# 题目

给定一个正整数，输出它的补数。补数是对该数的二进制表示取反。

**示例 1:**

输入: 5

输出: 2

解释: 5 的二进制表示为 101（没有前导零位），其补数为 010。所以你需要输出 2 。

**示例 2:**

输入: 1

输出: 0

解释: 1 的二进制表示为 1（没有前导零位），其补数为 0。所以你需要输出 0 。

**注意:**

给定的整数保证在32位带符号整数的范围内。

你可以假定二进制数不包含前导零位。

本题与1009相同：

https://leetcode-cn.com/problems/complement-of-base-10-integer/

# 分析

## 方法一：位运算

**思路：**

1、对输入num去反，取反后二进制数字第一个1前面的零均变为1，这bit的1变为0,所有要将去反后第一个0前面的1删除

例如：0x05=0000000000000101 取反：1111111111111010 那么需要将第一个零1111111111111010 前面的1删掉

2、删除1前面的0:使用&=~清零

**代码：**

class Solution {

public:

int findComplement(int num) {

int oneNum = ~num;

int oneBitFlag = 0x80000000;

while((oneNum & oneBitFlag) != 0) {

oneNum &= ~oneBitFlag;

oneBitFlag >>= 1;

}

return oneNum;

}

};

或：

5的二进制是：0101，7的二进制是：0111，它们的异或为：0010，去掉前导零位即为取反。

再来一个例子，假设a为1110 0101，b为1111 1111，a^b = 0001 1010是a的取反。也就是说二进制位数与num相同，且全为1的数tmp与num的异或即为所求。

class Solution {

public:

int findComplement(int num) {

int tmp = 1;

while (tmp < num)//当tmp>num时,此时正好右侧多1位

{

tmp <<= 1;

tmp += 1;

}

return (tmp^num);

//此时^操作必然最后多的1位为1,其余为0,即取反

}

};

## 方法二：数学运算

**思路：**

正整数num+num的补数=2^k-1，2^k>num且2^(k-1)<num。

**代码：**

class Solution {

public:

int findComplement(int num) {

int x=log2(num)+1;

return (unsigned)(1<<x)-num-1;//防止越界

}

};