

Candidate Report: Anonymous

Test Name:

[Summary](#)[Timeline](#)

Test Score

100 out of 100 points

100%

Tasks in Test

BinaryGap
Submitted in: Java 8

Time Spent ⓘ

1 min

Task Score

100%

TASKS DETAILS

EASY

1. BinaryGap

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

Task Score

Correctness

100%

Performance

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:

```
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].

Copyright 2009–2020 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Solution

Programming language used: Java 8

Total time used: 1 minutes ⓘ

Effective time used: 1 minutes ⓘ

Notes: *not defined yet*

Task timeline

 ⓘ

17:45:34

17:45:51

Code: 17:45:51 UTC, java, final,
score: 100[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 static int max = 30;
9     public static boolean[] mem = new boolean[max + 1]
```

```
10
11     public static int solution(int n) {
12
13         for (int i = max; i >= 0; i--) {
14             int bitmask = (int) Math.pow(2, i)
15
16             // System.out.println(i + " " + ((
17             mem[i] = ((n & bitmask) != 0);
18         }
19
20         // System.out.println("n= " + n + " " + Ar
21
22         int r = 0;
23         int c = 0;
24         boolean count = false;
25         for (int i = 0; i < mem.length; i++) {
26             if (!count && mem[i]) {
27                 count = true;
28                 continue;
29             }
30
31             if (count && mem[i]) {
32                 if (c > r) {
33
34                     r = c;
35                     // System.out.prin
36
37                 }
38                 c = 0;
39
40                 continue;
41             }
42
43             if (count && !mem[i]) {
44                 c++;
45                 // System.out.println("n=
46
47             }
48         }
49
50         return r;
51     }
52 }
```

Analysis summary

The solution obtained perfect score.

Analysis ?

expand all	Example tests
▶ example1	✓ OK
example test n=1041=10000010001_2	
▶ example2	✓ OK
example test n=15=1111_2	
▶ example3	✓ OK
example test n=32=100000_2	
expand all	Correctness tests
▶ extremes	✓ OK
n=1, n=5=101_2 and n=2147483647=2**31-1	
▶ trailing_zeroes	✓ OK
n=6=110_2 and n=328=101001000_2	
▶ power_of_2	✓ OK
n=5=101_2, n=16=2**4 and n=1024=2**10	
▶ 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	✓ OK
n=51712=110010100000000_2 and n=20=10100_2	
▶ medium2	✓ OK
n=561892=10001001001011100100_2 and n=9=1001_2	
▶ medium3	✓ OK
n=66561=10000010000000001_2	
▶ large1	✓ OK
n=6291457=1100000000000000000001_2	
▶ large2	✓ OK
n=74901729=100011101101110100011100 001	
▶ large3	✓ OK
n=805306373=1100000000000000000000 0000101_2	
▶ large4	✓ OK
n=1376796946=1010010000100000100000 100010010_2	
▶ large5	✓ OK
n=1073741825=1000000000000000000000 000000001_2	
▶ large6	✓ OK
n=1610612737=1100000000000000000000 000000001_2	

PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.