codility

Check out Codility training tasks

Candidate Report: Anonymous

Test Name:

Summary Timeline

Test Score

100 out of 100 points

100%

Tasks in Test

	Time Spent	Task Score
BinaryGap Submitted in: C++	2 min	100%

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:

Solution

Programming language used:	C++
Total time used:	2 minutes
Effective time used:	2 minutes
Notes:	*not defined yet*

Task timeline

```
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].

11:06:27 11:07:59

Code: 11:07:59 UTC, cpp, final, score: **100**

show code in pop-up

```cpp
#include <stack>
#include <algorithm>
#include <iostream>
#include <array>

constexpr static int MAX_DIGITS = 32;
int solution(int number) {

   // Get all the binary digits
   std::array<int, MAX_DIGITS> digits;

   std::uint32_t to_find_digit = (std::uint32_t
   //for (auto i = MAX_DIGITS-1; i >=0 ; --i) {
   for (auto i = 0; i < MAX_DIGITS; ++i) {
      digits[i] = ((number & to_find_digit) ? 1
      to_find_digit >>= 1;
   }

   // For each ones do
   // Find the next one and count zeros on the
   // Repeat for all the elements
   int length = 0;
   for (auto i = 0; i < MAX_DIGITS; ++i) {
      auto curlen = 0;
      if (1 == digits[i]) {
         bool found = false;
         for (auto j = i + 1; j < MAX_DIGITS; +
            if (1 == digits[j]) {
               // Found next 1
               found = true;
               break;
            }
            ++curlen;
         }
         if (found && (curlen > length)) {
            length = curlen;
         }
      }
   }
   return length;
}
```

Analysis summary

The solution obtained perfect score.

Analysis

collapse all **Example tests**

▼ example1 ✓ OK
example test n=1041=10000010001_2

1. 0.001 s **OK**

▼ example2 ✓ OK
example test n=15=1111_2

1. 0.001 s **OK**

▼ example3 ✓ OK
example test n=32=100000_2

1. 0.001 s **OK**

collapse all **Correctness tests**

▼ extremes ✓ OK
n=1, n=5=101_2 and n=2147483647=2**31-1

1. 0.001 s **OK**
2. 0.001 s **OK**
3. 0.001 s **OK**

▼ trailing_zeroes ✓ OK
n=6=110_2 and n=328=101001000_2

1. 0.001 s **OK**
2. 0.001 s **OK**

▼ power_of_2 ✓ OK
n=5=101_2, n=16=2**4 and n=1024=2**10

1. 0.001 s **OK**
2. 0.001 s **OK**
3.

0.001 s OK

▼ simple1 ✓ OK
n=9=1001_2 and n=11=1011_2

1. 0.001 s OK
2. 0.001 s OK

▼ simple2 ✓ OK
n=19=10011 and n=42=101010_2

1. 0.001 s OK
2. 0.001 s OK

▼ simple3 ✓ OK
n=1162=10010001010_2 and n=5=101_2

1. 0.001 s OK
2. 0.001 s OK

▼ medium1 ✓ OK
n=51712=110010100000000_2 and n=20=10100_2

1. 0.001 s OK
2. 0.001 s OK

▼ medium2 ✓ OK
n=561892=10001001001011100100_2 and n=9=1001_2

1. 0.001 s OK
2. 0.001 s OK

▼ medium3 ✓ OK
n=66561=10000010000000001_2

1. 0.001 s OK

▼

large1 ✓ OK
n=6291457=11000000000000000000001_2

1. 0.001 s OK

▼ large2 ✓ OK
n=74901729=100011101101110100011100001

1. 0.001 s OK

▼ large3 ✓ OK
n=805306373=110000000000000000000000000101_2

1. 0.001 s OK

▼ large4 ✓ OK
n=1376796946=1010010000100000100000100010010_2

1. 0.001 s OK

▼ large5 ✓ OK
n=1073741825=1000000000000000000000000000001_2

1. 0.001 s OK

▼ large6 ✓ OK
n=1610612737=1100000000000000000000000000001_2

1. 0.001 s OK