7 位操作型动规 Leetcode 338 Counting Bits

笔记本： DP Note
创建时间： 10/20/2019 10:13 PM **更新时间：** 10/20/2019 10:36 PM
作者： tanziqi1756@outlook.com

---

338. Counting Bits

难度 中等 190 收藏 分享 切换为中文 关注

题目描述 评论 (170) 题解(39) New 提交记录

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.

input 2
00 has 0 1
01 has 1 1
10 has 1 1
output 0 1 1

input 5
000 0
001 1
010 1

011 2
100 1
101 2
output 0 1 1 2 1 2

当然，如果用普通的求二进制方法，输入是N
时间复杂度为O(N*logN)

但是动态规划给我们提供了一个trick
0001 有1个
0010 有1个
0011 有2个
0100 有1个
0101 有2个
0110 有2个
0111 有3个
1000 有1个
1001 有2个
1010 有2个
1011 有3个
1100 有2个
如果我现在要计算1001，我只需要看除了最后一位，前面有多少个1即可;
如果最后一位是1，加一即可。
转移方程：f[i] = f[i>>1] + i mod 2

注意初始化

```java
class Solution {
    public int[] countBits(int num) {

        int[] dp = new int[num+1];
        dp[0] = 0;
        for( int i = 1; i < dp.length; i++ ) {
            dp[i] = dp[i>>1] + i % 2;
        }

        return dp;
    }
}
```