codility

Rob Starbuck

Candidate

E-mail:
Last school attended: University Of Sunderland
Academic degree: Bachelor of Arts (BA)
Field of study: N/A
Programming experience: 10
Profile URL:

Notes:
N/A

Session

ID: 6PNTT7-Y4T
Time limit: 120 min.
Report recipients: santiago.castineira@mvs-corp.com
Accessed from: 82.44.143.26
Invited by: santiago.castineira@mvs-corp.com

Status: closed

Created on: 2015-08-21 09:48 UTC
Started on: 2015-08-21 11:05 UTC
Finished on: 2015-08-21 12:42 UTC

Tasks in test	Correctness	Performance	Task score
1 DecReprSenior Submitted in: JavaScript	100%	100%	100%
2 ArrListLen Submitted in: JavaScript	75%	not assessed	75%
3 IntegerReductionCount Submitted in: JavaScript	71%	0%	50%

Test score

75%

225 out of 300 points

EASY

1. DecReprSenior

Find the largest value obtained by permutations of digits.

score: 100 of 100

Task description

Two non-negative integers are called *siblings* if they can be obtained from each other by rearranging the digits of their decimal representations. For example, 123 and 213 are siblings. 535 and 355 are also siblings.

A set consisting of a non-negative integer N and all of its siblings is called the *family* of N. For example, the family of 553 comprises three numbers: 355, 535 and 553.

Write a function:

```
function solution(N);
```

that, given a non-negative integer N, returns the largest number in the family of N. The function should return −1 if the result exceeds 100,000,000.

For example, given N = 213 the function should return 321. Given N = 553 the function should return 553.

Assume that:

- N is an integer within the range [0..2,147,483,647].

Complexity:

- expected worst-case time complexity is O(1);
- expected worst-case space complexity is O(1).

Solution

Programming language used: JavaScript

Total time used: 97 minutes

Effective time used: 13 minutes

Notes: *not defined yet*

Source code

Code: 12:42:02 UTC, js, final, score: **100.00**

```
// you can use console.log for debugging purposes, i.e.
// console.log('this is a debug message');

function solution(N) {
    // write your code in JavaScript (Node.js 0.12)
    // I'm going to make N a string and then split it into an array and sort it

    var result;
    var strN = String(N);
    var arrN = strN.split('');

    arrN.sort().reverse();

    result = Number(arrN.join(''));

    if(result > 100000000){
        return -1;
    }

    return result;
}
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity:

O(1)

Example tests		
example1 example test, N = 213	✔	**OK**
example2 example test, N = 553	✔	**OK**
Correctness tests		
simple1 N = 1234	✔	**OK**
simple2 N = 1001	✔	**OK**
simple3 N = 987,659	✔	**OK**
extreme_one_digit one digit number	✔	**OK**
Performance tests		
extreme_large extreme large numbers	✔	**OK**
limit N = 100,000,000	✔	**OK**
medium1	✔	**OK**

	N = 10,000,001	
	medium2 N = 10,101,010	✔ **OK**

EASY

2. ArrListLen

Compute length of a single-link list encoded in an array.

score: 75 of 100

Task description

A non-empty zero-indexed array A consisting of N integers is given.

Array A represents a linked list. A list is constructed from this array as follows:

- the first node (the head) is located at index 0;
- the value of a node located at index K is A[K];
- if the value of a node is −1 then it is the last node of the list;
- otherwise, the successor of a node located at index K is located at index A[K] (you can assume that A[K] is a valid index, that is 0 ≤ A[K] < N).

For example, for array A such that:

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

the following list is constructed:

- the first node (the head) is located at index 0 and has a value of 1;
- the second node is located at index 1 and has a value of 4;
- the third node is located at index 4 and has a value of 2;
- the fourth node is located at index 2 and has a value of −1.

Write a function:

```
function solution(A);
```

that, given a non-empty zero-indexed array A consisting of N integers, returns the length of the list constructed from A in the above manner.

For example, given array A such that:

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

the function should return 4, as explained in the example above.

Assume that:

- N is an integer within the range [1..200,000];
- each element of array A is an integer within the range [−1..N−1];
- it will always be possible to construct the list and its length will be finite.

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

Solution

Programming language used: JavaScript

Total time used: 88 minutes

Effective time used: 35 minutes

Notes: *not defined yet*

Source code

Code: 12:42:20 UTC, js, final, score: **75.00**

