Daily Coding Problem #214

Problem

This problem was asked by Stripe.

Given an integer n, return the length of the longest consecutive run of 1s in its binary representation.

For example, given 156, you should return 3.

Solution

The most straightforward way to solve this would be to loop over the bits of the number, keeping track of a counter of the maximum number of consecutive 1s seen. Whenever we see a longer run of set bits, we update our counter.

```
def find_length(n):
    n = bin(n)[2:]
    max_length = current_length = 0

    for digit in n:
        if digit == '1':
            current_length += 1
            max_length = max(max_length, current_length)
        else:
            current_length = 0

    return max_length
```

This is O(N), where N is the number of digits in our input. Can we do better?

Let's try using using bit manipulation. In particular, note that if we perform the operation `x & x << 1`, the longest consecutive run of 1s must decrease by one. This is because all but one of the set bits in the original number will correspond to set bits in the shifted number. Using the example in the problem, we can see that the maximum length changes from 3 to 2:

```
  10011100
& 00111000
-----------
  00011000
```

With this in mind, we can continue to AND our input with a shifted version of itself until we reach 0. The number of times we perform this operation will be our answer.

```
def find_length(n):
    max_length = 0

    while n:
        max_length += 1
        n = n & (n << 1)

    return max_length
```

While the worst case here is the same as above, the number of operations we must perform is just the length of the longest consecutive run.