Codility_

Paras Mittal

Candidate

E-mail:
mittal.1807@gmail.com

Session

ID: BDKPPV-MB3
Time limit: 100 min.
Report recipients: No one
Accessed from: 217.95.79.196, 217.95.79.196, 217.95.79.196
Invited by: cristiana.martins@zalando.de

Status: completed

Invitation: sent
Created on: 2020-07-13 10:47 UTC
Started on: 2020-07-13 20:34 UTC
Finished on: 2020-07-13 21:41 UTC

Notes:
N/A

Similarity Check

Status: not found
No similar solutions have been detected.

Tasks in test	Score
1 NextSameDigitsSum Submitted in: Java 8	100%
2 FairSplits Submitted in: Java 8	100%
3 BoardRecovery Submitted in: Java 8	100%

Test score

100%

Tasks Details

Easy

1. NextSameDigitsSum

Given an integer N, find the smallest integer greater than N whose digits add up to the value of the sum of the digits of N.

Task Score	Correctness	Performance
100	100	Not assessed

Task description

Write a function:

```
class Solution { public int solution(int N); }
```

which, given an integer N, returns the smallest integer that is greater than N and the sum of whose digits is equal to the sum of the digits of N.

Examples:

1. Given N = 28, your function should return 37. The sum of the digits of 28 is equal to 2 + 8 = 10. The subsequent numbers are (with the sum of their digits in brackets): 29 (11), 30 (3), 31 (4), 32 (5), 33 (6), 34 (7), 35 (8), 36 (9) and 37 (10). 37 is the smallest number bigger than 28 whose digits add up to 10.

2. Given N = 734, your function should return 743. The sum of the digits of 734 and 743 are equal 7 + 3 + 4 = 7 + 4 + 3 = 14. No other integer between 735 and 742 adds up to 14.

3. Given N = 1990, your function should return 2089. The sum of the digits of both numbers is equal to 19 and there is no other integer between them with the same sum of digits.

4. Given N = 1000, your function should return 10000. The sum of the digits of both numbers is equal to 1 and there is no other integer between them with the same sum of digits.

Assume that:

- N is an integer within the range [1..50,000].

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

Solution

See Live Version

Programming language used:	Java 8
Total time used:	11 minutes
Effective time used:	11 minutes
Notes:	*not defined yet*

Source code

Code: 20:45:07 UTC, java, final, score: **100**

```java
// you can also use imports, for example:
// import java.util.*;

// you can write to stdout for debugging purposes, e.g.
// System.out.println("this is a debug message");

class Solution {
    public int solution(int N) {
        // write your code in Java SE 8
        long sumN = getSumOfDigits(N);
        // System.out.println("sum N "+ sumN);
        int result = N + 1;
        while(sumN != getSumOfDigits(result)) {
            result ++;
        }
        return result;
```

```
    }

    private long getSumOfDigits(int n) {
        long sum = 0;
        while(n != 0) {
            sum += n % 10;
            n/= 10;
        }
        return sum;
    }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

Example tests	
example1 First example test.	✔ OK
example2 Second example test.	✔ OK
example3 Third example test.	✔ OK
example4 Fourth example test.	✔ OK
Correctness tests	
fully_random_small Small random tests. N <= 100.	✔ OK
biggest_difference_small Small tests with the biggest possible difference between output and input. N <= 100.	✔ OK
one_digit_number One digit numbers.	✔ OK
fully_random_medium Medium random tests. N <= 1,000.	✔ OK
biggest_difference_medium Medium tests with the biggest possible difference between output and input. N <= 1,000.	✔ OK
fully_random_large Large random tests. N <= 50,000.	✔ OK
biggest_difference_large Large tests with the biggest possible difference between output and input. N <= 50,000.	✔ OK
suffix90 Numbers ending with a sequence of nines followed by zeroes.	✔ OK
suffix9 Numbers ending with a sequence of nines.	✔ OK
suffix0 Numbers ending with a sequence of zeroes.	✔ OK
suffix10 Numbers ending with a one followed by zeroes.	✔ OK
suffix9d0 Numbers ending with a sequence of nines and zeroes separated by a single digit.	✔ OK
powers_of_10 Powers of 10. N <= 10,000.	✔ OK
output_exceeds_5e4 Tests in which outputs are greater than 50,000.	✔ OK

Hard

2. FairSplits

Count the number of ways in which we can split a given two-row array into four parts with the same sum.

Task Score	Correctness	Performance
100	100	100

Task description

You are given two arrays A and B consisting of N integers each.

Index K is named *fair* if the four sums (A[0] + ... + A[K-1]), (A[K] + ... + A[N-1]), (B[0] + ... + B[K-1]) and (B[K] + ... + B[N-1]) are all equal. In other words, K is the index where the two arrays, A and B, can be split (into two non-empty arrays each) in such a way that the sums of the resulting arrays' elements are equal.

For example, given arrays A = [4, -1, 0, 3] and B = [-2, 5, 0, 3], index K = 2 is *fair*. The sums of the subarrays are all equal: 4 + (-1) = 3; 0 + 3 = 3; -2 + 5 = 3 and 0 + 3 = 3. On the other hand, index K = 1 is not *fair*; the sums of the subarrays are: 4; (-1) + 0 + 3 = 2; -2 and 5 + 0 + 3 = 8.

Write a function:

