给定一个正整数 n，找出小于或等于 n 的非负整数中，其二进制表示不包含 **连续的1** 的个数。

示例 1:

输入： 5
输出： 5
解释：
下面是带有相应二进制表示的非负整数<= 5：
0 : 0
1 : 1
2 : 10
3 : 11
4 : 100
5 : 101
其中，只有整数3违反规则（有两个连续的1），其他5个满足规则。

说明: 1 <= n <= 109

思路，我这里用的是暴力计算，把每个不是连续的1的数给计算进去，若是一位数，最后一位为1，那么他只能拼接0，如果为0，那么他可以拼接1或0，在这里可以用<<左移快速计算，需注意值的大小，是否符合小于n，还有当从0开始时，左移一直为0，需要把这个去除。

```
class Solution {
public:
    void dfs(int &num,int n,int target,bool flag){
        if(flag){
            if((n<<1)<=target&&(n<<1)!=n){
                num++;
                dfs(num,n<<1,target,true);
            }
            if(((n<<1)+1)<=target){
                num++;
                dfs(num,(n<<1)+1,target,false);
            }
        }else{
            if((n<<1)<=target){
                num++;
                dfs(num,n<<1,target,true);
            }
        }
    }
```

```
    int findIntegers(int num) {
        int result=1;
        dfs(result,0,num,true);
        return result;
    }
};
```