test	0.064 s	**OK**
Correctness tests		
small_nondominator all different and all the same elements	0.060 s	**OK**
small_half_positions half elements the same, and half + 1 elements the same	0.064 s	**OK**
small small test	0.064 s	**OK**
small_pyramid decreasing and plateau, small	0.064 s	**OK**
extreme_empty_and_single_item empty and single element arrays	0.064 s	**OK**
extreme_half1 array with exactly N/2 values 1, N even + [0,0,1,1,1]	0.064 s	**OK**
extreme_half2 array with exactly floor(N/2) values 1, N odd + [0,0,1,1,1]	0.064 s	**OK**
extreme_half3 array with exactly ceil(N/2) values 1 + [0,0,1,1,1]	0.064 s	**OK**
Performance tests		
medium_pyramid decreasing and plateau, medium	0.080 s	**OK**
large_pyramid decreasing and plateau, large	0.244 s	**OK**
medium_random random test with dominator, N = 10,000	0.084 s	**OK**
large_random random test with dominator, N = 100,000	0.256 s	**OK**

Training center