**JavaScript计数排序算法**

1. 计数排序算法

计数排序（Counting sort）是一种稳定的线性时间排序算法。该算法于1954年由 Harold H. Seward 提出。计数排序使用一个额外的数组，数组的下标对应待排序的数字。然后根据新数组的下标来获得正确的顺序。就像给每个位置按数字顺序做好标记，然后把对应数组放入其中，最后把下标打印出来即可。计数排序只适合数量较少的正整数排序。其算法复杂度接近于：O(N + K)

步骤是：

1. 找到待排序中最大和最小的元素；
2. 新建一个计数数组，长度为最大与最小值的差值+1；
3. 遍历待排序数组，将数字与计数数组下标对应，按出现次数做标记；做对应时为了减少数组长度，可以用下标减去最小值；
4. 反向取出数据，按出现的次数逐个追加到输出数组中。
5. 计数排序算法执行过程分析：

**待排序数组：**

3, 4, 15, 3, 3, 7, 10

3, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1

计数数组

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

取出数组

3, 3, 3, 4, 7, 10, 15

1. 计数排序代码标准版

function countingSort(arr) {

let countList = []

let min = Math.min.apply(null, arr)

let max = Math.max.apply(null, arr)

// 计数数组的长度在最大和最小差值+1

countList.length = max - min + 1

for (var i = 0, l = arr.length; i < l; i++) {

// 把下标减去min值，以便减少计数数组的长度

let idx = arr[i] - min

if (!countList[idx]) {

countList[idx] = 1

} else {

// 多个相同的数字则需要多个标记

countList[idx] += 1

}

}

let output = []

// 遍历计数数组

countList.forEach(

(val, i) => {

// 下标若大于0，则取出来，如果相同则取多次

for (var j = 0; j < val; j++) {

if (val && val > 0) {

// 取出的值要+min还原

output.push(i + min)

}

}

}

)

return output

}

countingSort([3, 4, 15, 3, 3, 7, 10])

// [3, 3, 3, 4, 7, 10, 15]

1. 计数排序代码实现简版

function countingSort(arr) {

let countList = []

// 用一个计数器来计数，长度为数组长度+1

countList.length = Math.max.apply(null, arr) + 1

// 根据待排序项给对应下标的位置增加标记，多个相同的则需要递增

// 仅适合正整数, 如果是负数和小数就不可以

for (var i = 0, l = arr.length; i < l; i++) {

// 填充0或者增加1

countList[arr[i]] = countList[arr[i]] || 0

countList[arr[i]] += 1

}

console.log(countList)

let output = []

// 遍历全部

countList.forEach(

(val, i) => {

// 下标若大于0，则取出来，如果相同则取多次

for (var j = 0; j < val; j++) {

if (val && val > 0) {

output.push(i)

}

}

}

)

return output

}

countingSort([3, 4, 15, 3, 3, 7, 10])

// [3, 3, 3, 4, 7, 10, 15]