Codility_

CodeCheck Report: trainingTKUK4Q-VB5

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
SparseBinaryDecomposition Python	28 min	100%

Total score

Tasks Details

1. SparseBinaryDecomposition

Hard

Decompose int into sum of ints having no consecutive 1s in binary form.

Task Score	Correctness	Performance
100%	100%	100%

Task description

A non-negative integer N is called *sparse* if its binary representation does not contain two consecutive bits set to 1. For example, 41 is sparse, because its binary representation is "`101001`" and it does not contain two consecutive 1s. On the other hand, 26 is not sparse, because its binary representation is "`11010`" and it contains two consecutive 1s.

Two non-negative integers P and Q are called a *sparse decomposition* of integer N if P and Q are sparse and N = P + Q.

For example:

- 8 and 18 are a sparse decomposition of 26 (binary representation of 8 is "`1000`", binary representation of 18 is "`10010`");
- 9 and 17 are a sparse decomposition of 26 (binary representation of 9 is "`1001`", binary representation of 17 is "`10001`");
- 2 and 24 are not a sparse decomposition of 26; though 2 + 24 = 26, the binary representation of 24 is "`11000`", which is not sparse.

Write a function:

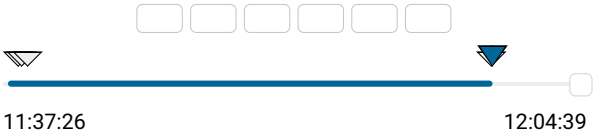
```
def solution(N)
```

that, given a non-negative integer N, returns any integer that is one

Solution

Programming language used:	Python
Total time used:	28 minutes
Effective time used:	28 minutes
Notes:	*not defined yet*

Task timeline



11:37:26 12:04:39

Code: 12:04:39 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debugging purposes
2  # print("this is a debug message")
3
4  def solution(N):
5      # Implement your solution here
```

part of a sparse decomposition of N. The function should return −1 if there is no sparse decomposition of N.

For example, given N = 26 the function may return 8, 9, 17 or 18, as explained in the example above. All other possible results for N = 26 are 5, 10, 16 and 21.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [0..1,000,000,000].


```
6      pass
7      a = N & 0x55555555
8      b = N & 0xAAAAAAAA
9
10     if (a + b == N):
11         return a
12
13     return -1
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(log(N)) or O(1)**

expand all	Example tests	
▶	example1 example test n=26	✔ OK
expand all	**Correctness tests**	
▶	simple1 n=1166	✔ OK
▶	simple2 n=561892	✔ OK
▶	simple3 n=1031	✔ OK
▶	small_power_of_two_minus_one n=1023	✔ OK
▶	extreme n <= 5	✔ OK
expand all	**Performance tests**	
▶	medium1 n=74901729	✔ OK
▶	medium2 n=216188401	✔ OK
▶	power_of_two_minus_one n=536870911	✔ OK
▶	big_random n=~1000000000	✔ OK
▶	maximal n=1000000000	✔ OK