189. Rotate Array

```
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。
示例 1:
输入: [1,2,3,4,5,6,7] 和 k = 3
输出: [5,6,7,1,2,3,4]
解释:
向右旋转 1 步: [7,1,2,3,4,5,6]
向右旋转 2 步: [6,7,1,2,3,4,5]
向右旋转 3 步: [5,6,7,1,2,3,4]
示例 2:
输入: [-1,-100,3,99] 和 k = 2
输出: [3,99,-1,-100]
解释:
向右旋转 1 步: [99,-1,-100,3]
向右旋转 2 步: [3,99,-1,-100]
说明:
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

- 所以可以自定义个一个reverse函数，实现数组反转，在rotate函数里调用三次即可。

```
class Solution {
public:
    void reverse(vector<int>& nums, int begin, int end) {
        while(begin < end) {
            swap(nums[begin++], nums[end--]);
        }
    }
    void rotate(vector<int>& nums, int k) {
        if(nums.empty()) { return; }
        k %= (int)nums.size();
        reverse(nums, 0, (int)nums.size() - 1);
```

```
        reverse(nums, 0, k - 1);
        reverse(nums, k, (int)nums.size() - 1);
    }
```

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。

```
void rotate(vector<int>& nums, int k) {
    if(nums.empty()) { return; }
    k %= (int)nums.size();
    reverse(nums.begin(), nums.end());
    reverse(nums.begin(), nums.begin() + k);
    reverse(nums.begin() + k, nums.end());
}
```

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

```
class Solution {
public:
    int gcd(int a, int b)
    {
        return b==0 ? a : gcd(b,a%b);
    }
    void rotate(vector<int>& nums, int k) {
        if(nums.empty()) { return; }
        k %= (int)nums.size();
        int m = gcd(k, (int)nums.size());
        for(int pos = 0; m > 0; pos++, m--) {
            int current = pos;
            int value = nums[current];
            do {
                int next = (current + k) % (int)nums.size();
                swap(nums[next], value);
                current = next;
```

```
            }while(current != pos);
        }
    }
};
```

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

```
class Solution {
public:
    void rotate(vector<int>& nums, int k) {
        if(nums.empty()) { return; }
        k %= (int)nums.size();
        int count = 0;
        for (int pos = 0; count < (int)nums.size(); pos++) {
            int current = pos;
            int value = nums[current];
            do {
                int next = (current + k) % (int)nums.size();
                swap(nums[next], value);
                current = next;
                count++;
            }while(current != pos);
        }
    }
};
```

Code

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
//reverse function customized O(n)
// void reverse(vector<int>& nums, int begin, int end) {
//     while (begin < end) {
//         swap(nums[begin++], nums[end--]);
//     }
// }
// void rotate(vector<int>& nums, int k) {
//     if(nums.empty()) { return; }
//     k %= (int)nums.size();
//     reverse(nums, 0, (int)nums.size() - 1);
//     reverse(nums, 0, k - 1);
//     reverse(nums, k, (int)nums.size() - 1);
// }
//reverse function of stl O(n)
```

```cpp
// void rotate(vector<int>& nums, int k) {
//     if(nums.empty()) { return; }
//     k %= (int)nums.size();
//     reverse(nums.begin(), nums.end());
//     reverse(nums.begin(), nums.begin() + k);
//     reverse(nums.begin() + k, nums.end());
// }
//ring substitution and using gcd
// int gcd(int m, int n) {
//     return n == 0? m : gcd(n, m % n);
// }
// void rotate(vector<int>& nums, int k) {
//     if(nums.empty()) { return; }
//     k %= (int)nums.size();
//     int m = gcd(k, (int)nums.size());
//     for (int pos = 0; m > 0; pos++, m--) {
//         int current = pos;
//         int value = nums[current];
//         do {
//             int next = (current + k) % (int)nums.size();
//             swap(nums[next], value);
//             current = next;
//         }while (current != pos);
//     }
// }
//ring substitution and count of exchanges
void rotate(vector<int>& nums, int k) {
    if (nums.empty()) { return; }
    k %= (int)nums.size();
    int count = 0;
    for(int pos = 0; count < (int)nums.size(); pos++) {
        int current = pos;
        int value = nums[current];
        do {
            int next = (current + k) % (int)nums.size();
            swap(nums[next], value);
            current = next;
            count++;
        }while (current != pos);
    }
}
int main() {
    int nums[] = {1,2,3,4,5,6,7};
    std::vector<int> array(nums, nums + 7);
    int k = 3;
    // int nums[] = {-1,-100,3,99};
    // std::vector<int> array(nums, nums + 4);
    // int k = 2;
    rotate(array, k);
```

```
    //print
    std::cout << "[" ;
    for (int i = 0; i < (int)array.size(); i++) {
        std::cout << array[i];
        if (i != (int)array.size() - 1) {
            std::cout << ",";
        }
    }
    std::cout << "]" <<std::endl;
    return 0;
}
```