```
class Solution { public int solution(int[] A, int[] B); }
```

which, given two arrays of integers A and B, returns the number of *fair* indexes.

Examples:

1. Given A = [4, -1, 0, 3] and B = [-2, 5, 0, 3], your function should return 2. The *fair* indexes are 2 and 3. In both cases, the sums of elements of the subarrays are equal to 3.

2. Given A = [2, -2, -3, 3] and B = [0, 0, 4, -4], your function should return 1. The only *fair* index is 2. Index 4 is not *fair* as the subarrays containing indexes from K to N - 1 would be empty.

3. Given A = [4, -1, 0, 3] and B = [-2, 6, 0, 4], your function should return 0. There are no *fair* indexes.

4. Given A = [3, 2, 6] and B = [4, 1, 6], your function should return 0.

5. Given A = [1, 4, 2, -2, 5], B = [7, -2, -2, 2, 5], your function should return 2. The *fair* indexes are 2 and 4.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [2..100,000];
- each element of arrays A, B is an integer within the range [−1,000,000,000..1,000,000,000].

Solution

See Live Version

Programming language used:	Java 8
Total time used:	21 minutes
Effective time used:	21 minutes
Notes:	*not defined yet*

Source code

Code: 21:06:03 UTC, java, final, score: **100**

```
1  // you can also use imports, for example:
2  // import java.util.*;
3
4  // you can write to stdout for debugging purposes, e.g.
5  // System.out.println("this is a debug message");
6
7  class Solution {
8      public int solution(int[] A, int[] B) {
9          // write your code in Java SE 8
10         long sumA = 0, sumB = 0;
11         for (int i = 0; i < A.length; i ++) {
```

```
12              sumA += A[i];
13              sumB += B[i];
14          }
15
16          int fairIndex = 0;
17
18
19          long leftSumA = A[0];
20          long leftSumB = B[0];
21          for (int i = 1; i < A.length; i ++) {
22              long rightSumA = sumA - leftSumA;
23              long rightSumB = sumB - leftSumB;
24              if (leftSumA == leftSumB
25                  && leftSumA == rightSumA
26                  && leftSumB == rightSumB) {
27                  fairIndex ++;
28              }
29              leftSumA += A[i];
30              leftSumB += B[i];
31          }
32          return fairIndex;
33      }
34  }
```

Analysis summary

The solution obtained perfect score.

Analysis

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

Example tests	
example1 First example test.	✔ OK
example2 Second example test.	✔ OK
example3 Third example test.	✔ OK
example4 Fourth example test.	✔ OK
example5 Fifth example test.	✔ OK
Correctness tests	
simple_test Simple tests. N = 4.	✔ OK
corner_test Minimum size. N = 2.	✔ OK
big_sum The sum of the values exceeds integer range. N <= 20.	✔ OK
identical_values All values in each of the arrays are identical. N <= 20.	✔ OK
small_random Small random test. N = 200.	✔ OK
Correctness/performance tests	
medium_random Medium random test. N = 500.	✔ OK
Performance tests	
big_simple Big test. N = 100,000.	✔ OK

big_identical_values All the values are identical. N <= 100,000.	✔ OK
big_random Big random test. N = 100,000.	✔ OK
big_random_big_result Big random test with big results. N = 100,000.	✔ OK

Hard

3. BoardRecovery

Recover a 2xN matrix containing only zeroes and ones, knowing only the sum of elements in every row and every column.

Task Score	Correctness	Performance
100	100	100

Task description

There is a board with 2 rows and N columns, represented by a matrix M. Rows are numbered from 0 to 1 from top to bottom and columns are numbered from 0 to N−1 from left to right. Each cell contains either a 0 or a 1. You know that:

- the sum of integers in the 0-th (upper) row is equal to U,
- the sum of integers in the 1-st (lower) row is equal to L,
- the sum of integers in the K-th column is equal to C[K].

Your job is to recover M based on this information.

Write a function:

