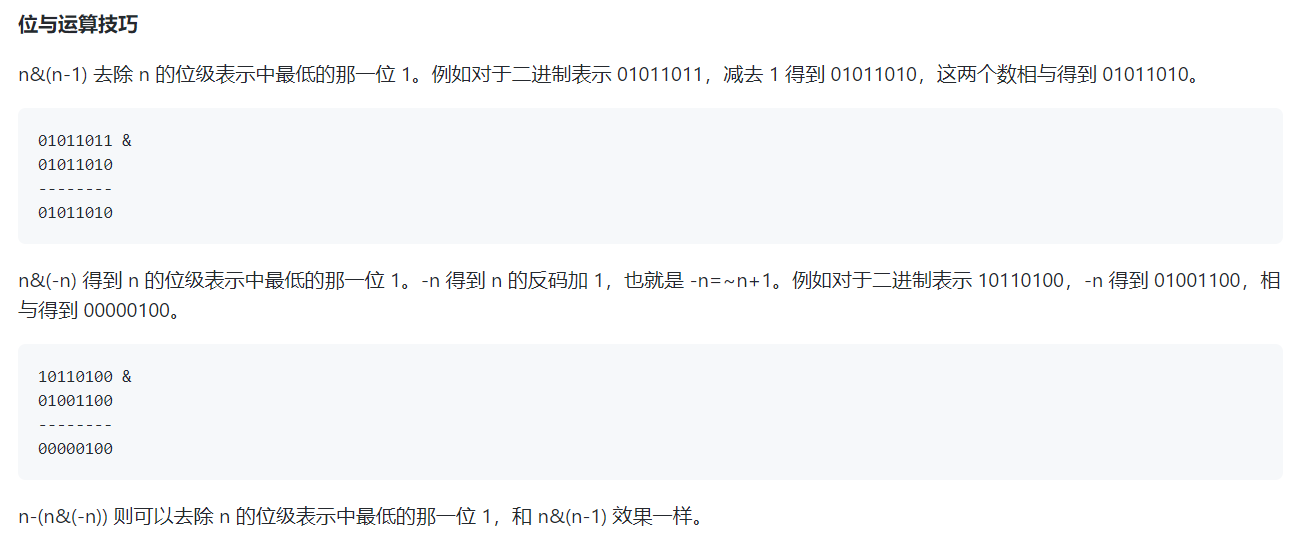
^ xor 一样出0 不一样出1

& 有0出0

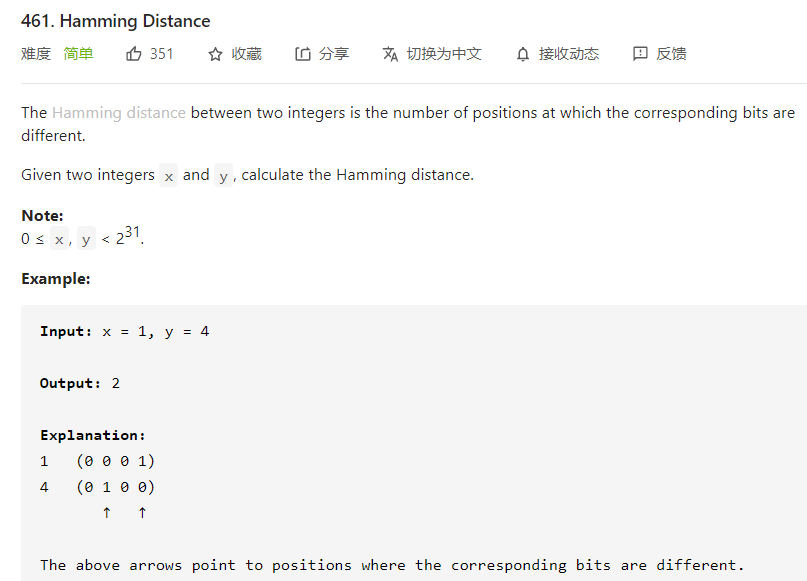
| or 有1出1

>>>无符号右移 左边直接补0

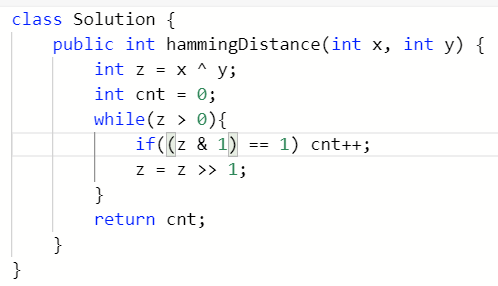
>>算术右移，左边补1或0



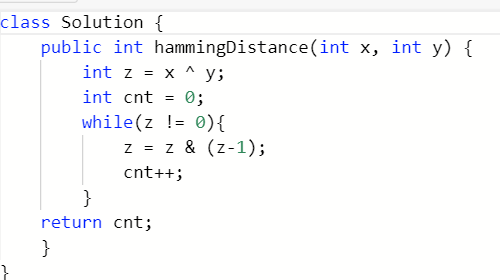
1.



解法1：先把两个相互xor 得到不同的位数为1，再把这个xor完的数字与1作&，== 1则说明这位是1，之后把xor完的数字向右移动一位 也就是减去最右边的那位，再在最左边填0

