Codility

Exercises  
12/22/2015

#### Background

Codility is a tool used to measure programming capabilities of given specialist giving usually three selected tasks which involve creating code that meets **running time** and **space** requirements in [**Big O**](http://en.wikipedia.org/wiki/Big_O_notation) notation within short amount of time. Tasks can be quite challenging but give it your best shot. You might want to get something working first then refine the solution. Don’t cheat. It is a challenge for yourself.

**Ignore this piece but this is how this exercise would work in codility if you took the exercise in codility.** A test will typically have 3 – 30 minute tasks to be performed by the developer. Each task is scored on a 100 scale range. If your code does not execute, the task will score 0 points. Once you start the test you must complete in that session as the exercise is timed.

#### Exercises:

#### Perm Missing Element

A zero-indexed array A consisting of N different integers is given. The array contains integers in the range [1..(N + 1)], which means that exactly one element is missing. Your goal is to find that missing element.

**Write a function:**

class Solution { public int solution(int[] A); **}**

that, given a zero-indexed array A, returns the value of the missing element.

For example, given array A such that:

A[0] = 2

A[1] = 3

A[2] = 1

A[3] = 5

the function should return 4, as it is the missing element.

**Assume that:**

* N is an integer within the range [0..100,000];
* the elements of A are all distinct;
* each element of array A is an integer within the range [1..(N + 1)].

**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.

#### Tape Equilibrium

A non-empty zero-indexed array A consisting of N integers is given. Array A represents numbers on a tape.  
Any integer P, such that 0 < P < N, splits this tape into two non−empty parts: A[0], A[1], …, A[P − 1] and A[P], A[P + 1], …, A[N − 1].

The difference between the two parts is the value of: |(A[0] + A[1] + … + A[P − 1]) − (A[P] + A[P + 1] + … + A[N − 1])|  
In other words, it is the **absolute** difference between the sum of the first part and the sum of the second part.

**Write a function:**

Write a function that, given a non-empty zero-indexed array A of N integers, returns the minimal difference that can be achieved.

Example: A[0] = 3 A[1] = 1 A[2] = 2 A[3] = 4 A[4] = 3

We can split this tape in four places:  
P = 1, difference = |3 − 10| = 7  
P = 2, difference = |4 − 9| = 5  
P = 3, difference = |6 − 7| = 1  
P = 4, difference = |10 − 3| = 7  
In this case I would return 1 as it is the smallest difference.

**Assume that:**

N is an int, range [2..100,000];

each element of A is an int,

range [−1,000..1,000].

**Complexity:**

It needs to be O(n) time complexity