**P39 2-10**

设计：将待排序的数组分成大小大致相同的√n个子数组

分别对子数组进行排序，再将排好序的子数组合并成排好序的数组

算法描述：

1. 若数组中元素仅有一个则直接返回

2. 计算√n的值和各个√n等分点的索引

3. 递归地对√n个子数组分别调用归并排序算法

4. 将√n个数组合并成新数组

5. 返回新数组

复杂度分析：见右图

**P40 2-1**

算法实现（环境：JavaScript + node.js 14.16）：

const fs = require('fs')

let data = fs.readFileSync("./input.txt", "utf-8").split(' ')

let res = [], maxFreq = 1

let m = new Map()

for (let i = 1; i <= data.length; i++) {

    m.set(data[i], m.get(data[i]) + 1 || 1)

}

m.forEach((v, e) => {

    if (v > maxFreq) {

        maxFreq = v

        res = [e]

    } else if (v === maxFreq) {

        res.push(e)

    }

})

res = res.join(' ').toString()

fs.writeFileSync("./output.txt", `${maxFreq}\n${res}`)

**作业：**

分析：初始状态下将最大值max和最小值min分别设定为数组两端的较大值和较小值，从两端开始向中间遍历整个数组，两端值中较大的与max进行比较，较小的与min进行比较，若对应值大于max/小于min则更新对应的值，每次循环需比较三次，共需(n / 2 – 2)次比较(第一次无需比较)，总时间复杂度为(3 \* (n / 2) - 2)

算法实现：

arr[0] > arr[arr.length - 1] ? {max = arr.unshift(), min = arr.pop()} : {max = arr.pop(), min = arr.unshift()}

for (let i = 0, j = arr.length; i <= j; i++, j--) {

    if (arr[i] < arr[j]) {

        if (arr[i] < min) min = arr[i]

        if (arr[j] > max) max = arr[j]

    } else {

        if (arr[i] > max) max = arr[i]

        if (arr[j] < min) min = arr[j]

    }

}

复杂度分析：遍历过程中每次循环可能执行1或2个操作，最坏情况下（数组单调递减）需执行(2n – 1)次，最好情况下（数组单调递增）需执行(n - 1)次，由于数组数据不定，平均情况下需((2n - 1) + (n - 1)) / 2 = (3n - 2) / 2次符合题意