

283. Move Zeroes

```
1 给定一个数组 nums，编写一个函数将所有 0 移动到数组的末尾，同时保持非零元素的相对顺序。
2
3 示例：
4
5 输入：[0,1,0,3,12]
6
7 输出：[1,3,12,0,0]
8
9 说明：
10 必须在原数组上操作，不能拷贝额外的数组。
11
12 尽量减少操作次数。
13
```

Analysis

- 暴力枚举，双循环，满足条件交换数据($O(n^2)$)，效率不高

```
1 class Solution {
2 public:
3     void moveZeroes(vector<int>& nums) {
4         for (int i = 0; i < (int)nums.size() - 1; i++) {
5             if (nums[i] == 0) {
6                 for(int j = i + 1; j < (int)nums.size(); j++) {
7                     if(nums[j] != 0) {
8                         swap(nums[i++], nums[j]);
9                     }
10                }
11            }
12        }
13    }
14};
```

- 两次循环 $O(n)$
- 第一次循环把非0的数放到指定位置，循环完成则记录下了最后一个非0数的后面一个位置
- 第二次循环是把之前记录的位置开始到数组结束的所有位置都置为0

```
1 class Solution {
2 public:
3     void moveZeroes(vector<int>& nums) {
4         int pos = 0;
5         for (int i = 0; i < (int)nums.size(); i++) {
6             if (nums[i] != 0) {
7                 nums[pos++] = nums[i];
8             }
9         }
10        for(;pos < nums.size(); pos++) {
```

```

12         nums[pos] = 0;
13     }
14 }
15 };

```

- **单循环，遇见非0就交换($O(n)$) – 推荐**

- 最差的情况时，如果全是非0的，每一个位置都会重新写一下自己位置的值

```

1 class Solution {
2 public:
3     void moveZeroes(vector<int>& nums) {
4         int pos = 0;
5         for (int i = 0; i < (int)nums.size(); i++) {
6             if (nums[i] != 0) {
7                 swap(nums[pos++], nums[i]);
8             }
9         }
10    }
11 };

```

- **使用stl里的stable_partition和自定义的compare函数 $O(n)$ – 推荐**

- 自定义一个compare函数来判断是否非0
- stable_partition会根据传入compare函数数组划分成2组，非0在前，0在后，并且保证每个组里的数的相对位置不变
- 如果使用partition的话，也是非0在前，0在后，但每组中的数的相对位置不保证和原来的一样。

```

1 class Solution {
2 public:
3     static bool compareNonZero(int x){
4         return (x != 0);
5     }
6     void moveZeroes(vector<int>& nums) {
7         stable_partition(nums.begin(), nums.end(), compareNonZero);
8     }
9 };

```

- **使用stl里的stable_partition和lambda表达式 $O(n)$ – 推荐**

- 原理同上，只是比较函数直接使用lambda表达式替代了自定义的compare函数。

```

1 class Solution {
2 public:
3     void moveZeroes(vector<int>& nums) {
4         stable_partition(nums.begin(), nums.end(), [](int x)
5         {return (x != 0);});
6     }
7 };

```

- **使用stl里的stable_partition和logical_not $O(n)$**

- 原理同上，只是比较函数直接使用stl里的logical_not表达式来判断是否是非0。
- 但这时需要反向遍历数组，因为logical_not是为0返回真，不为0返回假。

```

1 class Solution {
2 public:
3     void moveZeroes(vector<int>& nums) {
4         stable_partition(nums.rbegin(), nums.rend(), logical_not<int>());
5     }
6 };

```

Code

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 #include <functional>
5 using namespace std;
6
7 // double loop with swap O(nlongn)
8 // void moveZeroes(vector<int>& nums) {
9 //     for (int i = 0; i < (int)nums.size() - 1; i++) {
10 //         if (nums[i] == 0) {
11 //             for(int j = i + 1; j < (int)nums.size(); j++) {
12 //                 if(nums[j] != 0) {
13 //                     swap(nums[i++], nums[j]);
14 //                 }
15 //             }
16 //         }
17 //     }
18 // }
19
20 // first pos that should be set to 0 O(n)
21 // void moveZeroes(vector<int>& nums) {
22 //     int pos = 0;
23 //     for (int i = 0; i < (int)nums.size(); i++) {
24 //         if (nums[i] != 0) {
25 //             nums[pos++] = nums[i];
26 //         }
27 //     }
28 //     for(; pos < nums.size(); pos++) {
29 //         nums[pos] = 0;
30 //     }
31 // }
32
33 //last non-zero position with swap O(n)
34 // void moveZeroes(vector<int>& nums) {
35 //     int pos = 0;
36 //     for (int i = 0; i < (int)nums.size(); i++) {
37 //         if (nums[i] != 0 && i != pos) {
38 //             swap(nums[pos++], nums[i]);
39 //         }
40 //     }
41 // }
42
43 //using stl-stable_partition & compare function customsied O(n)
44 // bool compareNonZero(int x){

```

```

53 //     return (x !=0);
54 // }
55 // void moveZeroes(vector<int>& nums) {
56 //     stable_partition(nums.begin(), nums.end(), compareNonZero);
57 // }
58 //using stl-stable_partition & lambda O(n)
61 // void moveZeroes(vector<int>& nums) {
62 //     stable_partition(nums.begin(), nums.end(), [](int x)
63 // {return (x != 0);});
64 // }
65 //
67 //using stl-stable_partition and logical_not during reverse traversal
68 void moveZeroes(vector<int>& nums) {
69     stable_partition(nums.rbegin(), nums.rend(), logical_not<int>());
70 }
71 int main()
72 {
73     int nums[] = {0,1,0,3,12};
74     vector<int> array(nums, nums + 5);
75     moveZeroes(array);
76     //print
77     std::cout << "[" ;
78     for (int i = 0; i < (int)array.size(); i++) {
79         std::cout << array[i];
80         if (i != (int)array.size() - 1) {
81             std::cout << ",";
82         }
83     }
84     std::cout << "]" <<std::endl;
85     return 0;
86 }

```