338 比特位计数

Label：数学

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。

输入: 2
输出: [0,1,1]
输入: 5
输出: [0,1,1,2,1,2]

- API

```
class Solution {
    public int[] countBits(int num) {
        int[] array = new int[num+1];
        for (int i = 0; i <= num; i++) {
            array[i] = Integer.bitCount(i);
        }
        return array;
    }
}
```

- 最高位

```
class Solution {
    public int[] countBits(int num) {
        int[] bits = new int[num + 1];
        int highBit = 0;

        for (int i = 1; i <= num; i++) {
            if ((i & (i - 1)) == 0) {  //如果 是 2 的整数次幂，那么 & 为 0 ，负责如果按照顺序来计算的话，都为 1
                highBit = i;
            }
            bits[i] = bits[i - highBit] + 1;   // 比前一位只多了 1
        }
        return bits;
    }
}
```

- 跳跃动态规划

```
class Solution {
    public int[] countBits(int num) {
        int[] bits = new int[num + 1];
        for (int i = 1; i <= num; i++) {
            //  bits[i >> 1] 相当于跳着利用动态规划 i/2有几个1已经算了，并且他与x只相差一
个1
            bits[i] = bits[i >> 1] + (i & 1); //(i & 1)判断奇偶
        }
        return bits;
    }
}
```

- 动态规划 (最低设置位)

```
class Solution {
    public int[] countBits(int num) {
        int[] bits = new int[num + 1];
        for (int i = 1; i <= num; i++) {
            bits[i] = bits[i & (i - 1)] + 1;
        }
        return bits;
    }
}
```