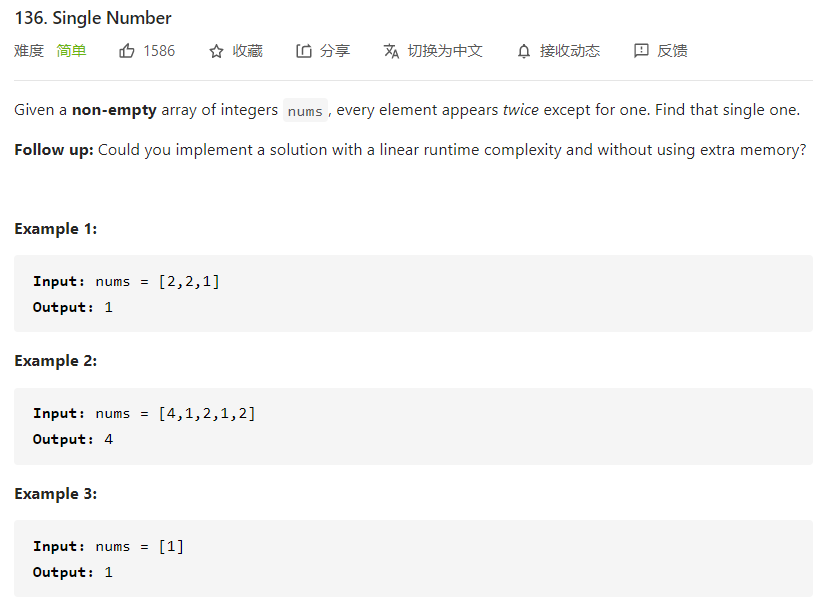


解法2：先xor得到不同的位数上是1，用n&(n-1)去去掉最右边的1，知道xor之后的数=0

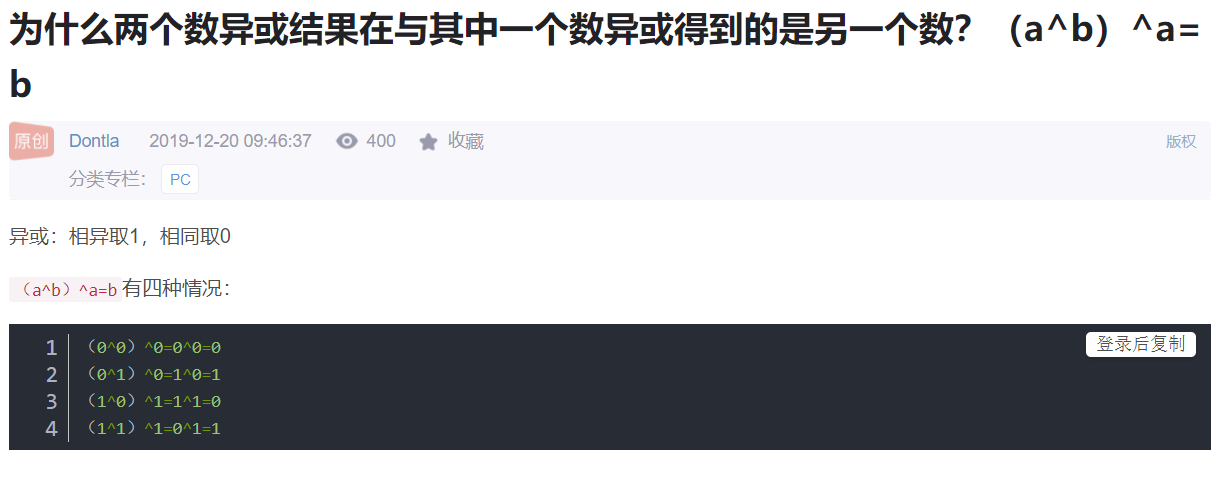


