## 用空间复杂度o(1) 数组元素向左或右移n位

function reverse(nums, start, end) {

while (start < end) {

const temp = nums[start];

nums[start] = nums[end];

nums[end] = temp;

start++;

end--;

}

}

var rotate1 = function(nums, k) {

k %= nums.length;

reverse(nums, 0, nums.length - 1);

reverse(nums, 0, k - 1);

reverse(nums, k, nums.length - 1);

};

rotate1([1,2,3,4,5,6,7,8], 3)

1 2 3 4 5 6 7 8 🡪 8 7 6 5 4 3 2 1 🡪 6 7 8 1 2 3 4 5

很精妙这个

var rotate = function (nums, k) {

k = k % nums.length;

let count = 0;

for (let start = 0; count < nums.length; start++) {

let current = start;

let prev = nums[start];

do {

current = (current + k) % nums.length;

let temp = nums[current];

nums[current] = prev;

prev = temp;

count++;

} while (start != current);

}

}

rotate([1,2,3,4,5,6,7,8], 3)

1 2 3 4 5 6 7 8 🡪 1 2 3 1 5 6 7 8 🡪 1 2 3 1 5 6 4 8 🡪 1 7 3 1 5 6 4 8 🡪 1 7 3 1 2 6 4 8 🡪 1 7 3 1 2 6 4 5 🡪 1 7 8 1 2 6 4 5 🡪 1 7 8 1 2 3 4 5 🡪 6 7 8 1 2 3 4 5

当nums.length % 3 != 0时，for循环只执行一次，就能把所有的值循环到

当 nums.length % 3 == 0时，for循环执行3次，

方法3：每次只移一位，移n次

方法4：暂时没看懂

public void rotate\_3(int[] nums, int k) {

int n = nums.length;

k %= n;

// 第一次交换完毕后，前 k 位数字位置正确，后 n-k 位数字中最后 k 位数字顺序错误，继续交换

for (int start = 0; start < nums.length && k != 0; n -= k, start += k, k %= n) {

for (int i = 0; i < k; i++) {

swap(nums, start + i, nums.length - k + i);

}

}

}

public void rotate(int[] nums, int k) {

// 原理同上

recursiveSwap(nums, k, 0, nums.length);

}

private void recursiveSwap(int[] nums, int k, int start, int length) {

k %= length;

if (k != 0) {

for (int i = 0; i < k; i++) {

swap(nums, start + i, nums.length - k + i);

}

recursiveSwap(nums, k, start + k, length - k);

}

}

## 数组是否重复元素

方法1：放到set里，set.length==array.length

方法2：放到对象里，对象属性名为array的值