# 题目

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:**

数组中元素的范围为从 0到 10^9。

数组的长度不超过 10^4。

# 分析

## 方法一：位运算

**思路：**

汉明距离等于两个数二进制表示中对应位置不同的数量。假设数组中的每个数都表示为 k 位的二进制数（高位补 0），那么我们可以发现，要计算数组中任意两个数的汉明距离的总和，可以先算出数组中任意两个数二进制第 i 位的汉明距离的总和，在将所有的 k 位之和相加。也就是说，二进制中的每一位都是可以独立计算的。

因此，我们考虑数组中每个数二进制的第 i 位，假设一共有 t 个 0 和 n - t 个 1，那么显然在第 i 位的汉明距离的总和为 t \* (n - t)。

由于所有的数都在 [0, 10^9] 的范围内，因此 k 最大为 31。我们只要计算出每一位上的汉明距离的总和，再相加即可。

**代码：**

class Solution {

public:

int totalHammingDistance(vector<int>& nums) {

if (nums.empty())

return 0;

int ret = 0;

vector<int> cnt(32, 0); // count of elements with a particular bit ON

for (auto num : nums) { // loop over every element

int i = 0;

while (num > 0) { // check every bit

cnt[i] += (num & 0x1);

num >>= 1;

i++;

}

}

for (int k=0;k<cnt.size();k++) { // loop over every bit count

ret += cnt.at(k) \* (nums.size() - cnt.at(k));

}

return ret;

}

};

复杂度分析

时间复杂度：O(NlogC)，其中C是常数，表示数组中数可能的最大值。

空间复杂度：O(logC)，也可以优化到O(1)，但可能会减少缓存命中，从而略微增加运行时间。