链滴

归并排序: 912. 排序数组

作者：superstonne
原文链接：https://ld246.com/article/1566834147902
来源网站：链滴

给定一个整数数组 nums，将该数组升序排列。

示例 1：

输入：[5,2,3,1]
输出：[1,2,3,5]

示例 2：

输入：[5,1,1,2,0,0]
输出：[0,0,1,1,2,5]

提示：

1 <= A.length <= 10000
-50000 <= A[i] <= 50000

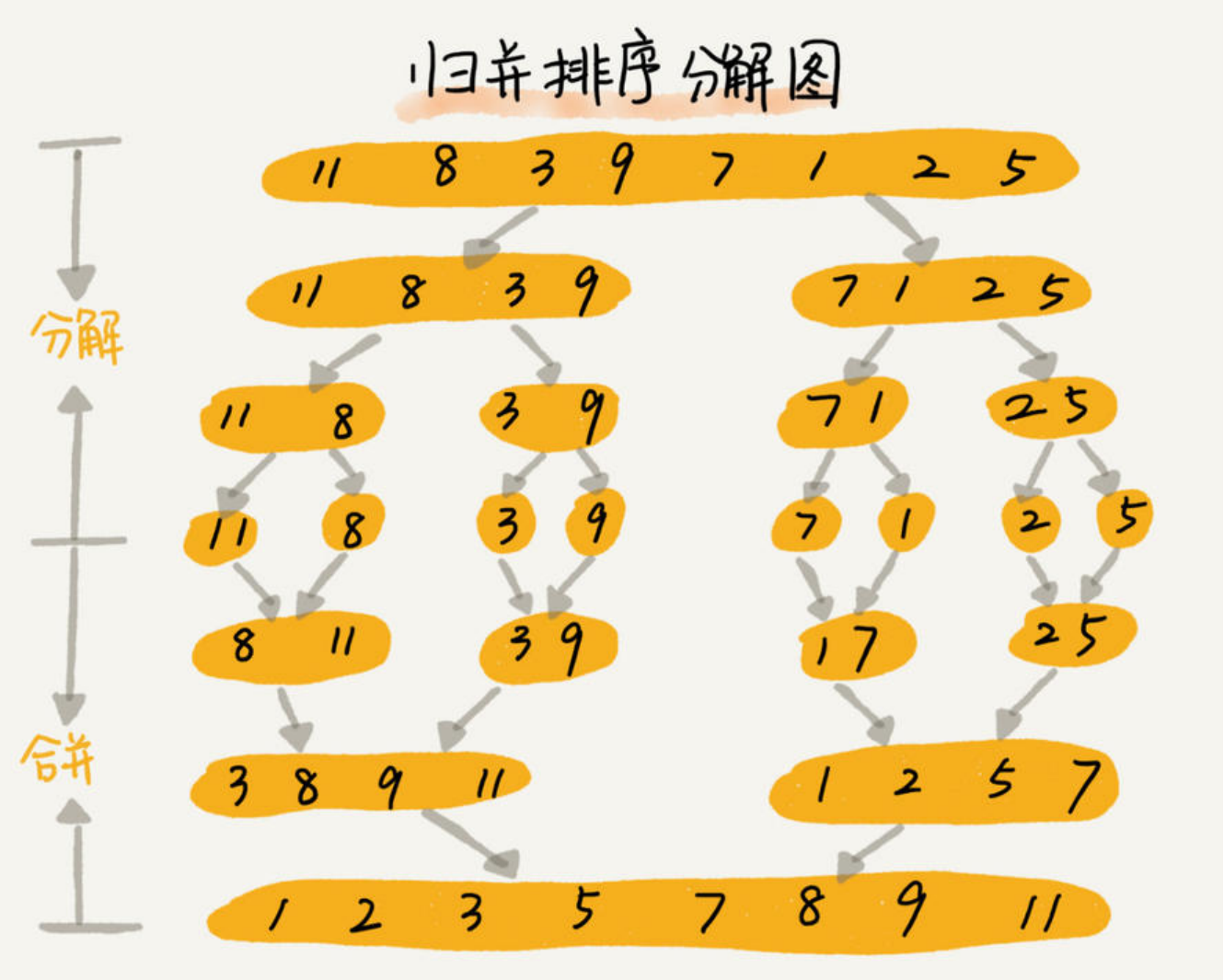
用归并排序解决此题，时间复杂度为 nlogn

```
class Solution {
    public int[] sortArray(int[] nums) {
        mergeSort(nums, 0, nums.length - 1);
        return nums;
    }

    public void mergeSort(int[] array, int startIndex, int endIndex) {
```

```
    if (startIndex >= endIndex) {
        return;
    }
    // 获取最中间的元素索引
    int middleIndex = (startIndex + endIndex) / 2;

    // 左半边排序
    mergeSort(array, startIndex, middleIndex);

    // 右半边排序
    mergeSort(array, middleIndex + 1, endIndex);

// 合并左右排序结果
    merge(array, startIndex, endIndex);
}

public void merge(int[] array, int startIndex, int endIndex) {
// 零时数组用来存放排好序的元素，因此归并排序的空间复杂度为 O（n）
    int[] tmp = new int[endIndex - startIndex + 1];
    int middleIndex = (startIndex + endIndex) / 2;
    int i = startIndex;
    int j = middleIndex + 1;
    int k = 0;
    while (i <= middleIndex && j <= endIndex) {
// 从此处可以看出归并排序为原地排序，不会打乱本来的顺序
        if (array[i] <= array[j]) {
            tmp[k++] = array[i++];
        } else {
            tmp[k++] = array[j++];
```

```
            }
        }


        int start = i;
        int end = middleIndex;
        if (i > middleIndex) {
            start = j;
            end = endIndex;
        }
        while (start <= end) {
            tmp[k++] = array[start++];
        }
    // 将零时数组中的元素转入原数组
        for (int m = 0; m < tmp.length; m++, startIndex++) {
            array[startIndex] = tmp[m];
        }
    }
}
```

归并排序图解



原文链接：归并排序: 912. 排序数组