# 题目

给你一个整数数组arr。请你将数组中的元素按照其二进制表示中数字1的数目升序排序。

如果存在多个数字二进制中1的数目相同，则必须将它们按照数值大小升序排列。

请你返回排序后的数组。

示例 1：

输入：arr = [0,1,2,3,4,5,6,7,8]

输出：[0,1,2,4,8,3,5,6,7]

解释：[0] 是唯一一个有 0 个 1 的数。

[1,2,4,8] 都有 1 个 1 。

[3,5,6] 有 2 个 1 。

[7] 有 3 个 1 。

按照 1 的个数排序得到的结果数组为 [0,1,2,4,8,3,5,6,7]

示例 2：

输入：arr = [1024,512,256,128,64,32,16,8,4,2,1]

输出：[1,2,4,8,16,32,64,128,256,512,1024]

解释：数组中所有整数二进制下都只有 1 个 1 ，所以你需要按照数值大小将它们排序。

示例 3：

输入：arr = [10000,10000]

输出：[10000,10000]

示例 4：

输入：arr = [2,3,5,7,11,13,17,19]

输出：[2,3,5,17,7,11,13,19]

示例 5：

输入：arr = [10,100,1000,10000]

输出：[10,100,10000,1000]

提示：

1 <= arr.length <= 500

0 <= arr[i] <= 10^4

# 分析

## 方法一：暴力法

思路：

对每个十进制的数转二进制的时候统计二进制表示中的1的个数即可。

代码：

class Solution {

public:

static int bitCount(int n) {

int cnt = 0;

while (n) {

if (n & 1) cnt++;

n = n >> 1;

}

return cnt;

}

static int compare(int m, int n) {

int bitM = bitCount(m);

int bitN = bitCount(n);

//如果两个数的1数量相同，则直接比较数值大小

if (bitM == bitN) return m < n;

// 如果两个数的1数量不相同，则比较bit中1的数量大小

return bitM < bitN;

}

vector<int> sortByBits(vector<int>& arr) {

sort(arr.begin(), arr.end(), compare);

return arr;

}

};