- Given an integer n, transform to zero by choosing repetetively:

  1.) Change the 0^{th} bit in the binary version of n

  2.) Change the i^{th} bit if

  a.) $i-1$ is 1 AND

  b.) $i-2 \to 0$ are 0

Return the minimum number of operations to make $n = 0$

Given:	0 1 1	0 1 1
1.) Choose a.	0 1 0	0 0 1
2.) Blocked!		0 0 0

My first instinct was do #1 whenever possible. Does not work if $i + 1 = 1$ for $i = 0$

Given:	1 1 0	
1.) 3rd bit to 0 via #		0 1 0
2.) Step #1	0 1 1	
3.) Step #2	0 0 1	
4.) Step #3	0 0 0	

Total steps: 4

Algorithm:

- Find highest set bit
- Find lowest set bit

	1 1 0	1 0 0 0		
1→2	1 1 0	1 0 0 1	1	
	0 1 0			
1→2	0 1 1	1 0 1 1	3	1
	0 0 1			
3→0	0 0 0	1 0 1 0	2	2
2→4		1 1 1 0	6	3
1→1		1 1 1 1	7	4

7 + 3 + 1 + highest bit → 0

1 → 1 → 1

10 → 11 → 01 → 00 3

100 → 101 → 111 → 110 → 010 → 011 → 001 → 000 7

11 → 01 → 00 2

00100

100 - 4

111 - 7

```
should_add = true
mask =    0xEFFF
while mask != 0
    bit = input & mask
    add = 0
    while add < bit
        add = (add << 1) + 1
    if bit and should_add
        ans += add
    else if bit and !should_add
        ans -= add
return ans
```

Ex.

110

From left to right

Bit masking!

100 → 7 ans += 7 : 7

010 → 3 ans -= 3

ans = 4 !

1101	5	5
1100	4	6
0100	0	7
0101	1	1
0111	1	2
0110	3	3
0010	0	1
0011	1	
0001	0	
0000	0	

1.) Start with bitmask for largest value w/ only 1 bit set
If unsigned: 0xE0000000 (For 32bit only)
If signed: 0x40000000

2.) Capture n masked by 0x40000000
If capture is not 0, and the number
of operations performed is even or zero,
augment assign captured value w/ all right bits
filled.
If # operations is odd, subtract that value from
result.

3.) Right shift mask by 1

4.) When mask = 0, you have answer!