476. Number Complement

The **complement** of an integer is the integer you get when you flip all the `0`'s to `1`'s and all the `1`'s to `0`'s in its binary representation.

- For example, The integer `5` is `"101"` in binary and its **complement** is `"010"` which is the integer `2`.

Given an integer `num`, return *its complement*.

Example 1:

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

Constraints:

- `1 <= num < 2`31

476. Number Complement

/*

C++ built-in functions

Time complexity: O(log_2n)

Space complexity: O(1)

*/

```
class Solution {
public:
    int mylog(int x) {
        return 32-__builtin_clz(x)-1;
    }

    int findComplement(int n) {
        // Find number of bits of n
        int m=floor(mylog(n))+1;

        // One's complement=((2^m)-1) xor n
        return (int)(((1*1ll)<<m)-1)^n;
    }
};
```

```
class Solution {
public:
    int findComplement(int n) {
        // Find number of bits of n
        int m=floor(log2(n))+1;

        // One's complement=((2^m)-1) xor n
        return (int)(((1*1ll)<<m)-1)^n;
    }
};
```

```
class Solution {
public:
    int findComplement(int n) {
        return ((~0) << (32-__builtin_clz(n))) ^ (~n);
    }
};
```

476. Number Complement

```
/*
    Bit manipulation
    Time complexity: O(log2n)
    Space complexity: O(1)
*/
class Solution {
public:
    int findComplement(int num) {
        if (num == 0) return 1;

        // Create a 32 set bits
        unsigned int mask = ~0;

        // ~mask is equal to(2^m)-1
        while (num & mask) {
            mask <<= 1;
        }

        // ((2^m)-1) ^ num
        return ~mask & ~num;
    }
};
```