codility

Grzegorz Jakacki

Candidate

E-mail: foo@example.com
Last school attended: University of Warsaw (Poland)
Academic degree: Master of Science (MSc)
Field of study: Computer sciences
Profile URL: http://www.codility.com/

Notes:
N/A

Session

ID: 5VM69K-BEF
Time limit: 90 min.
Report recipients: hr@example.com
Accessed from: 198.51.100.0
Invited by: demo@codility.com

Status: closed

Created on: 2014-04-30 08:24 UTC
Started on: 2014-04-30 08:25 UTC
Finished on: 2014-04-30 08:48 UTC

Tasks in test	Correctness	Performance	Task score
1 Equi Submitted in: C	60%	54%	58%
2 BugfixingLeader Submitted in: Python	100%	100%	100%
3 PtrListLen Submitted in: Python	100%	not assessed	100%

Test score

86%

258 out of 300 points

MEDIUM

1. Equi

Find an index in an array such that its prefix sum equals its suffix sum.

score: 58 of 100

Task description

This is a demo task. You can read about this task and its solutions in this blog post.

A zero-indexed array A consisting of N integers is given. An *equilibrium index* of this array is any integer P such that $0 \le P < N$ and the sum of elements of lower indices is equal to the sum of elements of higher indices, i.e.

$$A[0] + A[1] + \ldots + A[P-1] = A[P+1] + \ldots + A[N-2] + A[N-1].$$

Sum of zero elements is assumed to be equal to 0. This can happen if $P = 0$ or if $P = N-1$.

For example, consider the following array A consisting of N = 8 elements:

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

P = 1 is an equilibrium index of this array, because:

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

P = 3 is an equilibrium index of this array, because:

- $A[0] + A[1] + A[2] = -2 = A[4] + A[5] + A[6] + A[7]$

P = 7 is also an equilibrium index, because:

- $A[0] + A[1] + A[2] + A[3] + A[4] + A[5] + A[6] = 0$

and there are no elements with indices greater than 7.

P = 8 is not an equilibrium index, because it does not fulfill the condition $0 \le P < N$.

Write a function:

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

that, given a zero-indexed array A consisting of N integers, returns any of its equilibrium indices. The function should return −1 if no equilibrium index exists.

For example, given array A shown above, the function may return 1, 3 or 7, as explained above.

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].

Complexity:

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

Elements of input arrays can be modified.

Solution

Programming language used: C

Total time used: 24 minutes

Effective time used: 15 minutes

Notes: *not defined yet*

Source code

Code: 08:48:51 UTC, c, final,
score: **58.00**

```c
int solution(int A[], int N) {
    // return 3 for [3,2,-5,1]
    int sum_prefix = 0;
    int sum_suffix = 0;
int i, j;
for (i = 0; i < N; i++) {
    // Check if is is the equilibrium point
    sum_prefix = 0;
    for (j = 0; j < i; j++) {
       sum_prefix += A[j];
    }
    sum_suffix = 0;
    for (j = i+1; j < N; j++) {
        sum_suffix += A[j];
       // Compute suffix sum
    }
    if (sum_prefix == sum_suffix)
        return i;
}
return -1;
}
```

Analysis summary

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

Analysis

Example tests		
example Test from the task description	✔	**OK**
Correctness tests		
simple	✔	**OK**
extreme_large_numbers Sequence with extremely large numbers testing arithmetic overflow.	✘	**WRONG ANSWER** got 2, but it is not equilibrium point, sum[0..1]=4294967294, sum[3..3]=-2
extreme_negative_numbers Sequence with extremely large numbers testing arithmetic overflow.	✘	**WRONG ANSWER** got 0, but it is not equilibrium point, left sum (empty set)=0, sum[1..2]=-4294967296
overflow_tests1 arithmetic overflow tests	✘	**WRONG ANSWER** got 0, but it is not equilibrium point, left sum (empty set)=0, sum[1..2]=-4294967296
overflow_tests2 arithmetic overflow tests	✘	**WRONG ANSWER** got 2, but it is not equilibrium point, sum[0..1]=-4294967296, right sum (empty set)=0
one_large	✔	**OK**

	one large number at the end of the sequence	
	sum_0 sequence with sum=0	✔ **OK**
	single_empty single number or empty array	✔ **OK**
	combinations_of_two multiple runs, all pairs of values: -1, 0 and 1	✔ **OK**
	combinations_of_three multiple runs, all triples of values -1, 0 and 1	✔ **OK**
	small_pyramid	✔ **OK**
Correctness/performance tests		
	extreme_max Maximal size test	✘ **TIMEOUT ERROR** running time: 1.35 sec., time limit: 0.10 sec.
Performance tests		
	large_long_sequence_of_ones	✘ **TIMEOUT ERROR** running time: 2.51 sec., time limit: 0.10 sec.
	large_long_sequence_of_minus_ones	✘ **TIMEOUT ERROR** running time: 5.02 sec., time limit: 0.10 sec.
	medium_pyramid	✔ **OK**
	large_pyramid Large performance test, O(n^2) solutions should fail.	✔ **OK**
	huge_pyramid Large performance test, O(n^2) solutions should fail.	✔ **OK**

MEDIUM

2. BugfixingLeader

Find and correct bugs in a function that finds a value that occurs in more than half of the elements of an array.

score: 100 of 100

Task description

A non-empty zero-indexed array A consisting of N integers is given. The *leader* of this array is the value that occurs in more than half of the elements of A.