```
// you can use console.log for debugging purposes, i.e.
// console.log('this is a debug message');

function solution(A) {

    // I think I understand this now, I'm not sure that I like the wording
    // Recursive function could be the best way to go again

    var i = 0;

    var recurse = function(argInt,argArray){

        i++;

        if(argInt < 0){
            return i;
        }

        return recurse(argArray[argInt],argArray);
    };

    return recurse(A[0],A);

}
```

Analysis summary

The following issues have been detected: runtime errors.

Analysis

Example tests		
example example test [1,4,-1,3,2]	✔	**OK**
Correctness tests		
extreme_single one/two elements	✔	**OK**
extreme_other_values extreme values outside the list	✔	**OK**
smal_functional functional tests	✔	**OK**
some_zero_elems array [2,-1,1,0,0,0]	✔	**OK**
quite_long 4000 elements used, increasing	✔	**OK**
long_reverse 5001 elements used, left-right	✔	**OK**
very_long 10**5 elements used, decreasing	✘	**RUNTIME ERROR** tested program terminated unexpectedly
max_size max size, left-right	✘	**RUNTIME ERROR** tested program terminated unexpectedly

MEDIUM

3. IntegerReductionCount

Given a number N and two reduction operations, count the number of operations reducing N to 0.

score: 50 of 100

Task description

A non-negative integer variable V is given. There are two actions available that modify its value:

- if V is odd, subtract 1 from it;
- if V is even, divide it by 2.

These actions are performed until the value of V becomes 0.

For example, if V initially contains value 28, it will become 0 after seven steps:

- V contains value 28, which is even: divide by 2 and obtain 14;
- V contains value 14, which is even: divide by 2 and obtain 7;
- V contains value 7, which is odd: subtract 1 and obtain 6;
- V contains value 6, which is even: divide by 2 and obtain 3;
- V contains value 3, which is odd: subtract 1 and obtain 2;
- V contains value 2, which is even: divide by 2 and obtain 1;
- V contains value 1, which is odd: subtract 1 and obtain 0.

Write a function:

```
function solution(S);
```

that, given a zero-indexed string S consisting of N characters containing a binary representation of the initial value of variable V, returns the number of steps after which the value of V will become 0, as described above.

Assume that:

- N is an integer within the range [1..1,000,000];
- string S consists only of the characters "0" and/or "1";
- the binary representation is big-endian, i.e. the first character of string S corresponds to the most significant bit;
- the binary representation may contain leading zeros.

For example, given string S = "011100" the function should return 7, because string S represents the number 28 and 28 becomes 0 after seven steps, as explained above.

Complexity:

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

Solution

Programming language used: JavaScript

Total time used: 89 minutes

Effective time used: 50 minutes

Notes: *not defined yet*

Source code

Code: 12:42:39 UTC, js, final, score: **50.00**

```
// you can use console.log for debugging purposes, i.e.
// console.log('this is a debug message');

function solution(S) {
    // write your code in JavaScript (Node.js 0.12)
    var i = 0;
    var number = parseInt(S,2);

    var reduceTo0 = function(intArg){

        var current = intArg;

        if(!current){
            //The last modification is already factored in
            return i;
        }

        if(current % 2 === 0){
            current = current / 2;
        }else{
            --current;
        }

        i++;

        //In fact I had simply forgotten to return the end result.
        return reduceTo0(current);

    }

    // N is within a range small enough that the maximum call stack would not be exceeded
    // Should it be I understand that a trampoline could be used to get around this problem

    return reduceTo0(number);

}
```

Analysis summary

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

Analysis

Example tests	
example	✔ OK
Correctness tests	
simple1	✔ OK
simple2	✔ OK
simple3	✔ OK
simple4	✔ OK

extreme_n0	✔ **OK**
medium1	✘ **WRONG ANSWER** got 425 expected 598
medium2	✘ **RUNTIME ERROR** tested program terminated unexpectedly
Performance tests	
big1	✘ **RUNTIME ERROR** tested program terminated unexpectedly
big2	✘ **RUNTIME ERROR** tested program terminated unexpectedly
big_ones	✘ **RUNTIME ERROR** tested program terminated unexpectedly