Bitwise AND of Numbers Range

Given a range [m, n] where 0

For example, given the range [5, 7], you should return 4.

Credits:
Special thanks to @amrsaqr for adding this problem and creating all test cases.

Solution 1

The idea is very simple:

1. last bit of (odd number & even number) is 0.
2. when m != n, There is at least an odd number and an even number, so the last bit position result is 0.
3. Move m and n rigth a position.

Keep doing step 1,2,3 until m equal to n, use a factor to record the iteration time.

```
public class Solution {
    public int rangeBitwiseAnd(int m, int n) {
        if(m == 0){
            return 0;
        }
        int moveFactor = 1;
        while(m != n){
            m >>= 1;
            n >>= 1;
            moveFactor <<= 1;
        }
        return m * moveFactor;
    }
}
```

written by zwangbo original link here

Solution 2

Consider the bits from low to high. if n > m, the lowest bit will be 0, and then we could transfer the problem to sub-problem: rangeBitwiseAnd(m>>1, n>>1).

```
int rangeBitwiseAnd(int m, int n) {
    return (n > m) ? (rangeBitwiseAnd(m/2, n/2) << 1) : m;
}
```

written by applewolf original link here

Solution 3

The idea is to use a mask to find the leftmost common digits of m and n. Example: m=1110001, n=1110111, and you just need to find 1110000 and it will be the answer.

```
public class Solution {
public int rangeBitwiseAnd(int m, int n) {
    int r=Integer.MAX_VALUE;
    while((m&r)!=(n&r))  r=r<<1;
    return n&r;
}
```

}

written by yu14 original link here