比特位计数

给定一个非负整数 **num**。对于 **0 ≤ i ≤ num** 范围中的每个数字 **i** ，计算其二进制数中的 1 的数目并将它们作为数组返回。

示例 1:

输入： 2

输出： [0,1,1]

示例 2:

输入： 5

输出： [0,1,1,2,1,2]

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。

```
class Solution {
public:
    int countone(int x)
    {
        int sum=0;
        while(x>0)
        {
            x=x&(x-1);
            sum++;
        }
        return sum;
    }
    vector<int> countBits(int n) {
        vector<int> res;
        res.push_back(0);
        for(int i=1;i<=n;i++)
        {
            res.push_back(countone(i));
        }
        return res;
    }
};
```