Daily Coding Problem #317

Problem

This problem was asked by Yahoo.

Write a function that returns the bitwise AND of all integers between `M` and `N`, inclusive.

Solution

The simplest solution would be to loop through each number from `M` to `N`, applying a bitwise AND between the accumulated result and each new integer.

```
def and_product(m, n):
    res = m

    for i in range(m + 1, n + 1):
        res &= i

    return res
```

Since we must carry out up to `N` operations, this solution will run in `O(N)` time and `O(1)` space.

As is often the case with bitwise problems, there is a faster solution. Let us consider two

As is often the case with bitwise problems, there is a faster solution. Let us consider two arbitrary bits, say `M = 1001001` and `N = 1110001`.

Note that the most significant bit of both digits, 2^6, is the same. As a result, it is guaranteed that each number between these two will also have a `1` in that place.

Next, we can see that second-most significant bit of `M` is 2^3, while that of `N` is 2^5. This tells us that no other bits can occur in common in every number between these two. Why? Consider incrementing a count one by one to get from `M` to `N`. At some point, we would have to flip the second-highest order bit of `N`, going from `1011111` to `1100000`. Clearly, if we perform a bitwise AND between these two numbers, every bit except the first will get set to zero.

After the highest differing bit, then, a cumulative AND will set all bits to zero. Therefore, we can solve this by bit-shifting each integer down until the two numbers are equal, and then bit-shifting back up an equivalent number of places.

```
def and_product(m, n):
    i = 0

    while m != n:
        m >>= 1
        n >>= 1
        i += 1

    return n << i
```

There will be roughly `log N` bits in the binary representation of `N`, so this solution will run in `O(log N)` time and `O(1)` space.