题目描述

两个整数的 汉明距离 指的是这两个数字的二进制数对应位不同的数量。

计算一个数组中，任意两个数之间汉明距离的总和。

示例：

```
输入：4, 14, 2

输出：6

解释：在二进制表示中，4表示为0100，14表示为1110，2表示为0010。（这样表示是为了体现后四位之间关系）
所以答案为：
HammingDistance(4, 14) + HammingDistance(4, 2) + HammingDistance(14, 2) = 2 + 2 + 2 = 6.
```

注意:

1. 数组中元素的范围为从 `0` 到 `10^9`。
2. 数组的长度不超过 `10^4`。

题目解析

已示例为例，两两暴力计算的时间复杂度为o(n^2)，实现上肯定是没有问题，但是当数据量大的时候性能堪忧。

我们先将数组与结果的数字二进制写出来

```
 4    0 1 0 0
14    1 1 1 0
 2    0 0 1 0
HammingDistance(4, 14) = 1 0 1 0
HammingDistance(4, 2)  = 0 1 1 0
HammingDistance(14, 2) = 1 1 0 0
```

结合结果，从左往右按列观察这三个数字的二进制与运算结果的二进制可以发现一种关系：

数字个数 Count = 3

第一列： 0 1 0 ==> 1 * (3 -1) = 2 = 1 0 1

> *本列只有1个1，说明在所有数字的第一位中，有（Count - 1）个数字的第一位与* ***本数字*** *不同，也就是求距离的时候结果为1， 即这一位产生1的个数为1 * (3 -1)*

第二列： 1 1 0 ==> 2 * (3 -2) = 2 = 0 1 1

> *本列有2个1，说明在所有数字的第二位中，有（Count - 2）个数字的第二位与这* ***两个数字*** *不同，即这一位产生1的个数为（Count - 2）+ （Count - 2）= 2 *（3 - 2）*

第三列同第二列

第四列： 0 0 0 ==> 0 * (3 -0) = 0 = 0 0 0

> *本列所有数字相同，求距离时也就不会产生1， 结果为0*
>
> *如果是 1 1 1也一样，3 * (3 - 3)， 结果依旧为0*

总结： 每一列求距离产生1的个数 = 本列 1 的个数 * （数字个数 – 本列1的个数）= 本列 1 的个数 * 本列 0 的个数

动画理解

参考代码

```
class Solution {
    public int totalHammingDistance(int[] nums) {
        int len=nums.length;
        int[] bitCount = new int[32];
        if(len <= 1){
            return 0;
        }
        for(int numIndex = 0; numIndex < len; numIndex++){
            for(int bitIndex = 0; bitIndex < 32; bitIndex++){
                bitCount[bitIndex] += nums[numIndex] & 1;
                nums[numIndex] = nums[numIndex] >> 1;
                if(nums[numIndex] == 0){
                    break;
                }
            }
        }
        int oneCount = 0;
        for(int bitIndex = 0; bitIndex < 32; bitIndex++){
            oneCount += bitCount[bitIndex] * (len - bitCount[bitIndex]);
        }
        return oneCount;
    }
}
```

复杂度分析

时间复杂度：时间复杂度：O(N log C) 其中 C是常数，表示数组中数可能的最大值。

空间复杂度：O(log C)