# 题目

# 分析

思路：

这个问题可以通过逐位检查和移动来找到比给定数字稍大和稍小的数字，要点是找到比给定数字稍大的数字时，找到从右到左第一个01组合，并将01交换为10，然后将右边所有的1移到最右边；找到比给定数字稍小的数字时，找到从右到左第一个10组合，并将10交换为01，然后将右边所有的1移到右边，再补齐右边的0。

具体步骤如下：

1、找到稍大的数字：从右到左遍历二进制表示，找到第一个01组合，将01交换为10，然后将右边所有的1移到最右边。

2、找到稍小的数字：从右到左遍历二进制表示，找到第一个10组合，将10交换为01，然后将右边所有的1移到右边，再补齐右边的0。

代码：

class Solution {

public:

vector<int> findClosedNumbers(int num) {

int nextGreater = findNextGreater(num);

int nextSmaller = findNextSmaller(num);

return {nextGreater, nextSmaller};

}

private:

int findNextGreater(int num) {

int c = num, c0 = 0, c1 = 0;

while ((c & 1) == 0 && c != 0) {

c0++;

c >>= 1;

}

while ((c & 1) == 1) {

c1++;

c >>= 1;

}

// 如果 num 是 0 或者二进制表示中只有一个1，找不到比它稍大的数字

if (c0 + c1 == 31 || c0 + c1 == 0) {

return -1;

}

int p = c0 + c1; // rightmost non-trailing zero

num |= (1 << p); // Flip rightmost non-trailing zero

num &= ~((1 << p) - 1); // Clear all bits to the right of p

num |= (1 << (c1 - 1)) - 1; // Insert (c1-1) ones on the right

return num;

}

int findNextSmaller(int num) {

int c = num, c0 = 0, c1 = 0;

while ((c & 1) == 1) {

c1++;

c >>= 1;

}

if (c == 0) {

return -1;

}

while ((c & 1) == 0 && c != 0) {

c0++;

c >>= 1;

}

int p = c0 + c1; // rightmost non-trailing one

num &= ((~0) << (p + 1)); // Clear from bit p onwards

int mask = (1 << (c1 + 1)) - 1; // Sequence of (c1+1) ones

num |= mask << (c0 - 1); // Insert (c1+1) ones on the right

return num;

}

};