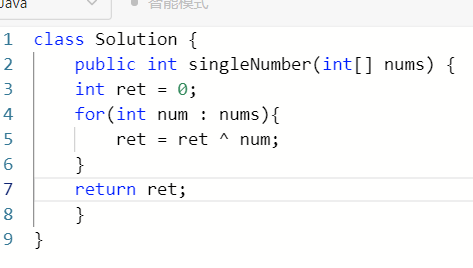
解法3：与之前一样用xor，再用Integer.bitCount()方法来统计1的数量

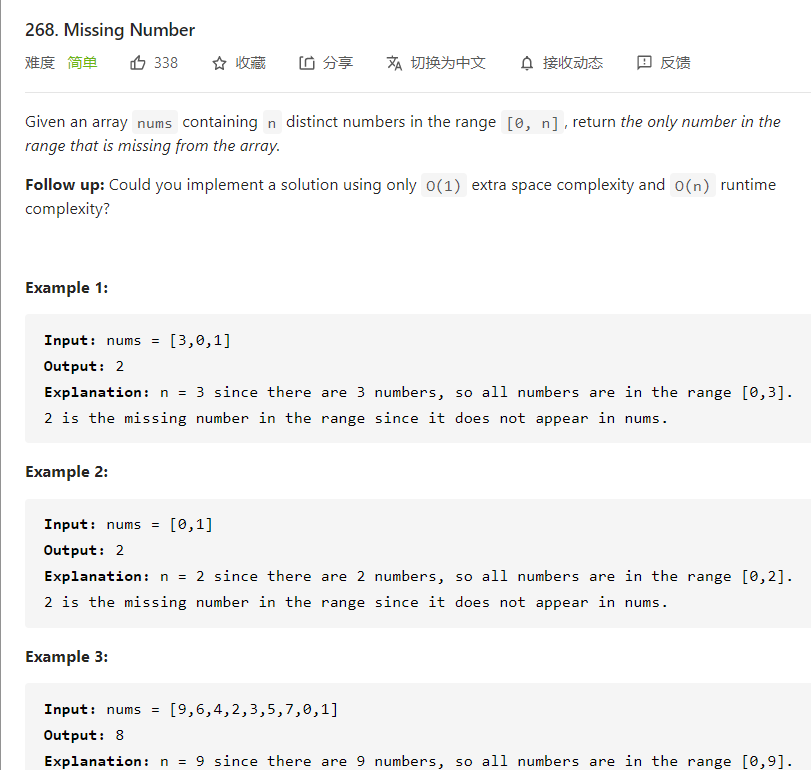
2.



