Number Complement (LeetCode476 - Easy)

Problem Description

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: **input:** x = 5 (101) => **output:** 2 (010)

$0 \le x, y < 2^{31}$

Problem Analysis

The key point of this problem is that we only flip the highest bit and all the bits on right of it. We do not flip leading zeros.

Example: input = 9 (0000 1001)

- Get the highest bit and only flip the right part. (Including highest bit)
  - Java build-in method **Integer.highestOneBit(input)** -> 0000 1000
- Construct a mask that has 0s on left part and 1s on right part.
  - The mask should be 0000 1111
  - How to get the mask? **(highestOneBit << 1) - 1**
- Return **mask & ~input**. (0000 1111 & 1111 0110 = 0000 0110)

Solution

```
/**
 * @param num
 * @return complement of num.
 */
public static int findComplement(int num) {
      int highestBit = Integer.highestOneBit(num);
      int mask = (highestBit << 1) - 1;
      return mask & (~num);
}
```