

# Codility

## CodeCheck Report: training29VSY5-DU9

Test Name:

[Check out Codility training tasks](#)

Summary

Timeline

### Tasks summary

Task	Time spent	Score
BinaryGap Python	8 min	100%

### Total score


 100%

### Tasks Details

Easy	1. <b>BinaryGap</b>	<b>Task Score</b>	<b>Correctness</b>	<b>Performance</b>
	Find longest sequence of zeros in binary representation of an integer.			
		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.

### Solution

Programming language used: Python

Total time used: 8 minutes ?

Effective time used: 8 minutes ?

Notes: not defined yet

### Task timeline

Write a function:

```
def solution(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 2009–2023 by Codility Limited. All Rights Reserved.  
Unauthorized copying, publication or disclosure prohibited.



07:31:27

07:38:58

Code: 07:38:57 UTC, py, [show code in pop-up](#)  
final, score: **100**

```
1 # you can write to stdout for debugging
2 # print("this is a debug message")
3
4 def solution(N):
5     # Implement your solution here
6     N = bin(N).strip("0b")
7     zero_sequences = N.split('1')
8
9     max_zeros = max(len(seq) for seq in
10
11     return max_zeros
```

## Analysis summary

The solution obtained perfect score.

## Analysis

expand all	Example tests
▶	example1 <span>✓ OK</span> example test n=1041=10000010001_2
▶	example2 <span>✓ OK</span> example test n=15=1111_2
▶	example3 <span>✓ OK</span> example test n=32=100000_2
expand all	Correctness tests
▶	extremes <span>✓ OK</span> n=1, n=5=101_2 and n=2147483647=2**31-1
▶	trailing_zeroes <span>✓ OK</span> n=6=110_2 and n=328=101001000_2
▶	power_of_2 <span>✓ OK</span> n=5=101_2, n=16=2**4 and n=1024=2**10
▶	simple1 <span>✓ OK</span> n=9=1001_2 and n=11=1011_2
▶	simple2 <span>✓ OK</span> n=19=10011 and n=42=101010_2
▶	simple3 <span>✓ OK</span> n=1162=10010001010_2 and n=5=101_2
▶	medium1 <span>✓ OK</span>

n=51712=110010100000000\_2  
and n=20=10100\_2

▶ medium2 ✓ OK  
n=561892=10001001001011100  
100\_2 and n=9=1001\_2

▶ medium3 ✓ OK  
n=66561=10000010000000001\_2

▶ large1 ✓ OK  
n=6291457=1100000000000000  
0000001\_2

▶ large2 ✓ OK  
n=74901729=100011101101110  
100011100001

▶ large3 ✓ OK  
n=805306373=110000000000000  
000000000000101\_2

▶ large4 ✓ OK  
n=1376796946=1010010000100  
000100000100010010\_2

▶ large5 ✓ OK  
n=1073741825=10000000000000  
00000000000000001\_2

▶ large6 ✓ OK  
n=1610612737=11000000000000  
00000000000000001\_2