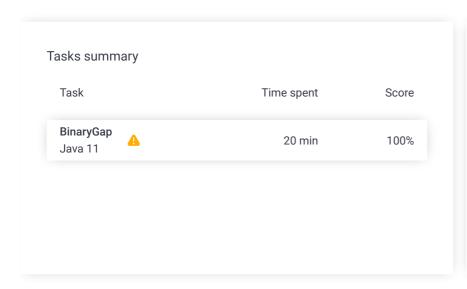
Codility_

CodeCheck Report: training2YYVZW-632

Test Name:

Summary Timeline

Check out Codility training tasks





Tasks Details

1. BinaryGap

Edsy

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

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.

Solution

Programming language used: Java 11

Total time used: 20 minutes

Effective time used: 20 minutes 2

Notes: not defined yet

Task timeline



17:00:32 17:19:59

Code: 17:19:59 UTC, java11, show code in pop-up final, score: 100

- 1 // you can also use imports, for example:
- 2 // import java.util.*;
 3

Write an efficient algorithm for the following assumptions:

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

Copyright 2009–2021 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
// you can write to stdout for debugging purposes, \epsilon
     // System.out.println("this is a debug message");
     // you can also use imports, for example:
 6
7
     import java.util.*;
 8
9
     // you can write to stdout for debugging purposes,
10
     // System.out.println("this is a debug message");
11
     import java.util.ArrayList;
12
     import java.util.List;
     import java.lang.*;
13
14
     class Solution {
15
         public int solution(int N) {
16
         int finalGap = 0;
         int n=N;
17
18
             //1 is 0
19
             if (n == 1) {
20
                 return 0;
21
             char binaryRep[] = Integer.toBinaryString(n)
22
23
             int tempGap=0;
             for (int x = 0; x < binaryRep.length; x++) {
24
25
                 if(binaryRep[x]=='0'){
                     tempGap++;
26
27
                     continue;
                 }else if(binaryRep[x]=='1'){
28
29
                     if(tempGap>finalGap)
30
                     finalGap=tempGap;
31
                     tempGap=0;
32
33
34
35
             return finalGap;
36
37
     }
```

Analysis summary

The solution obtained perfect score.

Analysis

ехра	nd all	Example tests	S	
>	example1 example test n=10	041=10000010001_2	✓ OK	
•	example2 example test n=15		✓ OK	
•	example3 example test n=32		√ OK	
ехра	nd all	Correctness tes	sts	
•	extremes n=1, n=5=101_2 at n=2147483647=2	nd	√ OK	
•	trailing_zeroes	=328=101001000_2	√ OK	
•	power_of_2 n=5=101_2, n=16= n=1024=2**10		√ OK	
•	simple1 n=9=1001_2 and r		√ OK	
•	simple2 n=19=10011 and r		√ OK	
>	simple3 n=1162=1001000 n=5=101_2		√ OK	

•	medium1 n=51712=110010100000000_2 and n=20=10100_2	✓	ОК
•	medium2 n=561892=10001001001011100100_2 and n=9=1001_2	✓	OK
•	medium3 n=66561=1000001000000001_2	✓	OK
•	large1 n=6291457=1100000000000000000000000000000000000	✓	OK
•	large2 n=74901729=1000111011011101000 11100001	✓	OK
•	large3 n=805306373=110000000000000000000000000000000000	✓	OK
•	large4 n=1376796946=10100100001000001 00000100010010_2	√	OK
•	large5 n=1073741825=1000000000000000000000000000000000000	√	OK
•	large6 n=1610612737=110000000000000000 00000000000001_2	✓	OK