



# 【华为OD机考 统一考试机试C卷】数字游戏 (C++ Java JavaScript Python C语言)

## 华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，**C卷真题已基本整理完毕**

抽到原题的概率为2/3到3/3，**也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。**

另外订阅专栏还可以联系笔者开通在线 [OJ](#) 进行刷题，提高刷题效率。

**真题目录：**华为OD机考机试 真题目录 (C卷 + D卷 + B卷 + A卷) + 考点说明

**专栏：**2023华为OD机试( B卷+C卷+D卷) (C++JavaJSPy)

**华为OD面试真题精选：**华为OD面试真题精选

**在线OJ：**[点击立即刷题](#)，模拟真实机考环境

## 题目描述

小扇和小船今天又玩起来了数字游戏，小船给小扇一个正整数  $n$  ( $1 \leq n \leq 1e9$ )，小扇需要找到一个比  $n$  大的数字  $m$ ，使得  $m$  和  $n$  对应的二进制中 1 的个数要相同，如：

- 4对应二进制100
- 8对应二进制1000
- 其中1的个数都为1个

现在求  $m$  的最小值。

## 输入描述

输入一个正整数  $n$  ( $1 \leq n \leq 1e9$ )

## 输出描述

输出一个正整数  $m$

## 用例

输入	2
输出	4
说明	2的二进制10，4的二进制位100，1的个数相同，且4是满足条件的最小数

输入	7
输出	11
说明	7的二进制111，11的二进制位1011，1的个数相同，且11是满足条件的最小数

解题思路

- 1. 理解题目要求：给定一个正整数  $n$ ，找到一个比  $n$  大的最小正整数  $m$ ，使得  $m$  和  $n$  的二进制表示中  $1$  的个数相同。
- 2. 观察二进制规律：在二进制数中，找到一个比当前数大的数，通常需要将一个较低位的  $0$  变成  $1$ 。同时，为了确保这个新的数尽可能小，我们希望这个  $0$  尽可能靠右，而且这个  $0$  右边的  $1$  尽可能少。
- 3. 找到关键位点：从低位到高位，找到第一个  $01$  模式的位置（即一个  $1$  后面紧跟着一个  $0$ ），这个  $01$  将被翻转成  $10$ 。这样做会增加数值，同时保持  $1$  的总数不变。
- 4. 翻转位：将找到的  $01$  模式翻转成  $10$ 。这可以通过将  $0$  的位置设为  $1$ （即  $n \mid= (1 \ll p)$ ），然后将该位右边的所有位清零（即  $n \&= \sim((1 \ll p) - 1)$ ）来实现。
- 5. 调整右侧位：由于我们已经将一个  $0$  变成了  $1$ ，为了保持  $1$  的总数不变，我们需要在右侧插入  $c1 - 1$  个  $1$ ，这里  $c1$  是  $01$  模式左侧  $1$  的数量。这可以通过  $n \mid= (1 \ll (c1 - 1)) - 1$  来实现。
- 6. 输入为4 4：对于输入  $n = 4$ ，其二进制表示为  $100$ 。按照上述步骤，我们需要找到第一个  $01$  模式，但是在  $100$  中不存在  $01$  模式。因此，我们需要在最低位添加一个  $1$ ，并且将最左边的  $1$  向右移动一位，得到  $1000$ ，即  $8$ 。这里  $8$  是比  $4$  大的下一个数，且二进制中  $1$  的数量相同。

C++

```
1 | #include <iostream>
2 | using namespace std;
3 | // 寻找下一个具有相同数量的1的数字
4 | int findNextNumberWithSameNumberOfOnes(int n) {
5 |     int c0 = 0, c1 = 0;
6 |     int temp = n;
7 |
8 | }
```

```
~
9 // 统计 "01" 模式中 0 的个数
10 while ((temp & 1) == 0 && temp != 0) {
11     c0++;
12     temp >>= 1;
13 }
14
15 // 统计 "01" 模式中 1 的个数
16 while ((temp & 1) == 1) {
17     c1++;
18     temp >>= 1;
19 }
20
21 // 如果 n 是形如 "111...11000...0" 的数, 则没有 "01" 模式
22 if (c0 + c1 == 31 || c0 + c1 == 0) {
23     return -1;
24 }
25
26 // p 是我们要翻转的 "01" 模式的位置
27 int p = c0 + c1;
28
29 // 翻转 "01" 为 "10"
30 n |= (1 << p); // 将 p 位置设为 1
31 n &= ~(1 << p) - 1; // 清除 p 位右边的所有位
32 n |= (1 << (c1 - 1)) - 1; // 在 p 位右边插入 (c1-1) 个 1
33
34 return n;
35 }
36
37 int main() {
38     int n;
39     cin >> n;
40     int m = findNextNumberWithSameNumberOfOnes(n);
41     cout << m << endl;
42     return 0;
}
```

