Codility_

Candidate Report: trainingN52RZT-GPE

Check out Codility training tasks

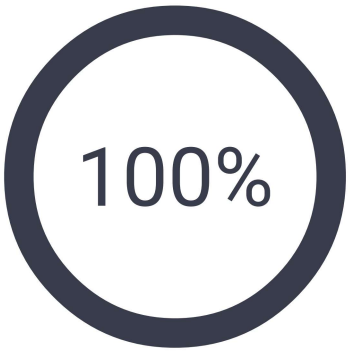
Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
BinaryGap Java 8	47 min	100%

Total score



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:

```
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].

Solution

Programming language used:	Java 8
Total time used:	47 minutes
Effective time used:	47 minutes
Notes:	*not defined yet*

Task timeline

14:17:02 15:03:43

Code: 15:03:42 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 int solution(int N) {
```

```
 9                  // write your code in Java SE 8
10                  if (N > 4 && N < 2147483647) {
11                          int max = 0, oneMetBack = 0, oneMe
12
13                          while (result >= 1) {
14                                  count++;
15                                  if (result % 2 == 1) {
16                                          if (oneMetBack > 0
17                                                  if (oneMet
18                                                          on
19                                                          on
20                                                  } else
21                                                          on
22                                          } else {
23                                                  oneMetBack
24                                          }
25                                          if (oneMetBack > 0
26                                                  max = Math
27                                  }
28
29                                  result = result / 2;
30                          }
31                          return max;
32                  }
33                  return 0;
34          }
35  }
```

Analysis summary

The solution obtained perfect score.

Analysis

expand all	Example tests	
▶	example1 example test n=1041=10000010001_2	✓ OK
▶	example2 example test n=15=1111_2	✓ OK
▶	example3 example test n=32=100000_2	✓ OK
expand all	Correctness tests	
▶	extremes n=1, n=5=101_2 and n=2147483647=2**31-1	✓ OK
▶	trailing_zeroes n=6=110_2 and n=328=101001000_2	✓ OK
▶	power_of_2 n=5=101_2, n=16=2**4 and n=1024=2**10	✓ OK
▶	simple1 n=9=1001_2 and n=11=1011_2	✓ OK
▶	simple2 n=19=10011 and n=42=101010_2	✓ OK
▶	simple3 n=1162=10010001010_2 and n=5=101_2	✓ OK
▶	medium1 n=51712=110010100000000_2 and n=20=10100_2	✓ OK
▶	medium2 n=561892=10001001001011100100_2 and n=9=1001_2	✓ OK
▶	medium3 n=66561=10000010000000001_2	✓ OK
▶	large1 n=6291457=110000000000000000000001_2	✓ OK

▶ large2 n=74901729=1000111011011101000111000001	✓ OK
▶ large3 n=805306373=11000000000000000000000000101_2	✓ OK
▶ large4 n=1376796946=10100100001000001000001000010010_2	✓ OK
▶ large5 n=1073741825=10000000000000000000000000001_2	✓ OK
▶ large6 n=1610612737=11000000000000000000000000001_2	✓ OK