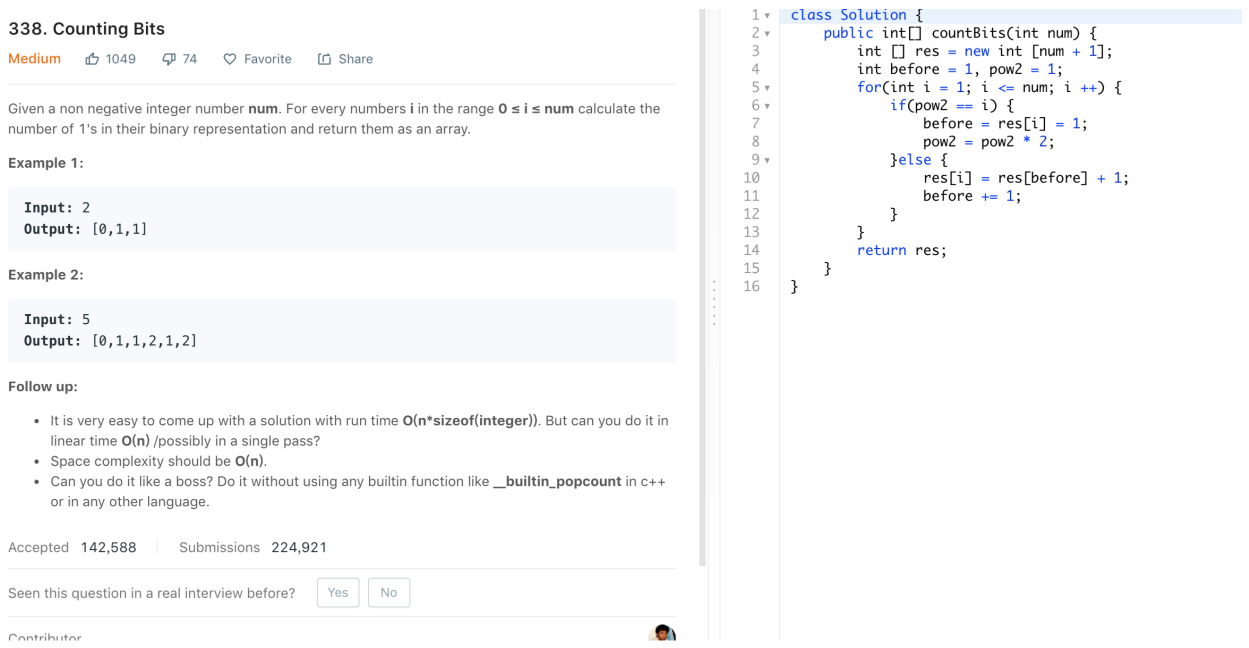


338. Counting Bits

Given a non negative integer number **num**. For every numbers **i** in the range **0 ≤ i ≤ num** calculate the number of 1's in their binary representation and return them as an array.