## Java

```
1 import java.util.Scanner;
2
~
```

```
3 public class Main {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         int n = scanner.nextInt();
7         int m = findNextNumberWithSameNumberOfOnes(n);
8         System.out.println(m);
9         scanner.close();
10    }
11
12    private static int findNextNumberWithSameNumberOfOnes(int n) {
13        // c0 表示在找到的 "01" 模式中 0 的个数
14        // c1 表示在找到的 "01" 模式中 1 的个数
15        int c0 = 0, c1 = 0;
16        int temp = n;
17
18        // 统计 "01" 模式中 0 的个数
19        while (((temp & 1) == 0) && (temp != 0)) {
20            c0++;
21            temp >>= 1;
22        }
23
24        // 统计 "01" 模式中 1 的个数
25        while ((temp & 1) == 1) {
26            c1++;
27            temp >>= 1;
28        }
29
30        // 如果 n 是形如 "111...11000...0" 的数, 则没有 "01" 模式
31        if (c0 + c1 == 31 || c0 + c1 == 0) {
32            return -1;
33        }
34
35        // p 是我们要翻转的 "01" 模式的位置
36        int p = c0 + c1;
37
38        // 翻转 "01" 为 "10"
39        // 第一步: 将 p 位置为 1 (即将 "01" 的 "0" 翻转为 "1")
40        n |= (1 << p);
41
42        // 第二步: 清除 p 位右边的所有位 (即将 "01" 后面的所有位清零)
43    }
```

```
44 // 创建一个掩码, 它在 p 位之前都是 1, 然后取反, 得到 p 位及其右边都是 0 的掩码
45 int mask = ~(1 << p) - 1;
46 n &= mask;
47
48 // 第三步: 在 p 位右边插入 (c1-1) 个 1 (即将 "01" 前面的 "1" 移动到 p 位右边)
49 // 创建一个序列, 其中包含 (c1-1) 个 1, 然后将这个序列放在 p 位的右边
50 int ones = (1 << (c1 - 1)) - 1;
51 n |= ones;
52
53 return n;
54 }
```

## javaScript

```
1 // 寻找下一个具有相同数量的1的数字
2 function find(n) {
3     let c0 = 0, c1 = 0;
4     let temp = n;
5
6     // 统计 "01" 模式中 0 的个数
7     while ((temp & 1) === 0 && temp !== 0) {
8         c0++;
9         temp >>= 1;
10    }
11
12    // 统计 "01" 模式中 1 的个数
13    while ((temp & 1) === 1) {
14        c1++;
15        temp >>= 1;
16    }
17
18    // 如果 n 是形如 "111...11000...0" 的数, 则没有 "01" 模式
19    if (c0 + c1 === 31 || c0 + c1 === 0) {
20        return -1;
21    }
22
23    // p 是我们要翻转的 "01" 模式的位置
24    let p = c0 + c1;
25
```

```

26 // 翻转 "01" 为 "10"
27 n |= (1 << p); // 将 p 位置设为 1
28 n &= ~(1 << p) - 1; // 清除 p 位右边的所有位
29 n |= (1 << (c1 - 1)) - 1; // 在 p 位右边插入 (c1-1) 个 1
30
31 return n;
32 }
33
34 // 读取输入并输出结果
35 const readline = require('readline').createInterface({
36   input: process.stdin,
37   output: process.stdout
38 });
39
40 readline.on('line', (n) => {
41   const m = find(parseInt(n, 10));
42   console.log(m);
43   readline.close();
44 });

```

## Python

```

1 def find(n):
2     c0, c1 = 0, 0
3     temp = n
4
5     # 统计 "01" 模式中 0 的个数
6     while (temp & 1) == 0 and temp != 0:
7         c0 += 1
8         temp >>= 1
9
10    # 统计 "01" 模式中 1 的个数
11    while (temp & 1) == 1:
12        c1 += 1
13        temp >>= 1
14
15    # 如果 n 是形如 "111...11000...0" 的数, 则没有 "01" 模式
16    if c0 + c1 == 31 or c0 + c1 == 0:
17        return -1
18
19

```

```
19 # p 是我们要翻转的 "01" 模式的位置
20 p = c0 + c1
21
22 # 翻转 "01" 为 "10"
23 n |= (1 << p) # 将 p 位置设为 1
24 n &= ~(1 << p) - 1 # 清除 p 位右边的所有位
25 n |= (1 << (c1 - 1)) - 1 # 在 p 位右边插入 (c1-1) 个 1
26
27 return n
28
29 # 读取输入并输出结果
30 n = int(input())
31 m = find(n)
32 print(m)
```

## C语言

```
1 #include <stdio.h>
2
3 // 寻找下一个具有相同数量1的数字的函数
4 int findNextNumberWithSameNumberOfOnes(int n) {
5     int c0 = 0, c1 = 0; // 分别用于统计0和1的个数
6     int temp = n;
7
8     // 统计 "01" 模式中0的个数
9     while ((temp & 1) == 0 && temp != 0) {
10         c0++;
11         temp >>= 1;
12     }
13
14     // 统计 "01" 模式中1的个数
15     while ((temp & 1) == 1) {
16         c1++;
17         temp >>= 1;
18     }
19
20     // 如果n是形如 "111...11000...0" 的数, 则没有 "01" 模式
21     if (c0 + c1 == 31 || c0 + c1 == 0) {
22         return -1;
23     }
24 }
```



```
24
25 // p是要翻转的 "01" 模式的位置
26 int p = c0 + c1;
27
28 // 翻转 "01" 为 "10"
29 n |= (1 << p); // 将p位置设为1
30 n &= ~((1 << p) - 1); // 清除p位右边的所有位
31 n |= (1 << (c1 - 1)) - 1; // 在p位右边插入(c1-1)个1
32
33 return n;
34 }
35
36 int main() {
37     int n;
38     scanf("%d", &n); // 读取输入的整数n
39     int m = findNextNumberWithSameNumberOfOnes(n); // 调用函数查找m
40     printf("%d\n", m); // 输出结果m
41     return 0;
42 }
```

## 文章目录

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

题目描述

输入描述

输出描述

用例

解题思路

C++

Java

JavaScript

Python

C语言

# 机考真题 华为OD



CSDN @算法大师