

Congratulations

You have completed a Codility training test.

[Sign up for our newsletter!](#)
[Like us on Facebook!](#)

Training ticket

Session

ID: training2ESWNV-XF2
Time limit: 120 min.

Status: closed

Created on: 2016-06-03 22:18 UTC
Started on: 2016-06-03 22:22 UTC
Finished on: 2016-06-03 22:37 UTC

Tasks in test

1 **BinaryGap**
Submitted in: Java

Correctness

80%

Performance

not assessed

Task score

80%

80%

80 out of 100 points

EASY

1. BinaryGap

Find longest sequence of zeros in binary representation of an integer.

score: 80 of 100



Task description

A *binary gap* within a positive integer N is any maximal sequence of consecutive zeros that is surrounded by ones at both ends in the binary representation of N .

For example, number 9 has binary representation 1001 and contains a binary gap of length 2. The number 529 has binary representation 1000010001 and contains two binary gaps: one of length 4 and one of length 3. The number 20 has binary representation 10100 and contains one binary gap of length 1. The number 15 has binary representation 1111 and has no binary gaps.

Write a function:

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

that, given a positive integer N , returns the length of its longest binary gap. The function should return 0 if N doesn't contain a binary gap.

For example, given $N = 1041$ the function should return 5, because N has binary representation 10000010001 and so its longest binary gap is of length 5.

Assume that:

- N is an integer within the range $[1..2,147,483,647]$.

Complexity:

- expected worst-case time complexity is $O(\log(N))$;
- expected worst-case space complexity is $O(1)$.

Copyright 2009–2016 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Solution

Programming language used: Java

Total time used: 15 minutes

2

Effective time used: 15 minutes

2

Notes: *not defined yet*

Task timeline

2

22:22:27

22:37:26

Code: 22:37:26 UTC, java, final,
score: 80

[show code in pop-up](#)

```
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 N) {
9         char[] SOURCE = Integer.toBinaryString(N).1
10
11         int BIGGEST = 0; // lenght of the sequence
12         int INDEX = 0; // start index of longest sequence
13         int START = 0; // start index of the current sequ
14         int[] LENGHTS = new int[SOURCE.length]; //
15         boolean prior = false;
```

```
16
17     for (int i=0;i < SOURCE.length; i++) {
18         if (SOURCE[i] == '0') {
19             if (!prior) {
20                 START = i;
21             }
22             LENGHTS[START] = LENGHTS[START] + 1;
23             prior = true;
24         } else {
25             if (prior) {
26                 if (LENGHTS[START] > BIGGEST) {
27                     BIGGEST = LENGHTS[START];
28                     INDEX = START;
29                 }
30             }
31             prior = false;
32         }
33     }
34
35     if (prior) {
36         if (LENGHTS[START] > BIGGEST) {
37             BIGGEST = LENGHTS[START];
38             INDEX = START;
39         }
40     }
41
42     //System.out.println("BV: "+BIGGEST+" at IDX: "+INI
43     return BIGGEST;
44 }
45 }
```

Analysis summary

The following issues have been detected: wrong answers.

Analysis



Example tests	
▶ example1	✓ OK
example test n=1041=10000010001_2	
▶ example2	✓ OK
example test n=15=1111_2	
Correctness tests	
▶ extremes	✓ OK
n=1, n=5=101_2 and n=2147483647=2**31-1	
▶ trailing_zeroes	✗ WRONG ANSWER
n=6=110_2 and n=328=101001000_2 got 1 expected 0	
▶ power_of_2	✗ WRONG ANSWER
n=5=101_2, n=16=2**4 and n=1024=2**10 got 4 expected 0	
▶ simple1	✓ OK
n=9=1001_2 and n=11=1011_2	
▶ simple2	✓ OK
n=19=10011 and n=42=101010_2	
▶ simple3	✓ OK
n=1162=10010001010_2 and n=5=101_2	
▶ medium1	✗ WRONG ANSWER
n=51712=110010100000000_2 and n=20=10100_2 got 9 expected 2	
▶ medium2	✓ OK
n=561892=10001001001011100100_2 and n=9=1001_2	
▶ medium3	✓ OK
n=66561=10000010000000001_2	
▶ large1	✓ OK
n=6291457=110000000000000000001_2	
▶ large2	✓ OK
n=74901729=1000111011011101000111000 01	
▶ large3	✓ OK
n=805306373=1100000000000000000000 000101_2	
▶ large4	✓ OK
n=1376796946=10100100001000001000001	

	00010010_2	
▶	large5 n=1073741825=1000000000000000000000 00000001_2	✓ OK
▶	large6 n=1610612737=1100000000000000000000 00000001_2	✓ OK

00010010_2
▶ large5 ✓ OK n=1073741825=1000000000000000000000 00000001_2
▶ large6 ✓ OK n=1610612737=11000000000000000000000 00000001_2

00010010_2
▶ large5 ✓ OK n=1073741825=100000000000000000000000 00000001_2
▶ large6 ✓ OK n=1610612737=110000000000000000000000 00000001_2

	00010010_2	
▶ large5	n=1073741825=100000000000000000000 00000001_2	✓ OK
▶ large6	n=1610612737=1100000000000000000000 00000001_2	✓ OK

00010010_2
▶ large5 ✓ OK n=1073741825=1000000000000000000000 00000001_2
▶ large6 ✓ OK n=1610612737=11000000000000000000000 00000001_2

00010010_2
▶ large5 ✓ OK n=1073741825=1000000000000000000000 00000001_2
▶ large6 ✓ OK n=1610612737=11000000000000000000000 00000001_2

<code>00010010_2</code>	
▶ <code>large5</code> <code>n=1073741825=10000000000000000000000</code> <code>00000001_2</code>	✓ OK
▶ <code>large6</code> <code>n=1610612737=11000000000000000000000</code> <code>00000001_2</code>	✓ OK

Training center