# 题目

给定一个非负整数n ，请计算0到n之间的每个数字的二进制表示中1的个数，并输出一个数组。

示例 1:

输入: n = 2

输出: [0,1,1]

解释:

0 --> 0

1 --> 1

2 --> 10

示例 2:

输入: n = 5

输出: [0,1,1,2,1,2]

解释:

0 --> 0

1 --> 1

2 --> 10

3 --> 11

4 --> 100

5 --> 101

说明 :

0 <= n <= 105

进阶:

给出时间复杂度为 O(n\*sizeof(integer)) 的解答非常容易。但你可以在线性时间 O(n) 内用一趟扫描做到吗？

要求算法的空间复杂度为 O(n) 。

你能进一步完善解法吗？要求在C++或任何其他语言中不使用任何内置函数（如C++中的 \_\_builtin\_popcount）来执行此操作。

注意：本题与主站338题相同：

https://leetcode-cn.com/problems/counting-bits/

# 分析

**方法一：位运算**

class Solution {

public:

vector<int> countBits(int num) {

vector<int> ret(num+1);

for(int i=1;i<=num;i++)

{

if(i & 1)

{

//如果是奇数,则相邻位必然为1,否则i就是偶数了

ret[i] = ret[i-1] + 1;

}

else

{

//如果是偶数,则相邻位不一定为1,需要右值计算

ret[i] = ret[i>>1];

}

}

return ret;

}

};

**方法二：动态规划**