```
class Solution { public String solution(int U, int L, int[] C); }
```

that, given two integers U, L and an array C of N integers, as described above, returns a string describing the matrix M in the following format. The first part of the string should be the description of the upper row (N characters: `0` or `1`), then there should be comma (`,`), and finally there should be the description of the lower row (N characters: `0` or `1`.) The output string should not contain any whitespace.

If there exist multiple valid Ms, your function may return any one of them. If no valid M exists, your function should return the word `IMPOSSIBLE`.

Examples:

1. Given U = 3, L = 2, C = [2, 1, 1, 0, 1], your function may, for example, return `11001,10100` which describes the following board:

	0	1	2	3	4
0	1	1	0	0	1
1	1	0	1	0	0

2. Given U = 2, L = 3, C = [0, 0, 1, 1, 2], your function should return the word `IMPOSSIBLE`, because no matrix M satisfies such conditions.

3. Given U = 2, L = 2, C = [2, 0, 2, 0], your function should return `1010,1010`, which describes the following board:

	0	1	2	3
0	1	0	1	0
1	1	0	1	0

Write an **efficient** algorithm for the following assumptions:

- U and L are integers within the range [0..100,000];
- N is an integer within the range [1..100,000];
- each element of array C is an integer within the range [0..2].

Solution

See Live Version

Programming language used:	Java 8
Total time used:	36 minutes
Effective time used:	36 minutes
Notes:	*not defined yet*

Source code

Code: 21:41:49 UTC, java, final, score: **100**

```java
// you can also use imports, for example:
// import java.util.*;

// you can write to stdout for debugging purposes, e.g.
// System.out.println("this is a debug message");

class Solution {
    public String solution(int U, int L, int[] C) {
        // write your code in Java SE 8
        int length = C.length;
        StringBuilder uBuilder = new StringBuilder(length);
        StringBuilder lBuilder = new StringBuilder(length);
        for (int i = 0; i < length; i++) {
            if (C[i] == 2) {
                // both must have 1
                uBuilder.append('1');
                lBuilder.append('1');
                U --;
                L --;
            } else if (C[i] == 1) {
                // lets assign to one with greater of two
                if (U > L) {
                    uBuilder.append('1');
                    lBuilder.append('0');
                    U--;
                } else {
                    uBuilder.append('0');
                    lBuilder.append('1');
                    L--;
                }
            } else {
                uBuilder.append('0');
                lBuilder.append('0');
            }
            if (U < 0 || L < 0) {
                return "IMPOSSIBLE";
            }
        }
        if (U != 0 || L != 0) {
            return "IMPOSSIBLE";
        }
        return uBuilder.toString() + "," + lBuilder.toString();
    }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

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

Example tests	
example1 First example test.	✔ OK
example2 Second example test.	✔ OK
example3 Second example test.	✔ OK
Correctness tests	
possible Small hand-written tests, N <= 5, at least one valid matrix M.	✔ OK

small_impossible Small hand-written tests, N <= 5, with answer "IMPOSSIBLE".	✔ OK
random_small_always_possible Small random tests, there is always a valid matrix M, N <= 20.	✔ OK
random_small_one_attraction Small random tests, elements in C always equal to 1, N <= 20.	✔ OK
random_small_both_open_or_both_closed Small random tests, elements in C always equal to 0 or 2, N <= 20.	✔ OK
r_plus_c_larger_than_sum_of_d Small tests with no valid answer, where U + L > sum(C).	✔ OK
r_plus_c_smaller_than_sum_of_d Small tests with no valid answer, where U + L < sum(C).	✔ OK
Correctness/performance tests	
random_medium_always_possible Medium random tests, there is always a valid answer, N <= 1000.	✔ OK
random_medium_one_attraction Medium random tests, values in C biased towards 1, N <= 10 000.	✔ OK
random_medium_both_open_or_both_closed Medium random tests, values in C biased towards 0 and 2, N <= 10 000.	✔ OK
medium_impossible Medium tests with answer "IMPOSSIBLE".	✔ OK
one_attraction_only_open_if_d_equals_2 Tests where, in all valid matrices M, one row is a subset of the other.	✔ OK
Performance tests	
big_random_tests Big random tests, N <= 100 000.	✔ OK
large_impossible Large tests where no valid matrix M exists.	✔ OK
maximal_tests Maximal tests, N <= 100 000.	✔ OK