Number Complement

Given a positive integer, output its complement number. The complement strategy is to flip the bits of its binary representation.

Note:

1. The given integer is guaranteed to fit within the range of a 32-bit signed integer.
2. You could assume no leading zero bit in the integer's binary representation.

Example 1:

```
Input: 5
Output: 2
Explanation: The binary representation of 5 is 101 (no leading zero bits), and its complement is 010. So you need to output 2.
```

Example 2:

```
Input: 1
Output: 0
Explanation: The binary representation of 1 is 1 (no leading zero bits), and its co
mplement is 0. So you need to output 0.
```

Solution 1

```
class Solution {
public:
    int findComplement(int num) {
        unsigned mask = ~0;
        while (num & mask) mask <<= 1;
        return ~mask & ~num;
    }
};
```

For example,

```
num          = 00000101
mask         = 11111000
~mask & ~num = 00000010
```

written by lzl124631x137 original link here

Solution 2

I post solution first and then give out explanation. Please think why does it work before read my explanation.

```
public class Solution {
    public int findComplement(int num) {
        return ~num & ((Integer.highestOneBit(num) << 1) - 1);
    }
}
```

According to the problem, the result is

1. The `flipped` version of the original `input` but
2. Only flip `N` bits within the range from `LEFTMOST` bit of `1` to `RIGHTMOST`. For example input = `5` (the binary representation is `101`), the `LEFTMOST` bit of `1` is the third one from `RIGHTMOST` (`100`, `N` = 3). Then we need to flip 3 bits from `RIGHTMOST` and the answer is `010`

To achieve above algorithm, we need to do 3 steps:

1. Create a bit mask which has `N` bits of `1` from `RIGHTMOST`. In above example, the mask is `111`. And we can use the decent Java built-in function `Integer.highestOneBit` to get the `LEFTMOST` bit of `1`, left shift one, and then minus one. Please remember this wonderful trick to create bit masks with `N` ones at `RIGHTMOST`, you will be able to use it later.
2. Negate the whole input number.
3. `Bit AND` numbers in step `1` and `2`.

Three line solution if you think one line solution is too confusing:

```
public class Solution {
    public int findComplement(int num) {
        int mask = (Integer.highestOneBit(num) << 1) - 1;
        num = ~num;
        return num & mask;
    }
}
```

written by shawngao original link here

Solution 3

```
class Solution(object):
    def findComplement(self, num):
        i = 1
        while i <= num:
            i = i << 1
        return (i - 1) ^ num
```

written by Ipeq1 original link here

From Leetcoder.