codility

Check out Codility training tasks

# Candidate Report: Anonymous

Test Name:

SUMMARY TIMELINE

## Test Score

100 out of 100 points

100%

## Tasks in Test

| | Time Spent | Task Score |
|---|---|---|
| BinaryGap<br>Submitted in: Java | 50 min | 100% |

## TASKS DETAILS

EASY

1. **BinaryGap**
Find longest sequence of zeros in binary representation of an integer.

| Task Score | Correctness | Performance |
|---|---|---|
| 100% | 100% | Not assessed |

## 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. The number 32 has binary representation `100000` and has no binary gaps.

Write a function:

## Solution

| | |
|---|---|
| Programming language used: | Java |
| Total time used: | 50 minutes |
| Effective time used: | 50 minutes |
| Notes: | *not defined yet* |

## Task timeline

```
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. Given N = 32 the function should return 0, because N has binary representation '100000' and thus no binary gaps.

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

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



09:18:25 10:07:38

Code: 10:07:38 UTC, java, final, score: **100** show code in pop-up

```
1  // you can also use imports, for example:
2  // import java.util.*;
3
4  /**
5   * Class: BinaryGapDecoder
6   * Author: Gautam
7   * Date: 15.11.2018
8   * Purpose: Converts integer into binary form. C
9   **/
10
11 class Solution {
12
13     /**
14      * MethodName: getLongestBinaryGap
15      * Returns the length of longest binary gap
16      **/
17     public int solution(int N) {
18         // Convert integer to binary form
19         String binaryIntegerStr = Integer.toBina
20
21         //System.out.println("Binary Form: " + b
22
23         int longestGap = 0;
24         StringBuilder binaryIntegerBuilder = new
25         int strLength = binaryIntegerBuilder.len
26         int currentIndex = binaryIntegerBuilder.
27
28         // If 1 is not present in binaryformat,
29         if(currentIndex != -1)
30         {
31             // Iterate through the string to ide
32             for(int i = 0; i < strLength; i++)
33             {
34                 int nextIndex = binaryIntegerBui
35                 //System.out.println("currentInd
36                 //System.out.println("nextIndex:
37                 if(nextIndex != -1)
38                 {
39                     int difference  = nextIndex
40
41                     longestGap = (longestGap > d
42                     i = nextIndex;
43                     currentIndex = nextIndex;
44                 }
45                 else
46                 {
47                     break;
48                 }
49             }
50         }
51
52         return longestGap;
```

```
    }
}
```

## Analysis summary

The solution obtained perfect score.

## Analysis

| expand all | Example tests | |
|---|---|---|
| ▶ | example1<br>example test<br>n=1041=10000010001_2 | ✓ OK |
| ▶ | example2<br>example test n=15=1111_2 | ✓ OK |
| ▶ | example3<br>example test n=32=100000_2 | ✓ OK |
| expand all | **Correctness tests** | |
| ▶ | extremes<br>n=1, n=5=101_2 and<br>n=2147483647=2**31-1 | ✓ OK |
| ▶ | trailing_zeroes<br>n=6=110_2 and n=328=101001000_2 | ✓ OK |
| ▶ | power_of_2<br>n=5=101_2, n=16=2**4 and<br>n=1024=2**10 | ✓ OK |
| ▶ | simple1<br>n=9=1001_2 and n=11=1011_2 | ✓ OK |
| ▶ | simple2<br>n=19=10011 and n=42=101010_2 | ✓ OK |
| ▶ | simple3<br>n=1162=10010001010_2 and<br>n=5=101_2 | ✓ OK |
| ▶ | medium1<br>n=51712=110010100000000_2 and<br>n=20=10100_2 | ✓ OK |
| ▶ | medium2<br>n=561892=10001001001011100100<br>_2 and n=9=1001_2 | ✓ OK |
| ▶ | medium3<br>n=66561=10000010000000001_2 | ✓ OK |
| ▶ | large1<br>n=6291457=11000000000000000000<br>0001_2 | ✓ OK |

| | | |
|---|---|---|
| ▶ | large2<br>n=74901729=100011101101110100011100001 | ✓ OK |
| ▶ | large3<br>n=805306373=110000000000000000000000000101_2 | ✓ OK |
| ▶ | large4<br>n=1376796946=1010010000100000100000100010010_2 | ✓ OK |
| ▶ | large5<br>n=1073741825=1000000000000000000000000000001_2 | ✓ OK |
| ▶ | large6<br>n=1610612737=1100000000000000000000000000001_2 | ✓ OK |