# 题目

给你一个整数数组 nums ，按要求返回一个新数组 counts 。数组 counts 有该性质： counts[i] 的值是 nums[i] 右侧小于 nums[i] 的元素的数量。

示例 1：

输入：nums = [5,2,6,1]

输出：[2,1,1,0]

解释：

5 的右侧有 2 个更小的元素 (2 和 1)

2 的右侧仅有 1 个更小的元素 (1)

6 的右侧有 1 个更小的元素 (1)

1 的右侧有 0 个更小的元素

示例 2：

输入：nums = [-1]

输出：[0]

示例 3：

输入：nums = [-1,-1]

输出：[0,0]

提示：

1 <= nums.length <= 105

-104 <= nums[i] <= 104

# 分析

## 方法一：归并排序

class Solution {

public:

void mergeAndCount(vector<int>& nums, vector<int>& indexes, vector<int>& counts, int left, int mid, int right) {

int i = left;

int j = mid + 1;

int k = 0;

vector<int> newIndexes(right - left + 1);

while (i <= mid && j <= right) {

if (nums[indexes[i]] > nums[indexes[j]]) {

newIndexes[k] = indexes[j];

j++;

} else {

newIndexes[k] = indexes[i];

counts[indexes[i]] += (j - mid - 1);

i++;

}

k++;

}

while (i <= mid) {

newIndexes[k] = indexes[i];

counts[indexes[i]] += (j - mid - 1);

i++;

k++;

}

while (j <= right) {

newIndexes[k] = indexes[j];

j++;

k++;

}

for (int l = 0; l < k; l++) {

indexes[left + l] = newIndexes[l];

}

}

void mergeSort(vector<int>& nums, vector<int>& indexes, vector<int>& counts, int left, int right) {

if (left < right) {

int mid = left + (right - left) / 2;

mergeSort(nums, indexes, counts, left, mid);

mergeSort(nums, indexes, counts, mid + 1, right);

mergeAndCount(nums, indexes, counts, left, mid, right);

}

}

vector<int> countSmaller(vector<int>& nums) {

int n = nums.size();

vector<int> indexes(n, 0);

vector<int> counts(n, 0);

for (int i = 0; i < n; i++) {

indexes[i] = i;

}

mergeSort(nums, indexes, counts, 0, n - 1);

return counts;

}

};

## 方法二：