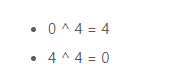
解法：对所有数进行异或，相同的数异或结果为0 剩下的就是那个没有相同数的数



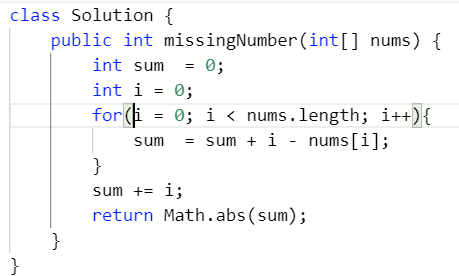




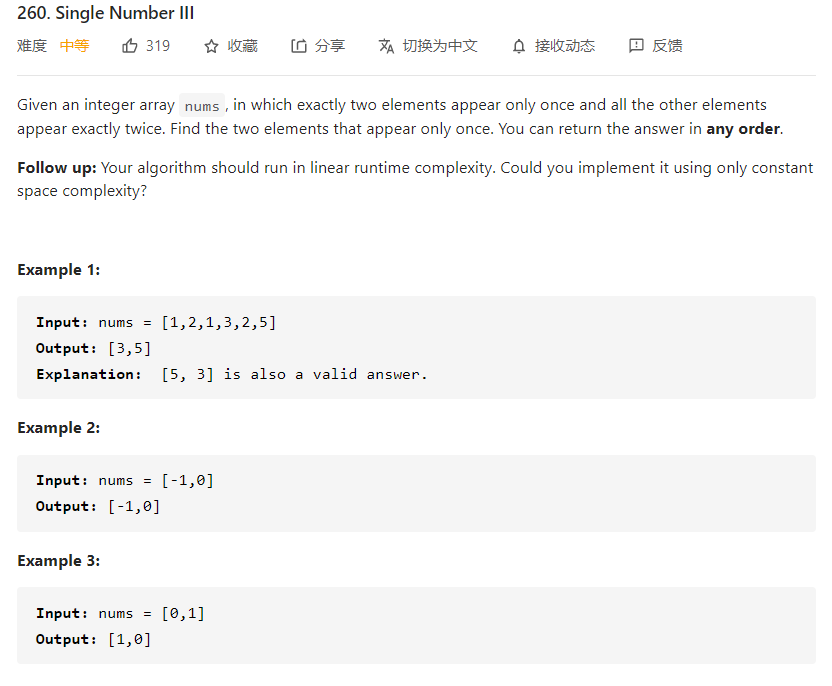
解法1：因为是0到n，所以每次用一个for loop去xor一个i就可以抵消掉array里的值，剩下没被抵消的就是缺失的



解法2：一个for loop每次都加i并且减去array里的值，剩下的数就是因为缺失没减的数



4.

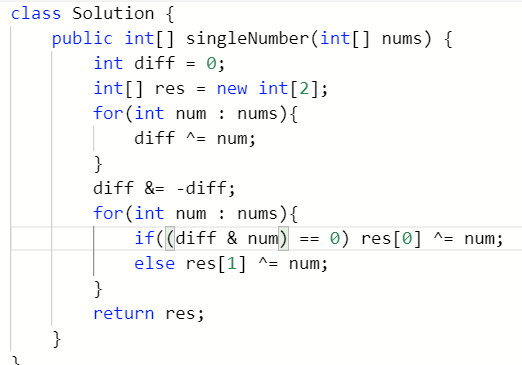


解法：两个不相等的元素在位级表示上必定会有一位存在不同。

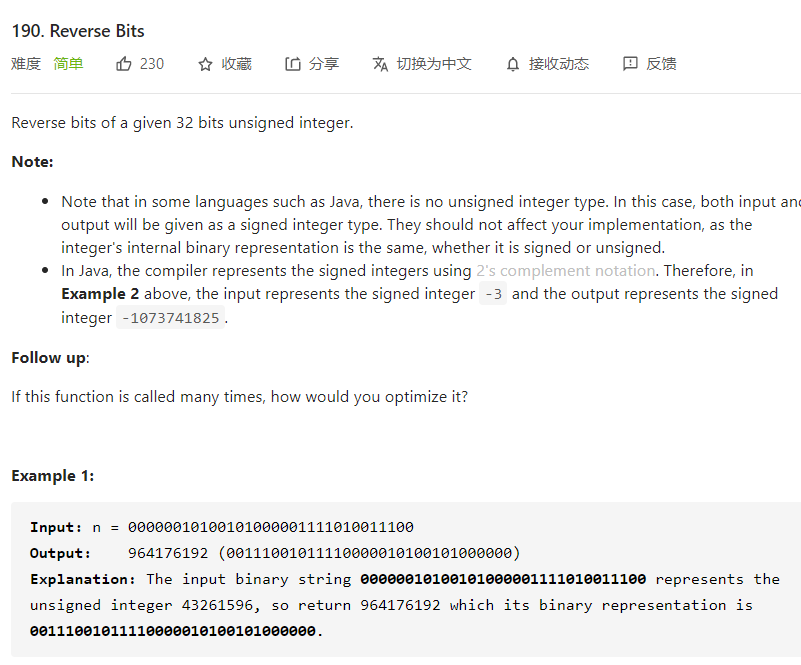
将数组的所有元素异或得到的结果为不存在重复的两个元素异或的结果。

diff &= -diff 得到出 diff 最右侧不为 0 的位，也就是不存在重复的两个元素在位级表示上最右侧不同的那一位，利用这一位就可以将两个元素区分开来。

