题目：

Given an array `nums`, write a function to move all `0`'s to the end of it while maintaining the relative order of the non-zero elements.

For example, given `nums = [0, 1, 0, 3, 12]`, after calling your function, `nums` should be `[1, 3, 12, 0, 0]`.

Note:

1. You must do this **in-place** without making a copy of the array.
2. Minimize the total number of operations.

Credits:

Special thanks to @jianchao.li.fighter for adding this problem and creating all test cases.

Subscribe to see which companies asked this question.

1.时间：O（N^2）;空间：O（1）

```
class Solution {
    /* 采用插入排序的思路 */
public:
    void moveZeroes(vector<int>& nums) {
```

```
        if (nums.size() < 2) return;
        for (int i = 1; i < nums.size(); ++i){
            int cur_num = nums[i];
            int k = i;
            while (k > 0 && nums[k - 1] == 0){
                nums[k] = nums[k - 1];
                k--;
            }
            nums[k] = cur_num;
        }
    }
};
```

2.时间：O（N）；空间：O（1）

```
class Solution {
    /* 维持两个指针 i，k，i 表示当前遍历的位置，0~k 存放的是非 0 */
public:
    void moveZeroes(vector<int>& nums) {
        if (nums.size() < 2) return;

        int notZeroIndex = 0;
        for (int i = 0; i < nums.size(); ++i){
            if (nums[i] != 0){ /* nums[i]不为 0 */
```

```
                    if (i != notZeroIndex){      /* nums[i]!=0 且 notZeroIndex != i，说明
此时 notZeroCount 位置是 0，于是交换两个数，这里
                                                由于 nums[notZeroCount]为 0 的原因，所
以不用 std::swap（减少一次赋值操作）*/
                        nums[notZeroIndex] = nums[i];
                        nums[i] = 0;
                    }
                    notZeroIndex++;
                }
            }


        }
};
```