Example 1:

```
Input: 2
Output: [0,1,1]
```

Example 2:

```
Input: 5
Output: [0,1,1,2,1,2]
```

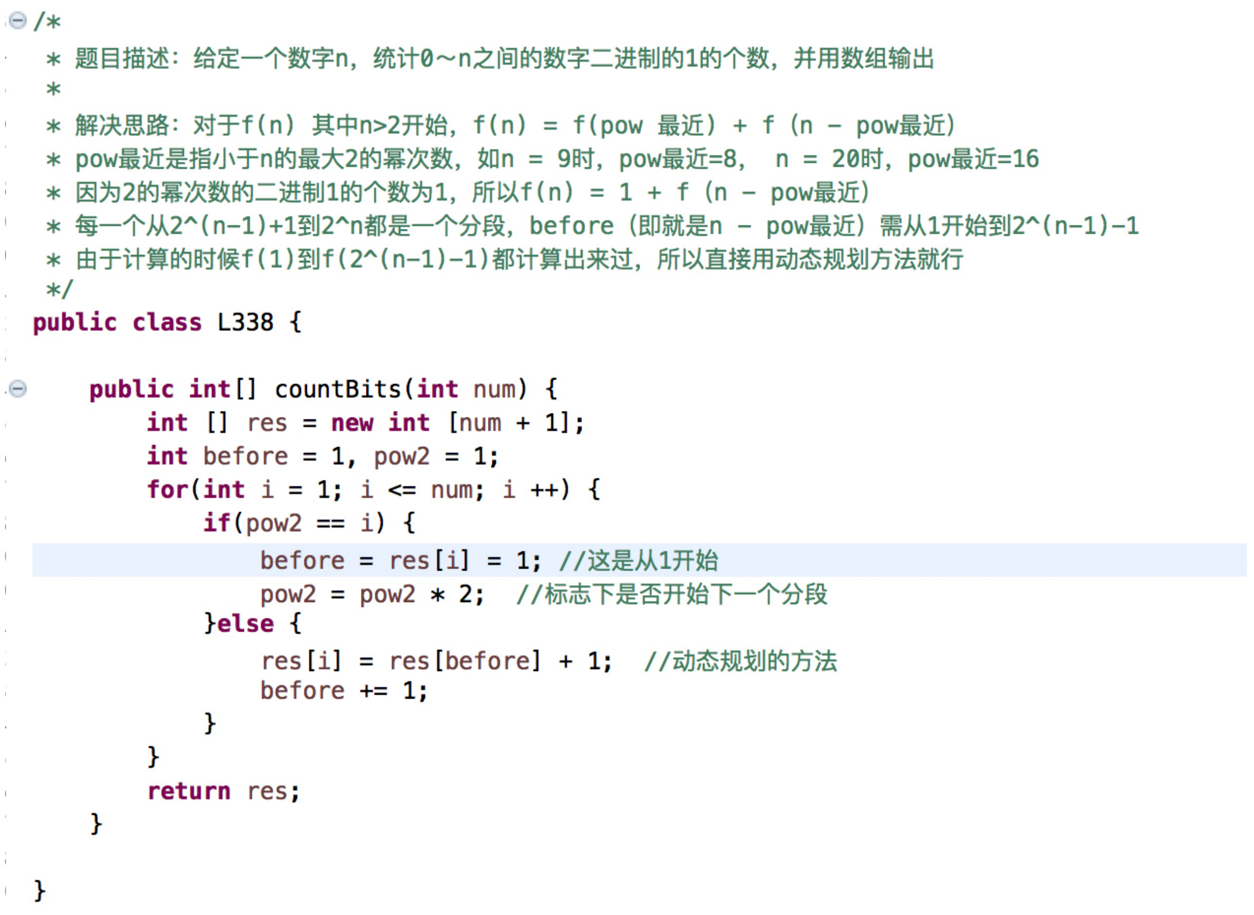
Follow up:

- It is very easy to come up with a solution with run time **O(n*sizeof(integer))**. But can you do it in linear time **O(n)** /possibly in a single pass?
- Space complexity should be **O(n)**.
- Can you do it like a boss? Do it without using any builtin function like **__builtin_popcount** in c++ or in any other language.

Accepted 142,588 | Submissions 224,921

Seen this question in a real interview before? Yes No

```
class Solution {
    public int[] countBits(int num) {
        int [] res = new int [num + 1];
        int before = 1, pow2 = 1;
        for(int i = 1; i <= num; i ++) {
            if(pow2 == i) {
                before = res[i] = 1;
                pow2 = pow2 * 2;
            }else {
                res[i] = res[before] + 1;
                before += 1;
            }
        }
        return res;
    }
}
```



```
/*
 * 题目描述：给定一个数字n，统计0~n之间的数字二进制的1的个数，并用数组输出
 *
 * 解决思路：对于f(n) 其中n>2开始，f(n) = f(pow 最近) + f (n - pow最近)
 * pow最近是指小于n的最大2的幂次数，如n = 9时，pow最近=8， n = 20时，pow最近=16
 * 因为2的幂次数的二进制1的个数为1，所以f(n) = 1 + f (n - pow最近)
 * 每一个从2^(n-1)+1到2^n都是一个分段，before (即就是n - pow最近) 需从1开始到2^(n-1)-1
 * 由于计算的时候f(1)到f(2^(n-1)-1)都计算出来过，所以直接用动态规划方法就行
 */
public class L338 {

    public int[] countBits(int num) {
        int [] res = new int [num + 1];
        int before = 1, pow2 = 1;
        for(int i = 1; i <= num; i ++) {
            if(pow2 == i) {
                before = res[i] = 1; //这是从1开始
                pow2 = pow2 * 2;  //标志下是否开始下一个分段
            }else {
                res[i] = res[before] + 1;  //动态规划的方法
                before += 1;
            }
        }
        return res;
    }

}
```