Demo ticket

Session

ID: demoEM96GH-DRW
Time limit: 120 min.

Status: closed

Created on: 2014-03-17 03:40 UTC
Started on: 2014-03-17 03:40 UTC
Finished on: 2014-03-17 04:16 UTC

Tasks in test

Task score

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

```
int solution(const vector<int> &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.
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).

Solution

Programming language used: C++

Total time used: 37 minutes

Effective time used: 37 minutes

Notes: correct functionality and scalability

Task timeline

03:40:07 — 04:16:32

Code: 04:16:32 UTC, cpp, final, score: **100.00**

```
01. int solution(const vector<int> &A) {
02.     int n = A.size();
03.     int candidate = -1;
04.     int leader_index = -1;
05.     int candidate_count = 0;
06.     int candidate_value = 0;
07.     int candidate_index = -1;
08.
09.     for (int i = 0; i < n; i++) {
10.         if (candidate_count == 0) {
11.             ++candidate_count;
12.             candidate_value = A[i];
13.         } else {
14.             if (candidate_value != A[i])
15.                 --candidate_count;
16.             else
17.                 ++candidate_count;
18.         }
19.     }
20.
```

Elements of input arrays can be modified.


```
21.     // defaults to -1
22.     if (candidate_count > 0)
23.         candidate = candidate_value;
24.
25.     candidate_count = 0;
26.     for (int i = 0; i < n; i++) {
27.         if (A[i] == candidate) {
28.             ++candidate_count;
29.             candidate_index = i;
30.         }
31.     }
32.
33.     // defaults to -1
34.     if (candidate_count > n/2)
35.         leader_index = candidate_index;
36.
37.     return leader_index;
38. }
```

Analysis

Detected time complexity:

O(N) or O(N*log(N))

test	time	result
example example test	0.020 s.	OK
small_nondominator all different and all the same elements	0.020 s.	OK
small_half_positions half elements the same, and half + 1 elements the same	0.020 s.	OK
small small test	0.020 s.	OK
small_pyramid decreasing and plateau, small	0.020 s.	OK
extreme_empty_and_single_item empty and single element arrays	0.020 s.	OK
extreme_half1 array with exactly N/2 values 1, N even + [0,0,1,1,1]	0.020 s.	OK
extreme_half2 array with exactly floor(N/2) values 1, N odd + [0,0,1,1,1]	0.020 s.	OK
extreme_half3 array with exactly ceil(N/2) values 1 + [0,0,1,1,1]	0.020 s.	OK
medium_pyramid decreasing and plateau, medium	0.020 s.	OK
large_pyramid decreasing and plateau, large	0.040 s.	OK
medium_random random test with dominator, N = 10,000	0.020 s.	OK
large_random random test with dominator, N = 100,000	0.050 s.	OK

Training center