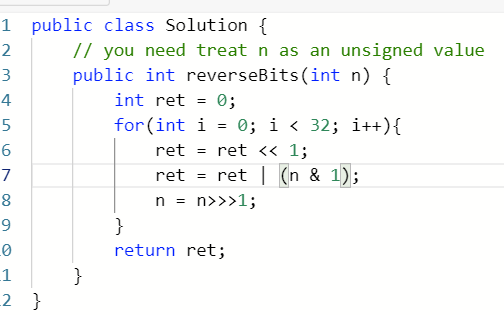
diff与不重复的两个数分别作&，一个会等于0 一个会等于他自己，所以通过不停xor找出这两个特征就行



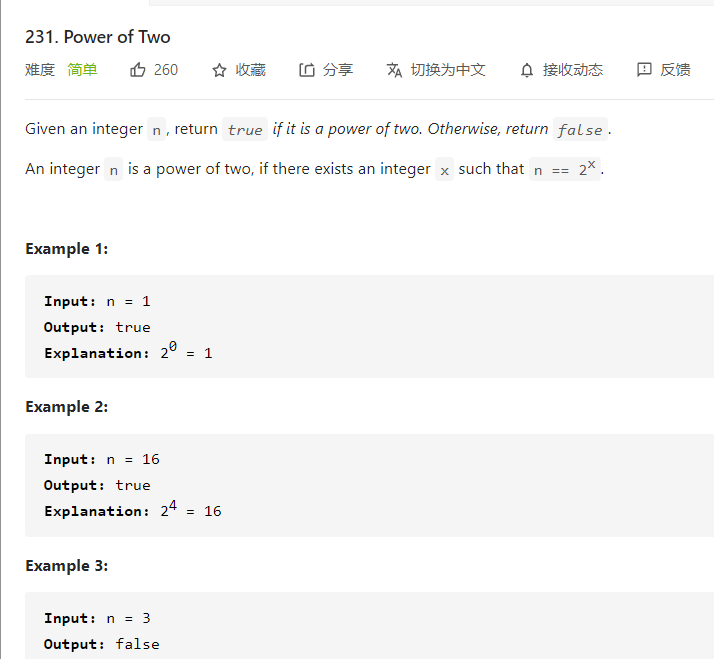
5.



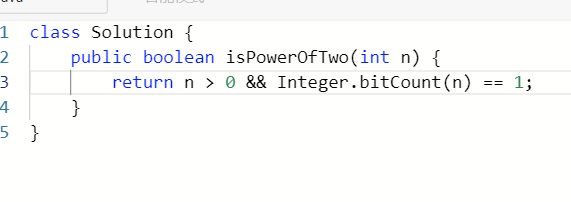
解法：用一个变量去从n的末尾记录下来，每次记录都要把变量往左移动一位来留给n的最低为，之后用or去得到n的最后一位



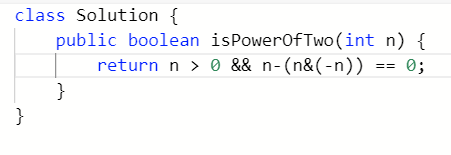
7.



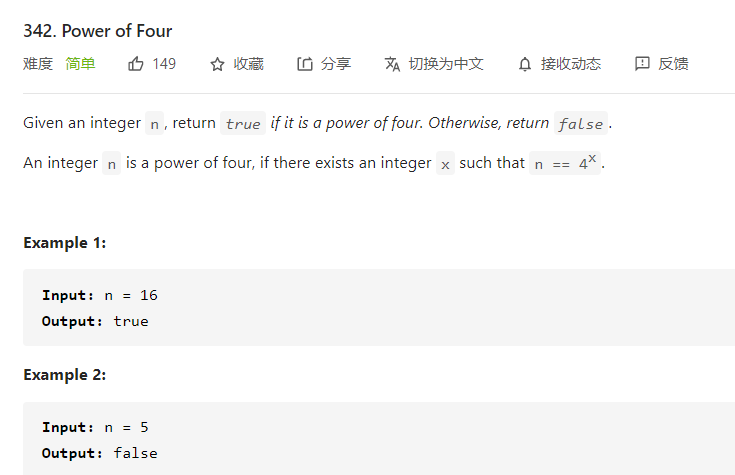
解法1：用Integer.bitCount()得到1的数量，因为是2的倍数所以只能有1个1



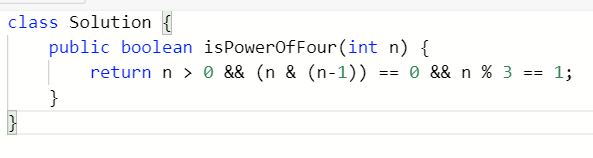
解法2：用n - (n&(-n)) 或者 n & (n -1)去掉n里面最低的那位1，看结果是不是0



8.



