

```

        smallSum += s[i] * (right - j + 1);
        h[hi++] = s[i++];
    } else {
        h[hi++] = s[j++];
    }
}
for (; (j < right + 1) || (i < mid + 1); j++, i++) {
    h[hi++] = i > mid ? s[j] : s[i];
}
for (int k = 0; k != h.length; k++) {
    s[left++] = h[k];
}
return smallSum;
}

```

## 自然数数组的排序

### 【题目】

给定一个长度为  $N$  的整型数组 `arr`，其中有  $N$  个互不相等的自然数  $1 \sim N$ ，请实现 `arr` 的排序，但是不要把下标  $0 \sim N-1$  位置上的数通过直接赋值的方式替换成  $1 \sim N$ 。

### 【要求】

时间复杂度为  $O(N)$ ，额外空间复杂度为  $O(1)$ 。

### 【难度】

士 ★☆☆☆

### 【解答】

`arr` 在调整之后应该是下标从  $0$  到  $N-1$  的位置上依次放着  $1 \sim N$ ，即 `arr[index]=index+1`。

本书提供两种实现方法，先介绍方法一：

1. 从左到右遍历 `arr`，假设当前遍历到  $i$  位置。
2. 如果 `arr[i]==i+1`，说明当前的位置不需要调整，继续遍历下一个位置。
3. 如果 `arr[i]!=i+1`，说明此时  $i$  位置的数 `arr[i]` 不应该放在  $i$  位置上，接下来将进行跳的过程。

举例来说明，比如 `[1,2,5,3,4]`，假设遍历到位置 2，也就是 5 这个数。5 应该放在位置 4 上，所以把 5 放过去，数组变成 `[1,2,5,3,5]`。同时，4 这个数是被 5 替下来的数，应该放在位置 3，所以把 4 放过去，数组变成 `[1,2,5,4,5]`。同时 3 这个数是被 4 替下来的数，应该放

在位置 2，所以把 3 放过去，数组变成[1,2,3,4,5]。当跳了一圈回到原位置后，会发现此时  $\text{arr}[i] == i + 1$ ，继续遍历下一个位置。

方法一的具体过程请参看如下代码中的 `sort1` 方法。

```
public void sort1(int[] arr) {
    int tmp = 0;
    int next = 0;
    for (int i = 0; i != arr.length; i++) {
        tmp = arr[i];
        while (arr[i] != i + 1) {
            next = arr[tmp - 1];
            arr[tmp - 1] = tmp;
            tmp = next;
        }
    }
}
```

下面介绍方法二：

1. 从左到右遍历 `arr`，假设当前遍历到  $i$  位置。
2. 如果  $\text{arr}[i] == i + 1$ ，说明当前的位置不需要调整，继续遍历下一个位置。
3. 如果  $\text{arr}[i] \neq i + 1$ ，说明此时  $i$  位置的数 `arr[i]` 不应该放在  $i$  位置上，接下来将在  $i$  位置进行交换过程。

比如[1,2,5,3,4]，假设遍历到位置 2，也就是 5 这个数。5 应该放在位置 4 上，所以位置 4 上的数 4 和 5 交换，数组变成[1,2,4,3,5]。但此时还是  $\text{arr}[2] \neq 3$ ，4 这个数应该放在位置 3 上，所以 3 和 4 交换，数组变成[1,2,3,4,5]。此时  $\text{arr}[2] == 3$ ，遍历下一个位置。

方法二的具体过程请参看如下代码中的 `sort2` 方法。

```
public void sort2(int[] arr) {
    int tmp = 0;
    for (int i = 0; i != arr.length; i++) {
        while (arr[i] != i + 1) {
            tmp = arr[arr[i] - 1];
            arr[arr[i] - 1] = arr[i];
            arr[i] = tmp;
        }
    }
}
```