```
                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=[1,2,2,2,3,3,4,5,6,6,7,7,8,8,8,9]，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

序的区域，*i* 是这个区域最后的位置，初始时 *i*=1，这个区域记为 B。

3．*i* 向右移动(*i*++)。因为数组整体有序，所以如果 arr[i]!=arr[u]，说明当前数 arr[i]应该加入到 A 区域里，所以交换 arr[u+1]和 arr[i]，此时 A 的区域增加一个数(*u*++)；如果 arr[i]==arr[u]，说明当前数 arr[i]的值之前已经加入到 A 区域，此时不用再加入。

4．重复步骤 3，直到所有的数遍历完。

具体请参看如下代码中的 leftUnique 方法。

```
public void leftUnique(int[] arr) {
        if (arr == null || arr.length < 2) {
                return;
        }
        int u = 0;
        int i = 1;
        while (i != arr.length) {
                if (arr[i++] != arr[u]) {
                        swap(arr, ++u, i - 1);
                }
        }
}
```

再来介绍补充问题的解法：

1．生成变量 left，含义是在 arr[0..left]（左区）上都是 0，left 是这个区域当前最右的位置，初始时 left 为-1。

2．生成变量 index，利用这个变量做从左到右的遍历，含义是在 arr[left+1..index]（中区）上都是 1，index 是这个区域的当前最右位置，初始时 index 为 0。

3．生成变量 right，含义是在 arr[right..N-1]（右区）上都是 2，right 是这个区域的当前最左位置，初始时 right 为 N。

4．index 表示遍历到 arr 的一个位置：

1）如果 arr[index]==1，这个值应该直接加入到中区，index++之后重复步骤 4。

2）如果 arr[index]==0，这个值应该加入到左区，arr[left+1]是中区最左的位置，所以把 arr[index]和 arr[left+1]交换之后，左区就扩大了，index++之后重复步骤 4。

3）如果 arr[index]==2，这个值应该加入到右区，arr[right-1]是右区最左边的数的左边，但也不属于中区，总之，在中区和右区的中间部分。把 arr[index]和 arr[right-1]交换之后，右区就向左扩大了(right--)，但是此时 arr[index]上的值未知，所以 index 不变，重复步骤 4。

5．当 index==right 时，说明中区和右区成功对接，三个区域都划分好后，过程停止。

遍历中的每一步，要么 index 增加，要么 right 减少，如果 index==right，过程就停止，

所以时间复杂度就是 $O(N)$，具体过程请参看如下代码中的 sort 方法。

```
public void sort(int[] arr) {
        if (arr == null || arr.length < 2) {
                return;
        }
        int left = -1;
        int index = 0;
        int right = arr.length;
        while (index < right) {
                if (arr[index] == 0) {
                        swap(arr, ++left, index++);
                } else if (arr[index] == 2) {
                        swap(arr, index, --right);
                } else {
                        index++;
                }
        }
}
```

# 求最短通路值

## 【题目】

用一个整型矩阵 matrix 表示一个网络，1 代表有路，0 代表无路，每一个位置只要不越界，都有上下左右 4 个方向，求从最左上角到最右下角的最短通路值。

例如，matrix 为：

1 0 1 1 1

1 0 1 0 1

1 1 1 0 1

0 0 0 0 1

通路只有一条，由 12 个 1 构成，所以返回 12。

## 【难度】

尉 ★★☆☆

## 【解答】

使用宽度优先遍历即可，如果矩阵大小为 $N \times M$，本文提供的方法的时间复杂度为 $O(N \times M)$，具体过程如下：