把一个数组旋转 k 步

例如： 一个数组 a 是 [1, 2, 3, 4, 5, 6, 7]，旋转 3 步结果是 [5, 6, 7, 1, 2, 3, 4]。

- 第一步，把最后一个数 7 放置于数组最前面。变成 [7, 1, 2, 3, 4, 5, 6]。
- 第二步，把最后一个数 6 放置于数组最前面。变成 [6，7, 1, 2, 3, 4, 5]。
- 第三步，把最后一个数 5 放置于数组最前面。变成 [5, 6, 7, 1, 2, 3, 4]。

有两种方法：

1. 数组拆分再组合

- 第一步，将其拆为 b 数组 [1, 2, 3, 4] 和 c 数组 [5, 6, 7] 两个数组
- 第二步，将两个数组 c.concat(b) 组合。

2. 数组删除最后一个元素再添加到头部。

- 将数组最后一个元素 7 删除 pop()，返回的是删除的元素 7，添加 unshift() 到数组的最前面。第一次返回 [7, 1, 2, 3, 4, 5, 6]。
- 重复 k 步。

```
// 第一种方法
function rotate1(arr, k) {
    if (!arr) {
        console.log("数组是必须的");
        return;
    }
    const len = arr.length;
    if (!len) return [];
    if (!k) return arr;
    if (typeof k !== 'number') {
        console.log("k 必须是一个数字");
        return;
    }
    // 时间复杂度 O(1)，空间复杂度 O(n)
    // 从计算量看。是截取和合并操作，3步，时间复杂度 O(1)
    // 从内存空间占用量看，不关系 k 的大小，关系 arr 的长度，不同的 arr 分别做三次操作，
所以空间复杂度是 O(n)
    const step = Math.abs(k % len); // 考虑 k 值大于数组长度情况
    const start = arr.slice(-step);
    const end = arr.slice(0, len - step);
    return start.concat(end);
}
```

```
// 第二种方法
function rotate2(arr, k) {
    if (!arr) {
        console.log("数组是必须的");
```

```
        return;
    }
    const len = arr.length;
    if (!len) return [];
    if (!k) return arr;
    if (typeof k !== 'number') {
        console.log("k 必须是一个数字");
        return;
    }
    // 时间复杂度 O(n^2)，空间复杂度 O(1)
    // 从计算量来看，循环一次是 O(n)，每一次 unshift() 操作是 O(n)，所以是 O(n^2)
    /*
     * pop() 和 push() 只是从数组末尾删元素和添加元素，不改变原来元素的位置。
     * shift() 和 unshift() 是在数组第一个位置删除和添加元素，原来的元素全部都要移位。
移位的多少和数组长度有关，这里是 O(n) 的复杂度，循环又是 O(n) 复杂度，所以是 O(n^2) 复杂
度
     */

    const step = Math.abs(k % len); // 考虑 k 值大于数组长度情况
    for (let i = 0; i < step; i++) {
        arr.unshift(arr.pop());
    }
    return arr;
}
```

```
let arr1 = [];
for (let i = 0; i < 20 * 10000; i++) {
    arr1.push(i);
}
console.time("rotate1");
rotate1(arr1, 100000);  // 2.072021484375 ms
console.timeEnd("rotate1");
```

```
let arr2 = [];
for (let i = 0; i < 20 * 10000; i++) {
    arr2.push(i);
}
console.time("rotate2");
rotate2(arr2, 100000);  // 2061.335205078125 ms,  比方法 1 时间相差了 1000 倍，显然
方法 1 更优
console.timeEnd("rotate2");
```

数组是一个有序结构，`unshift()`、`shift()`、`splice()` 操作都很慢。

`slice()` 操作很快，因为原数组没有改变。

优先选择时间复杂度更优的，如果时间复杂度一样，选择空间复杂度更优的。