**JavaScript桶排序算法**

1. 桶排序算法

桶排序（Bucket sort）也称箱排序，是一个[排序算法](https://zh.wikipedia.org/wiki/%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95)，工作的原理是将[数组](https://zh.wikipedia.org/wiki/%E9%99%A3%E5%88%97)分到几个桶里，桶的数量可由排序数组最大值与最小值关系决定，可以固定几个桶。每个桶内再通过插入、冒泡或简单方式进行排序。其算法复杂度接近于：O(N)

步骤是：

1. 得到桶的数量，比如以最大项减去最小项再除以最小项；
2. 新建一个桶列表，然后遍历数组，再将数组项除以桶的个数得到要存放的桶的下标，将数组项存入到对应桶中；
3. 存入到桶中时，按顺序插入，保持顺序；
4. 数据全部放入桶之后，再遍历桶列表，将二维数组按顺序展开取出即可。
5. 桶排序算法执行过程分析：

**待排序数组：**

3, 4.3, 15, -2.1, 3.2, 3, -7, 10

**[3, 3, 3.2, 4.3], empty, [10], [15], [-2.1], [-7]**

桶数组

0, 1, 2, 3, 4 (最大最小值得到桶数为负，故按默认5个桶

取出数组

3, 3, 3.2, 4.3, 10, 15

1. 桶排序代码JS简版

function bucketSort(arr) {

let max = Math.max(arr)

let min = Math.min(arr)

// 设置桶的数量，可以任意设置，也可以根据最大、最小值来设置

let bucketNumber

bucketNumber = Math.floor((max - min) / min + 1)

bucketNumber = bucketNumber > 5 ? bucketNumber : 5

let buckets = [], idx

arr.forEach( (item, i) => {

// 当前项除以桶数取整，决定应该放在某个桶内

idx = Math.floor(item / bucketNumber)

if (!buckets[idx]) {

buckets[idx] = []

buckets[idx].push(item)

} else {

let len = buckets[idx].length

while (len--) {

// 自后往前遍历，如果数字大于数组中的项目则插入其后

if (item > buckets[idx][len]) {

buckets[idx].splice(len + 1, 0, item)

break

// 如果数字最小则插入到最前

} else if (len === 0){

buckets[idx].unshift(item)

}

}

}

})

// 按照下标和顺序取出桶内的数字，回填到一个数组中

let output = []

buckets.forEach( (bucketItem) => {

if (bucketItem) {

output = output.concat(bucketItem)

}

})

return output

}

bucketSort([3, 4.3, 15, -2.1, 3.2, 3, -7, 10])

1. 桶排序标准版，支持负数

function bucketSort(arr) {

let max = arr[0]

let min = arr[0]

// 得到最大与最小值

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

if (arr[i] > max) {

max = arr[i]

}

if (arr[i] < min) {

min = arr[i]

}

}

// 设置桶的数量，可以任意设置，也可以根据最大小值来设置

let bucketNumber

bucketNumber = Math.floor((max - min) / min + 1)

bucketNumber = bucketNumber > 5 ? bucketNumber : 5

// 桶列表与桶下标

let buckets = [], idx, item

// 支持小数，如果要支持负数，则需要将负数下标单独存取，见注释

let negativeIdxList = []

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

// 排序数字除以桶数得到下标，以此来决定分配到哪个桶里面

item = arr[i]

idx = Math.floor(item / bucketNumber)

// 如果要支持负数则需要添加到负数下标列表中去

if (idx < 0) {

negativeIdxList.push(idx)

}

// 每个桶都是一个数组，若不存在则创建并放入数字

if (!buckets[idx]) {

buckets[idx] = []

buckets[idx].push(item)

} else {

// 如果桶内已有数字，先留出最后一个空位，然后按照顺序插入

let len = buckets[idx].length

buckets[idx].length = len + 1

while (len--) {

if (item < buckets[idx][len]) {

//如果插入的数字小于当前项，则将当前项逐个往后挪

buckets[idx][len + 1] = buckets[idx][len]

} else {

// 如果大于等于当前项，则插入到当前项后面，循环中止

buckets[idx][len + 1] = item

break

}

}

}

}

// 按照下标和顺序取出桶内的数字，回填到数组中

let output = []

// 支持负数，先将负数下标排序，然后根据顺序逐个按属性取出

negativeIdxList.sort( (a, b) => { return a - b } )

for (let i = 0, len = negativeIdxList.length; i < len; i++) {

let bucketItem = buckets[negativeIdxList[i]]

if (bucketItem) {

let j = 0, bucketItemLen = bucketItem.length

while (j < bucketItemLen) {

output.push(bucketItem[j])

j++

}

}

}

// 把数组内容取出

for (let i = 0, len = buckets.length; i < len; i++) {

let bucketItem = buckets[i]

if (bucketItem) {

let j = 0, bucketItemLen = bucketItem.length

while (j < bucketItemLen) {

output.push(bucketItem[j])

j++

}

}

}

return output

}

bucketSort([3, 4.3, 15, -2.1, 3.2, 3, -7, 10])