

# 189. Rotate Array

```
1  给定一个数组，将数组中的元素向右移动 k 个位置，其中 k 是非负数。
2
3  示例 1:
4
5  输入: [1,2,3,4,5,6,7] 和 k = 3
6
7  输出: [5,6,7,1,2,3,4]
8
9  解释:
10 向右旋转 1 步: [7,1,2,3,4,5,6]
11 向右旋转 2 步: [6,7,1,2,3,4,5]
12 向右旋转 3 步: [5,6,7,1,2,3,4]
13
14 示例 2:
15
16 输入: [-1,-100,3,99] 和 k = 2
17
18 输出: [3,99,-1,-100]
19
20 解释:
21 向右旋转 1 步: [99,-1,-100,3]
22 向右旋转 2 步: [3,99,-1,-100]
23
24 说明:
25 尽可能想出更多的解决方案，至少有三种不同的方法可以解决这个问题。
    要求使用空间复杂度为  $O(1)$  的 原地 算法。
```

## Analysis

### 1.反转解法 $O(n)$

- 首先将所有元素反转。然后反转前 k 个元素，再反转后面 n-k个元素，就能 得到想要的结果。假设 n=7 且 k=3:

原始数组	: 1 2 3 4 5 6 7
反转所有数字后	: 7 6 5 4 3 2 1
反转前 k 个数字后	: 5 6 7 4 3 2 1
反转后 n-k 个数字后	: 5 6 7 1 2 3 4 --> 结果

- 所以可以自定义一个reverse函数，实现数组反转，在rotate函数里调用三次即可。

```
1  class Solution {
2  public:
3      void reverse(vector<int>& nums, int begin, int end) {
4          while(begin < end) {
5              swap(nums[begin++], nums[end--]);
6          }
7      }
8      void rotate(vector<int>& nums, int k) {
9          if(nums.empty()) { return; }
10         k %= (int)nums.size();
11         reverse(nums, 0, (int)nums.size() - 1);
```

```

13         reverse(nums, 0, k - 1);
14         reverse(nums, k, (int)nums.size() - 1);
15     }

```

## 2.反转解法(精简)-STL O(n) –推荐

- 原理同上，但使用STL里的reverse函数完成功能。

```

1 void rotate(vector<int>& nums, int k) {
2     if(nums.empty()) { return; }
3     k %= (int)nums.size();
4     reverse(nums.begin(), nums.end());
5     reverse(nums.begin(), nums.begin() + k);
6     reverse(nums.begin() + k, nums.end());
7 }

```

## 3.环状替代 + 最大公约数控制外循环次数 O(n) – 推荐

- 运用移位的方法来完成功能，对于一个长度为  $n$  的数组，整体移动  $k$  个位置：
  - 当  $n$  和  $k$  的最大公约数等于 1 的时候：1 次环状替代就可以完成交换；比如  $n = 5, k = 3$
  - 当  $n$  和  $k$  的最大公约数不等于 1 的时候：需要(最大公约数 $m$ ) 次环状替代完成交换；比如  $n = 4, k = 2$
- 在  $n$  和  $k$  的最大公约数不等于 1 的时候，例如：[A,B,C,D,E,F]此时 $n=6, k=4$ ，其最大公约数为 2 ,因此需要 2 次环状替代，这2次环状替代的数字正好把这个数组的数分成两个不相同的组：
  - 第 1 次循环（分组1）：A, E, C, A
  - 第 2 次循环（分组2）：B, F, D, B
- 最外层循环可以用最大公约数  $m$  来控制外循环，即完成换位的最外层循环次数为 $m$

```

1 class Solution {
2 public:
3     int gcd(int a, int b)
4     {
5         return b==0 ? a : gcd(b,a%b);
6     }
7     void rotate(vector<int>& nums, int k) {
8         if(nums.empty()) { return; }
9         k %= (int)nums.size();
10        int m = gcd(k, (int)nums.size());
11        for(int pos = 0; m > 0; pos++, m--) {
12            int current = pos;
13            int value = nums[current];
14            do {
15                int next = (current + k) % (int)nums.size();
16                swap(nums[next], value);
17                current = next;
18            } while (next != pos);
19        }
20    }
21 };