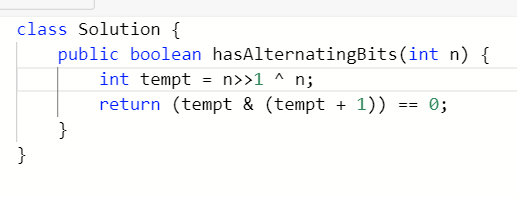
解法：4的倍数换算成2进制也只有一个1，而且永远是3的倍数+1



9.



解法：如果没有连续的0或1，则把n向左移动一位后得到的数与n是每一位都是1和0对位的，所以把他们作xor，则会获得1111，之后再用+1（得到0000）与自己作&看是不是0

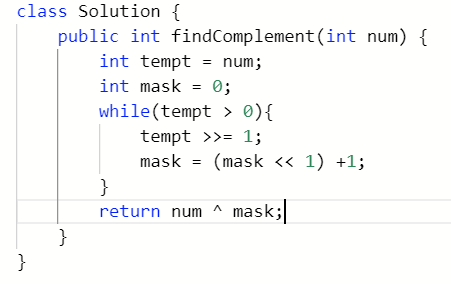


10.



题目：不考虑几bit 1就是1，2就是10

解法：用一个tempt得到和n一样位数的1111，再和n作xor得到补码



11.



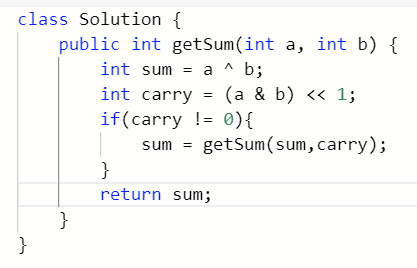
解法：把两个数相加变成他们不进位加起来的sum加上他们的carry

用^可以把两个数补进位加起来，

用&再左移一位可以得到carry

如果carry不为0则用递归把这个sum和carry加到carry=0为止

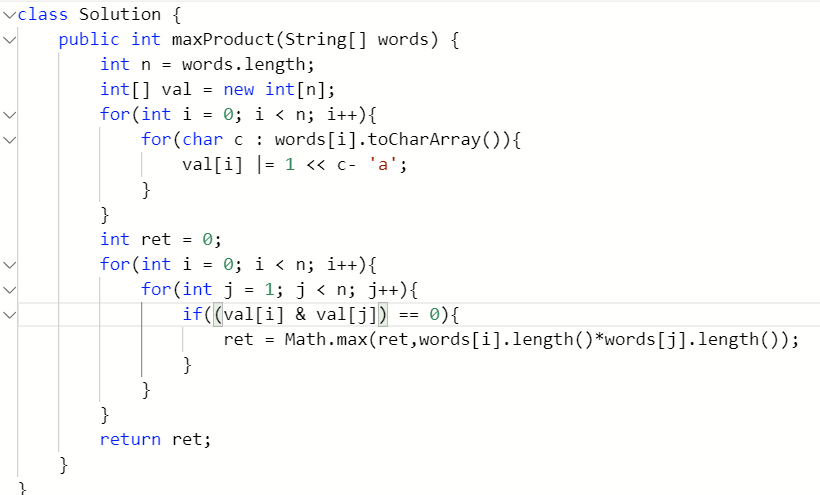




12.



解法：把每个word的字母以c- ‘a’的位数上位1放到一个array，这个array每个元素就是每个word的位数表示，因为都是小写字母，所以不会超过32位，用1<<每个字母值去作| 之后再去&每个word，如果结果是0说明没有字母重复



13.



题目：0到n的1的数量，第几个index就是几的1的数量

解法：如果i是偶数，则说明他和i/2的1的数量一样。因为本质就是bit都向左移动一位，如果i是奇数则先减1--变成偶数--当前index-1的1数量再+1

用i&(i-1)得到以上，因为如果i是奇数则去掉最末尾的1得到比他小1的偶数的1的数量，因为i比这个偶数多最后一位的1所以+1

如果i是偶数，则去掉最后一个1得到1的数量，再加（补)回来一个1

