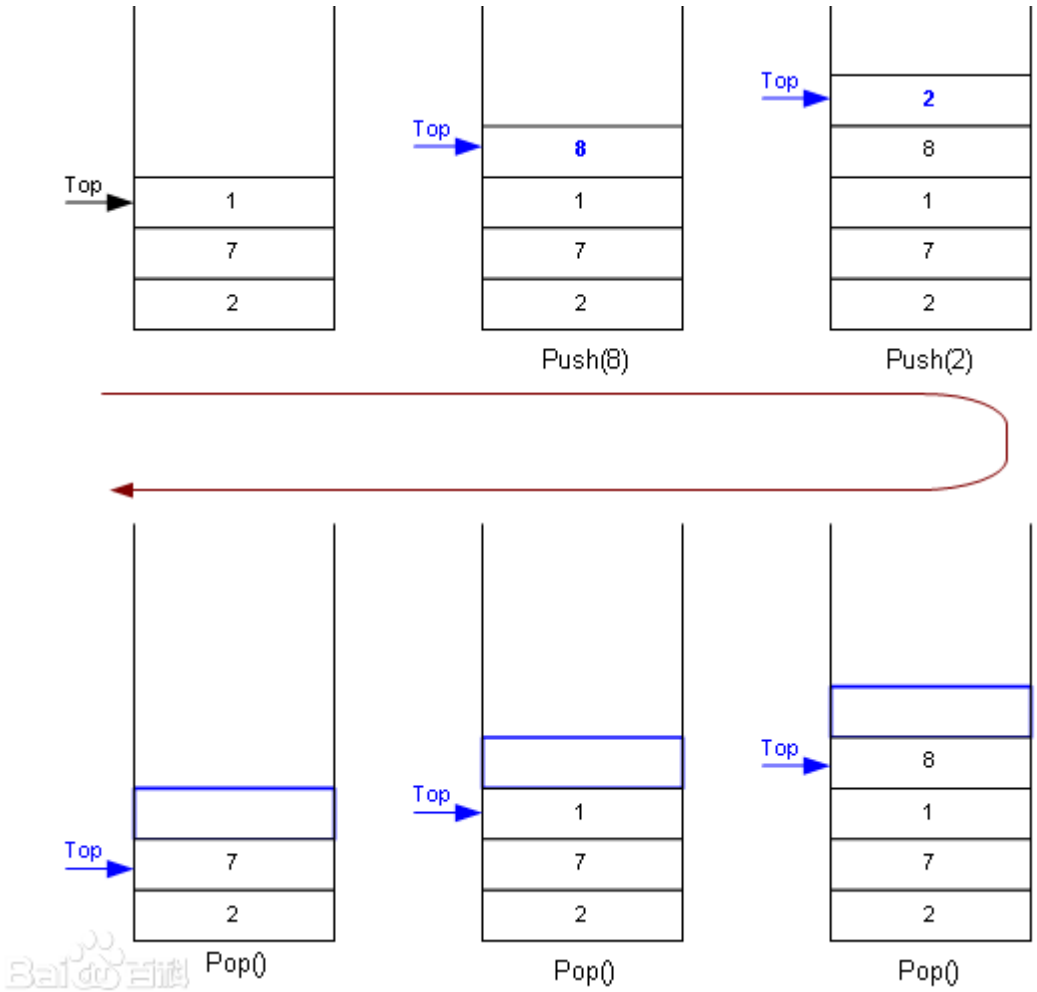
algorithm

algorithm with javascript

** 目录

栈

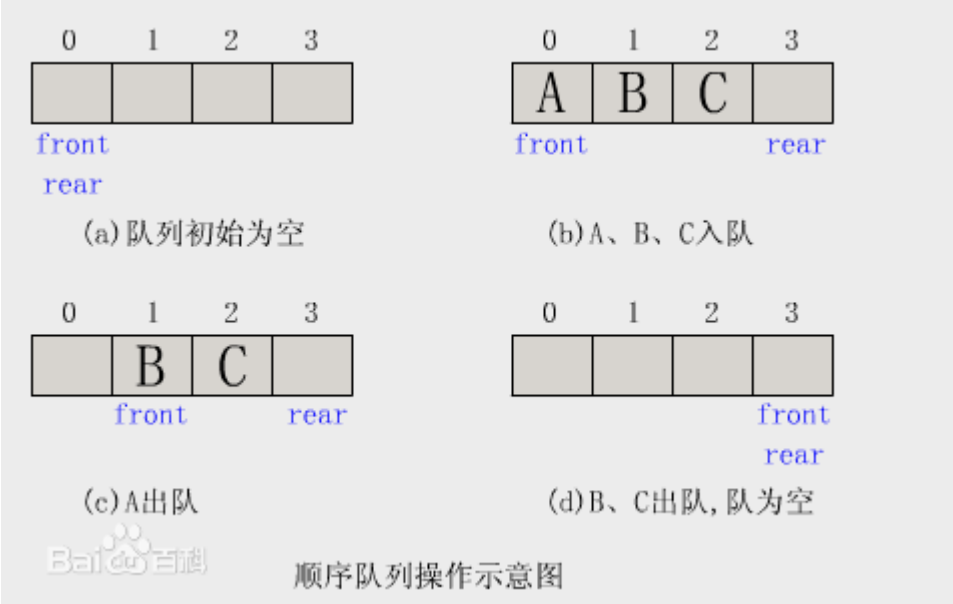
栈是一种特殊的数据结构，栈内元素只能从一端访问。先后栈的元素最先出栈 LIFO



eg：数制转换，括号匹配，回文数等

队列

队列元素只能从一端入队，另一端出队，即FIFO



eg：基数排序-最快的排序算法

排序算法

冒泡排序

最慢的排序算法之一，最容易实现，最方便写代码的排序算法。

实现过程（从小到大排序为例）：相邻数据之间不停比较，若左边大于右边，则左右数据交换位置，数据值像气泡一样从数组的一端“漂浮”到另一端。

选择排序

相比于冒泡快了一些，但因为速度和写法简便程度上都不是很优秀，不常用。

实现过程（从小到大）：每次选出数组中的最小元素，放在最前，再从剩余元素中继续找出最小元素，不断进行下去，直至排序结束。

插入排序

最符合人类习惯的排序算法，例如，斗地主时整理纸牌顺序。

实现过程（从小到大）：在一个有序数组中，找到合适位置插入新的元素，直至所有元素都插入。

归并排序

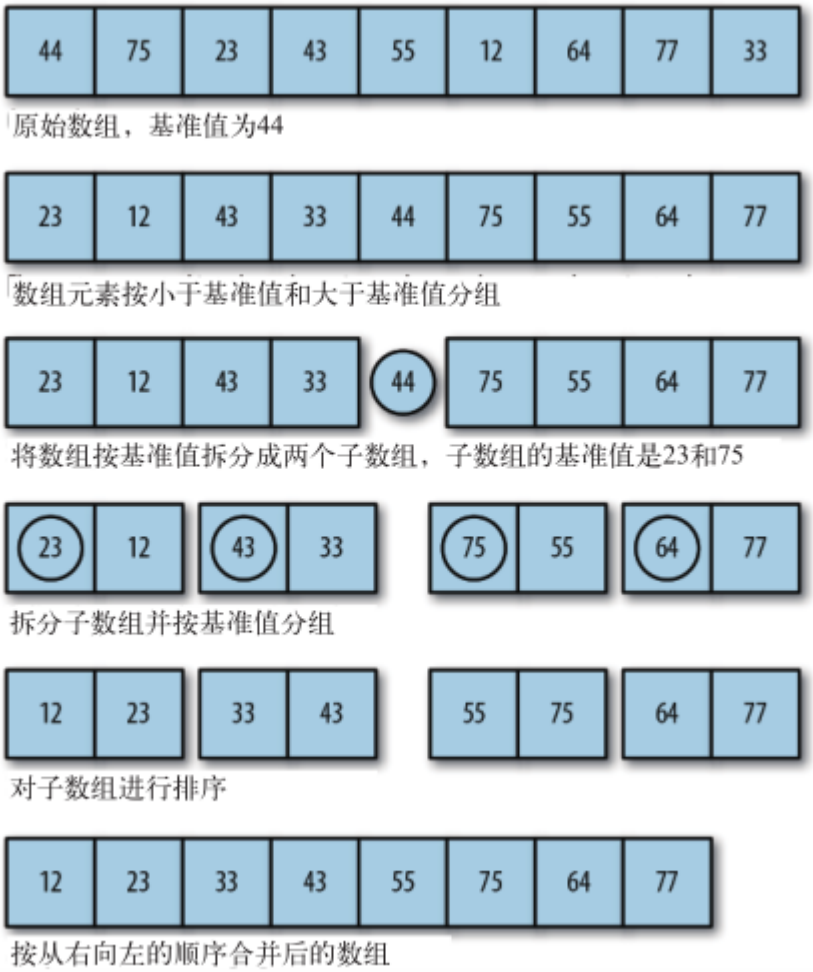
实现原理，把一系列已经排好序的子序列合并成一个大的完成的有序序列。

实现过程：先将无序序列不断拆分至不能再分，再将拆分后的子序列合并成完整有序序列。

缺点：需要很大的空间存储拆分的子数组，使用递归时，递归深度大。

快速排序（经典）