```

```

19         }while(current != pos);
20     }
21 }
22 };

```

#### 4.环状替代 + 最大公约数控制外循环次数 $O(n)$ – 推荐

- 环状替代原理同上
- 最外层循环可以也可以用移位次数来判断， **$n$ 个元素归位需要 $n$ 次交换**，所以每次移位计数即可

```

1  class Solution {
2  public:
3      void rotate(vector<int>& nums, int k) {
4          if(nums.empty()) { return; }
5          k %= (int)nums.size();
6          int count = 0;
7          for (int pos = 0; count < (int)nums.size(); pos++) {
8              int current = pos;
9              int value = nums[current];
10             do {
11                 int next = (current + k) % (int)nums.size();
12                 swap(nums[next], value);
13                 current = next;
14                 count++;
15             }while(current != pos);
16         }
17     }
18 };

```

#### Code

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  using namespace std;
5  //reverse function customized O(n)
6  // void reverse(vector<int>& nums, int begin, int end) {
7  //     while (begin < end) {
8  //         swap(nums[begin++], nums[end--]);
9  //     }
10 // }
11 // void rotate(vector<int>& nums, int k) {
12 //     if(nums.empty()) { return; }
13 //     k %= (int)nums.size();
14 //     reverse(nums, 0, (int)nums.size() - 1);
15 //     reverse(nums, 0, k - 1);
16 //     reverse(nums, k, (int)nums.size() - 1);
17 // }
18 //reverse function of stl O(n)

```

```

27 // void rotate(vector<int>& nums, int k) {
28 //     if(nums.empty()) { return; }
29 //     k %= (int)nums.size();
30 //     reverse(nums.begin(), nums.end());
31 //     reverse(nums.begin(), nums.begin() + k);
32 //     reverse(nums.begin() + k, nums.end());
33 // }
36 //ring substitution and using gcd
37 // int gcd(int m, int n) {
38 //     return n == 0? m : gcd(n, m % n);
39 // }
42 // void rotate(vector<int>& nums, int k) {
43 //     if(nums.empty()) { return; }
44 //     k %= (int)nums.size();
45 //     int m = gcd(k, (int)nums.size());
46 //     for (int pos = 0; m > 0; pos++, m--) {
47 //         int current = pos;
48 //         int value = nums[current];
49 //         do {
50 //             int next = (current + k) % (int)nums.size();
51 //             swap(nums[next], value);
52 //             current = next;
53 //         }while (current != pos);
54 //     }
55 // }
58 //ring substitution and count of exchanges
61 void rotate(vector<int>& nums, int k) {
62     if (nums.empty()) { return; }
63     k %= (int)nums.size();
64     int count = 0;
65     for(int pos = 0; count < (int)nums.size(); pos++) {
66         int current = pos;
67         int value = nums[current];
68         do {
69             int next = (current + k) % (int)nums.size();
70             swap(nums[next], value);
71             current = next;
72             count++;
73         }while (current != pos);
74     }
75 }
78 int main() {
79     int nums[] = {1,2,3,4,5,6,7};
80     std::vector<int> array(nums, nums + 7);
81     int k = 3;
82     // int nums[] = {-1,-100,3,99};
83     // std::vector<int> array(nums, nums + 4);
84     // int k = 2;
85     rotate(array, k);

```

```
93 //print
94
95 std::cout << "[" ;
96 for (int i = 0; i < (int)array.size(); i++) {
97     std::cout << array[i];
98     if (i != (int)array.size() - 1) {
99         std::cout << ",";
100     }
101 }
102 std::cout << "]" <<std::endl;
103 return 0;
104 }
105
106 }
```