```
function mergeSort (data) {
    if (data.length < 2) {
        return data;
    }

    let middle = Math.floor(data.length / 2);

    return merge(mergeSort(data.slice(0, middle)),
mergeSort(data.slice(middle)));
}

function merge (left, right) {
    let merged = [];

    let leftLength = left.length;
    let rightLength = right.length;

    let leftPointer = 0;
    let rightPointer = 0;

    while (leftPointer < leftLength && rightPointer < rightLength) {
        if (left[leftPointer] ⩽ right[rightPointer]) {
            merged[merged.length] = left[leftPointer++];
        } else {
            merged[merged.length] = right[rightPointer++];
        }
    }

    // 后面两个 while 是处理当左或右序列为空时，而另一个序列不为空的情况
    // 比如左边已经空了，但是右边里面还有元素，那就把右边的元素一次性推到合并数组中
    // 以下两个 while 语句可以用很多其他相同功能的语句替代：
    // merged = merged.concat(left.slice(leftPointer),
right.slice(rightPointer));
    // 或：
    // merged = [...merged, ...left.slice(leftPointer),
...right.slice(rightPointer)];

    while (left.length > 0 && leftPointer < left.length) {
        merged[merged.length] = left[leftPointer++];
    }

    while (right.length > 0 && rightPointer < right.length) {
        merged[merged.length] = right[rightPointer++];
    }

    return merged;
}
```