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。

思路，标记1~k个数组，将其记录下来，然后用将1~n-k个数组中元素按K平移，在到n-k到n个数用原数组代替。（n-k也一样）

```
class Solution {
public:
    void rotate(vector<int>& nums, int k) {
        vector<int> num;
        int n=nums.size();
        k%=n;
        for(int i=0;i<n-k;i++){
            num.push_back(nums[i]);
        }
        for(int i=0;i<k;i++){
            nums[i]=nums[(i-k+n)%n];
        }
        for(int i=k;i<n;i++){
            nums[i]=num[i-k];
        }
```

```
    }
};
```