codility

aining center
k out Codility training tasks

Congratulations

You have completed a Codility training test.

Tweet this!
I scored 100% in #java on @Codility!
https://codility.com/demo/take-sample-test/binary_gap/

Sign up for our newsletter!

Like us on Facebook!

Training ticket

Session

ID: trainingFYCPWZ-NP5
Time limit: 120 min.

Status: closed

Created on: 2016-06-17 21:39 UTC
Started on: 2016-06-17 21:39 UTC
Finished on: 2016-06-17 21:40 UTC

Tasks in test	Correctness	Performance	Task score
1 BinaryGap Submitted in: Java	100%	not assessed	100%

Test score

100%

100 out of 100 points

EASY

1. BinaryGap

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

score: 100 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:

Solution

Programming language used: Java

Total time used: 2 minutes

Effective time used: 2 minutes

Notes: *not defined yet*

Task timeline

21:39:13 21:40:34

Code: 21:40:34 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
```

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


```
7  class Solution {
8      private static final String STOP_SIGN = "1";
9
10     public int solution(int N) {
11         String binaryRepresentation = Integer.toBinaryS
12
13         int maxCounter = 0;
14         int prevOneOccurence = 0;
15         int nextOneOccurence = binaryRepresentation.inc
16
17         while (nextOneOccurence != -1) {
18             maxCounter = Math.max(maxCounter, nextOneOc
19             prevOneOccurence = nextOneOccurence;
20             nextOneOccurence = binaryRepresentation.inc
21         }
22
23         return maxCounter;
24     }
25 }
```

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

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=11000000000000000000001_2	✔ OK
▶	large2 n=74901729=10001110110111010001110001	✔ OK
▶	large3 n=805306373=11000000000000000000000000101_2	✔ OK
▶	large4 n=1376796946=10100100001000001000001 00010010_2	✔ OK
▶	large5	✔ OK

	n=1073741825=10000000000000000000000000000001_2	
▶	large6 n=1610612737=11000000000000000000000000000001_2	✔ OK

Training center