**JavaScript快速排序递归与非递归实现**

1. 快速排序概述

快速排序(Quiksort)是一种通过基准划分区块并不断交换左右项的排序方式，其采用了分治法，减少了交换的次数。平均算法复杂度：O(N logN)。

步骤是：

1. 先找到一个基准点(pivot)，便于理解一般是位于数组中间的那一项。
2. 逐个循环数组将小于基准的项放左侧，将大于基准的数放右侧。一般通过交换的方式来实现。
3. 将基点左侧全部项和基点右侧全部项分别通过递归(或遍历)形式重复第1项，直到所有数组都交换完成。

执行过程分析：

**数列： 4, 1,** **3, 5, 2**

4, 1, 3, 5, 2

2, 1, 3, 5, 4

2, 1, 3, 5, 4

2, 1,

5, 4

3

1, 2

4, 5

1. 快速排序实现
   1. 递归新建数组版本。无需交换，每个分区都是新数组。

function quickSort(arr) {

if (arr.length <= 1) {

return arr

}

var pivot, left = [], right = []

var midIndex = Math.floor(arr.length / 2)

pivot = arr[midIndex]  
  
 console.log('sort begin:', arr, midIndex, pivot)

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

if (midIndex === i) continue

if (arr[i] < pivot) {

left.push(arr[i])

} else {

right.push(arr[i])

}

}

return quickSort(left).concat(pivot, quickSort(right))

}

var arr = [4, 1, 3, 5, 2]

quickSort(arr)  
  


* 1. 标准递归版本。需要交换，无需新建数组。

function quickSort(arr, left, right) {

var i = left = left ? left : 0

var j = right = right ? right : arr.length - 1

var midIndex = Math.floor((i + j) / 2)

var pivot = arr[midIndex]

var tmp

while (i <= j) {

while (arr[i] < pivot) {

i++

}

while (arr[j] > pivot) {

j--

}

if (i <= j) {

tmp = arr[i]

arr[i] = arr[j]

arr[j] = tmp

i++

j--

}

}

if (left < j) {

quickSort(arr, left, j)

}

if (i < right) {

quickSort(arr, i, right)

}

return arr

}

var arr = [4, 1, 3, 5, 2]

quickSort(arr)  


* 1. 非递归版本。需要交换，无需新建数组，利用stack或queue遍历。

function quickSort(arr, left, right) {

left = left ? left : 0

right = right ? right : arr.length - 1

var stack = [], i, j, midIndex, pivot, tmp

stack.push(left)

stack.push(right)

while (stack.length) {

j = right = stack.pop()

i = left = stack.pop()

midIndex = Math.floor((i + j)/2)

pivot = arr[midIndex]

while (i <= j) {

while (arr[i] < pivot) {

i++

}

while (arr[j] > pivot) {

j--

}

if (i <= j) {

tmp = arr[j]

arr[j] = arr[i]

arr[i] = tmp

i++

j--

}

}

if (left < j) {

stack.push(left)

stack.push(j)

}

if (i < right) {

stack.push(i)

stack.push(right)

}

}

return arr

}

quickSort([2, 1, 3, 5, 4])

源码：

function quickSort(arr) {

if (arr.length <= 1) {

return arr

}

var pivot, left = [], right = []

var midIndex = Math.floor(arr.length / 2)

pivot = arr[midIndex]

console.log('sort begin:', arr, left, right, midIndex, pivot)

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

if (midIndex === i) continue

if (arr[i] < pivot) {

left.push(arr[i])

} else {

right.push(arr[i])

}

}

console.log('sorted:', arr, left, right)

//return quickSort(left).concat(pivot, quickSort(right))

return quickSort(left).toString() + '(' + pivot + ')' + quickSort(right).toString()

}

var arr = [4, 1, 3, 5, 2]

quickSort(arr)

****

function quickSort(arr, left, right) {

var i = left = left ? left : 0

var j = right = right ? right : arr.length - 1

var midIndex = Math.floor((i + j) / 2)

var pivot = arr[midIndex]

var tmp

while (i <= j) {

while (arr[i] < pivot) {

i++

}

while (arr[j] > pivot) {

j--

}

if (i <= j) {

tmp = arr[i]

arr[i] = arr[j]

arr[j] = tmp

i++

j--

}

}

if (left < j) {

quickSort(arr, left, j)

}

if (i < right) {

quickSort(arr, i, right)

}

return arr

}

var arr = [4, 1, 3, 5, 2]

quickSort(arr)

function quickSort(arr, left, right) {

left = left ? left : 0

right = right ? right : arr.length - 1

var stack = [], i, j, midIndex, pivot, tmp

stack.push(left)

stack.push(right)

while (stack.length) {

j = right = stack.pop()

i = left = stack.pop()

midIndex = Math.floor((i + j)/2)

pivot = arr[midIndex]

while (i <= j) {

while (arr[i] < pivot) {

i++

}

while (arr[j] > pivot) {

j--

}

if (i <= j) {

tmp = arr[j]

arr[j] = arr[i]

arr[i] = tmp

i++

j--

}

}

if (left < j) {

stack.push(left)

stack.push(j)

}

if (i < right) {

stack.push(i)

stack.push(right)

}

}

return arr

}

quickSort([2, 1, 3, 5, 4])