Bit Manipulation

136. Single Number

Sort; HashMap; Bit Manipulation; Single Number

解法一：排序查找

解法二：HashSet保存单独数字，与全部总和比较

解法三：Bit Manipulation

参考

任何一个数字XOR自己都是0，一个数字XOR 0 得到的还是这个数字，所以如果把所有的数字都XOR的话最后得到的结果就是那个想要的Single Number。感觉Bit Manipulation真的很巧妙诶。

137. Single Number II

Bit Manipulation; Single Number

解法一：Bit Manipulation

参考

评论区有大神介绍了这种有一个数出现k次，其他数字出现p次 (p % k != 0) 的题的通用解法。本质就是利用bit manipulation和结合律和交换律。136. Single Number中的情况，我们只需要一个counter，是因为k等于2，而2的表现形式就是00, 01, 10，而且2正好是2的次方数，所以最后XOR会把出现次数为2的数字都去除成0。

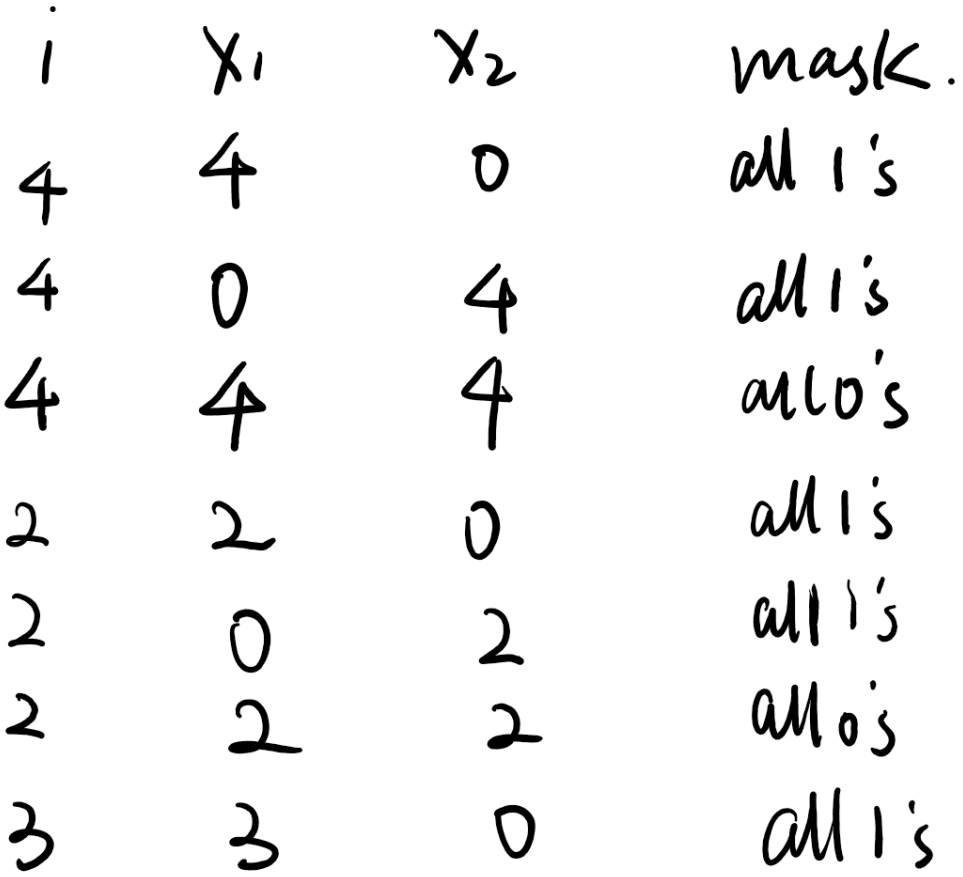
计数器可以看做是32位数每一位的记录，拼起来就是一整个数。

实际上我们需要m个，m > log2k（对应二进制的表达数量），接下来x1每次直接XOR这个数字，但是改动x2只在x1有记录的时候改，改动x3只在x2和x1均有记录的情况下改动，以此类推（类似二进制进位）。

同时我们需要一个mask，如果m不是正好是log2k的话，我们需要手动把计数器设置回0。当且仅当计数器的数量刚好是k个的时候mask需要把所有的数都调整回0。因此mask是~(?x1 & ?x2 & ?x3 .....)，每个bit前面的符号跟随k的二进制表示。这样任何有一个技术器的表达不符合k的二进制表示的时候，mask就会是1，全部符合的时候mask就是0。

这个题目里m是3，因此需要mask，且需要x1和x2。

这里有一个简单的例子（经过了排序，因为我们知道operations都是可以交换顺序的，就像排序了一样）。



190. Reverse Bits

Bit Manipulation

解法一：移动Bits

参考

其实就是一一把n右边的bits推进另一个数字里。具体的方法是建立一个数字ans，然后ans每次都左移，先假设新的这个bit是1。然后用n & 1得到n最右边的bit，用 | 来确定ans最右边的bit应该是什么，也不会改动前面的bit。然后n继续右移即可，因为是判断最右边的bit。

解法二：Integer.reverse(n)

参考

231. Power of Two

Bit Manipulation

解法一：不断除以2

解法二：Bit Manipulation

参考

任何2的次方都有一个特点就是2进制是1开头后面全是0，而且这个数减去1得到的数的2进制肯定全是1，且比原来的数少一位。所以如果对两个数取&的话肯定能得到0。

461. Hamming Distance

Bit Manipulation

解法一：XOR转二进制字符串数1的数量

解法二：XOR用t & (t - 1) 数1的数量