You are given an implementation of a function:

```
def solution(A)
```

that, given a non-empty zero-indexed array A consisting of N integers, returns the leader of array A. The function should return −1 if array A does not contain a leader.

For example, given array A consisting of ten elements such that:

```
A[0] = 4
A[1] = 2
A[2] = 2
A[3] = 3
A[4] = 2
A[5] = 4
A[6] = 2
A[7] = 2
A[8] = 6
A[9] = 4
```

the function should return −1, because the value that occurs most frequently in the array, 2, occurs five times, and 5 is not more than half of 10.

Given array A consisting of five elements such that:

```
A[0] = 1
A[1] = 1
A[2] = 1
A[3] = 50
A[4] = 1
```

the function should return 1.

Unfortunately, there is a bug in the implementation. Find it and correct it. You should modify at most **three** lines of code.

Assume that:

- N is an integer within the range [1..1,000,000];
- each element of array A is an integer within the range [0..2147483647].

Complexity:

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

Elements of input arrays can be modified.

Solution

Programming language used: Python

Total time used: 6 minutes

Effective time used: 6 minutes

Notes: *not defined yet*

Source code

Code: 08:36:49 UTC, py, final,
score: **100.00**

```
1   def solution(A):
2       n = len(A)
3       L = [-1] + A
4       L.sort()
5       count = 0
6       pos = (n + 1) // 2
7       candidate = L[pos]
8       for i in xrange(1, n + 1):
9           if (L[i] == candidate):
10              count = count + 1
11  -   if (count > pos):
    +   if (2*count > n):
12          return candidate
13      return -1
```

Analysis summary

The solution obtained perfect score.

Analysis

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

Example tests	
example1	✔ OK
example2	✔ OK
Correctness tests	
simple1	✔ OK
simple2	✔ OK
single	✔ OK
simple_random	✔ OK
extreme_big_values	✔ OK
medium_1	✔ OK
medium_2	✔ OK
Performance tests	
medium_3	✔ OK
medium_4	✔ OK
medium_5	✔ OK
large_1	✔ OK
large_2	✔ OK
large_3	✔ OK
large_4	✔ OK
large_random1	✔ OK
large_random2	✔ OK
large_inc1	✔ OK
large_inc2	✔ OK
large_range1	✔ OK
large_range2	✔ OK
large_range3	✔ OK

EASY

3. PtrListLen

Compute the length of single-link list without a cycle.

score: 100 of 100

Task description

A pointer is called a *linked list* if:

- it is an empty pointer (it is then called a *terminator* or an *empty list*); or
- it points to a structure (called a *node* or the *head*) that contains a value and a linked list (called the *tail*).

The *length* of a list is defined as the total number of nodes it contains. In particular, an empty list has length 0.

For example, consider the following linked list:

```
A -> B -> C -> D ->
```

This list contains four nodes: A, B, C and D. Node D is the last node and its tail is the terminator. The length of this list is 4.

Assume that the following declarations are given:

```
class IntList(object):
  value = 0
  next = None
```

Write a function:

```
def solution(L)
```

that, given a non-empty linked list L consisting of N nodes, returns its length.

For example, given list L shown in the example above, the function should return 4.

Assume that:

- N is an integer within the range [1..5,000];
- list L does not have a cycle (each non-empty pointer points to a different structure).

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

Solution

Programming language used: Python

Total time used: 4 minutes

Effective time used: 4 minutes

Notes: *not defined yet*

Source code

Code: 08:40:26 UTC, py, final,
score: **100.00**

```
1 def solution(L):
2     count = 0
3     while L:
4         count += 1
5         L = L.next
6     return count
```

Analysis summary

The solution obtained perfect score.

Analysis

Example tests	
example example, length=4	✔ OK
Correctness tests	
extreme_single_double length=1	✔ OK
three_elems length=3	✔ OK
twenty_elements length=20	✔ OK
medium length=93	✔ OK
medium2 length=999	✔ OK
1k_elements length=1,000	✔ OK
quite_long length=4,000	✔ OK
long length=5,000	✔ OK