在列表中选择一个元素作为基准值（pivot）。数据的排序围绕着基准值进行，将列表中小于基准值的元素放在基准值的左边，将列表中大于基准值的元素放在基准值的右边（由小到大排序）。然后对子序列重复该过程。



排序算法性能比较

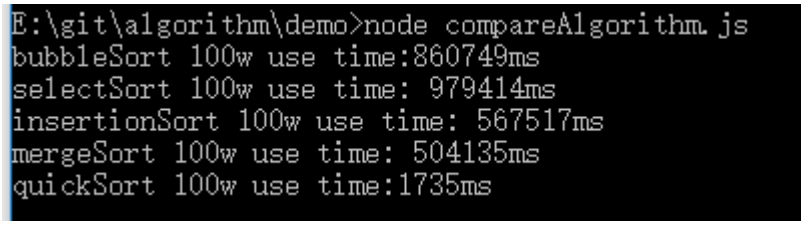
1. 1w随机数排序

```
E:\git\algorithm\demo>node compareAlgorithm.js
bubbleSort 1w use time:197ms
selectSort 1w use time: 99ms
insertionSort 1w use time: 57ms
mergeSort 1w use time: 21ms
quickSort 1w use time:11ms
```

2. 10w随机数排序

```
E:\git\algorithm\demo>node compareAlgorithm.js
bubbleSort 10w use time:11828ms
selectSort 10w use time: 9720ms
insertionSort 10w use time: 5657ms
mergeSort 10w use time: 219ms
quickSort 10w use time:109ms
```

3. 100w随机数排序



作业

1. 上述排序算法的时间复杂度，空间复杂度，是否稳定？
2. 快速排序算法的时间复杂度是怎么得到的？
3. 快速排序算法非递归实现。

了解更多 or 代码地址

https://github.com/ijinxin/algorithm