1223 An Easy Problem

题目描述

给定一个正整数N，求最小的、比N大的正整数M，使得M与N的二进制表示中有相同数目的1。

举个例子，假如给定的N为78，其二进制表示为1001110，包含4个1，那么最小的比N大的并且二进制表示中只包含4个1的数是83，其二进制是1010011，因此83就是答案。

输入

输入若干行，每行一个数n(1≤n≤1000000)，输入"0"结束。

输出

输出若干行对应的值。

输入样例

```
1
2
3
4
78
0
```

输出样例

```
2
4
5
8
83
```

解析

从n+1项开始，计算当前数字中所包含2的数量是否等于目标值，如果是则跳出循环

编码

解法1

```
#include<bits/stdc++.h>

using namespace std;

//计算一个数字在二进制下1的数量
int countOneNum(int temp) {
    int cnt = 0;
    while (temp > 0) {
        if (temp % 2) {
            cnt++;
        }
        temp /= 2;
    }
    return cnt;
}

int main() {
    int n;
    //循环读入若干个数字
   while (scanf("%d", &n) != EOF && n) {
        //目标数字中2的数量
       int cnt = countOneNum(n);
        //从比n大1的数字开始暴力枚举，计算每一个数字中含有2的个数是否与n相同
       while (true) {
            //遍历比当前数字大的数
          int temp = ++n;
            //统计该数字的1的数量
          int sum = countOneNum(temp);
            //二者相等，则找到答案，进行输出。
          //并且终止循环
          if (cnt == sum) {
                cout << n << endl;
                break;
            }
        }
    }
    return 0;
}
```

解法2

```
#include<bits/stdc++.h>

using namespace std;

int main() {
```

```
    int n;
    //循环读入若干个数字
   while (scanf("%d", &n) != EOF && n) {
        //目标数字中2的数量
       int cnt = __builtin_popcount(n);
        //从比n大1的数字开始暴力枚举，计算每一个数字中含有2的个数是否与n相同
       while (true) {
            int temp  = ++n;
            //求目标数字含有的1的数量
          int sum =__builtin_popcount(temp);
            //二者相等则输出
          if (cnt == sum) {
                cout << n << endl;
                break;
            }
        }
    }
    return 0;
}
```

逻辑航线培优教育，信息学奥赛培训专家。

扫码添加作者获取更多内容。