

```

        && right[i + size - 1][j] >= size
        && down[i][j + size - 1] >= size) {
            return true;
        }
    }
    return false;
}

```

不包含本位置值的累乘数组

【题目】

给定一个整型数组 `arr`，返回不包含本位置值的累乘数组。

例如，`arr=[2,3,1,4]`，返回`[12,8,24,6]`，即除自己外，其他位置上的累乘。

【要求】

1. 时间复杂度为 $O(N)$ 。
2. 除需要返回的结果数组外，额外空间复杂度为 $O(1)$ 。

【进阶题目】

对时间和空间复杂度的要求不变，而且不可以使用除法。

【难度】

士 ★☆☆☆

【解答】

先介绍可以使用除法的实现，结果数组记为 `res`，所有数的乘积记为 `all`。如果数组中不含 0，则设置 `res[i]=all/arr[i](0<=i<n)` 即可。如果数组中有 1 个 0，对唯一的 `arr[i]==0` 的位置令 `res[i]=all`，其他位置上的值都是 0 即可。如果数组中 0 的数量大于 1，那么 `res` 所有位置上的值都是 0。具体过程请参看如下代码中的 `product1` 方法。

```

public int[] product1(int[] arr) {
    if (arr == null || arr.length < 2) {
        return null;
    }
    int count = 0;

```

```

int all = 1;
for (int i = 0; i != arr.length; i++) {
    if (arr[i] != 0) {
        all *= arr[i];
    } else {
        count++;
    }
}
int[] res = new int[arr.length];
if (count == 0) {
    for (int i = 0; i != arr.length; i++) {
        res[i] = all / arr[i];
    }
}
if (count == 1) {
    for (int i = 0; i != arr.length; i++) {
        if (arr[i] == 0) {
            res[i] = all;
        }
    }
}
return res;
}

```

不能使用除法的情况下，可以用以下方法实现进阶问题：

1. 生成两个长度和 `arr` 一样的新数组 `lr[]` 和 `rl[]`。`lr[]` 表示从左到右的累乘（即 `lr[i]=arr[0..i]`）的累乘。`rl` 表示从右到左的累乘（即 `rl[i]=arr[i..N-1]`）的累乘。
2. 一个位置上除去自己值的累乘，就是自己左边的累乘再乘以自己右边的累乘，即 `res[i]=lr[i-1]*rl[i+1]`。
3. 最左的位置和最右位置的累乘比较特殊，即 `res[0]=rl[1]`，`res[N-1]=lr[N-2]`。

以上思路虽然可以得到结果 `res`，但是除 `res` 之外，又使用了两个额外数组，怎么省掉这两个额外数组呢？可以通过 `res` 数组复用的方式。也就是说，先把 `res` 数组作为辅助计算的数组，然后把 `res` 调整成结果数组返回。具体过程请参看如下代码中的 `product2` 方法。

```

public static int[] product2(int[] arr) {
    if (arr == null || arr.length < 2) {
        return null;
    }
    int[] res = new int[arr.length];
    res[0] = arr[0];
    for (int i = 1; i < arr.length; i++) {
        res[i] = res[i - 1] * arr[i];
    }
    int tmp = 1;
    for (int i = arr.length - 1; i > 0; i--) {
        res[i] = res[i - 1] * tmp;
    }
}

```

```
        tmp *= arr[i];  
    }  
    res[0] = tmp;  
    return res;  
}
```

数组的 partition 调整

【题目】

给定一个有序数组 `arr`，调整 `arr` 使得这个数组的左半部分没有重复元素且升序，而不用保证右部分是否有序。

例如，`arr=[1,2,2,2,3,3,4,5,6,6,7,7,8,8,9]`，调整之后 `arr=[1,2,3,4,5,6,7,8,9,...]`。

【补充题目】

给定一个数组 `arr`，其中只可能含有 0、1、2 三个值，请实现 `arr` 的排序。

另一种问法为：有一个数组，其中只有红球、蓝球和黄球，请实现红球全放在数组的左边，蓝球放在中间，黄球放在右边。

另一种问法为：有一个数组，再给定一个值 k ，请实现比 k 小的数都放在数组的左边，等于 k 的数都放在数组的中间，比 k 大的数都放在数组的右边。

【要求】

1. 所有题目实现的时间复杂度为 $O(N)$ 。
2. 所有题目实现的额外空间复杂度为 $O(1)$ 。

【难度】

士 ★☆☆☆

【解答】

先来介绍原问题的解法：

1. 生成变量 u ，含义是在 `arr[0..u]` 上都是无重复元素且升序的。也就是说， u 是这个区域最后的位置，初始时 $u=0$ ，这个区域记为 A 。
2. 生成变量 i ，利用 i 做从左到右的遍历，在 `arr[u+1..i]` 上是不保证没有重复元素且升