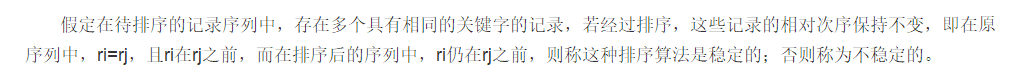
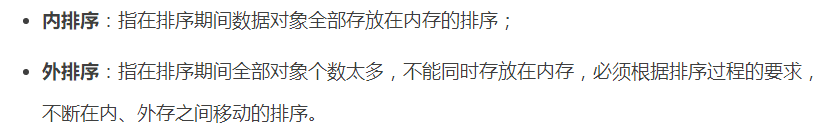
### 排序算法

经典算法汇总表↓：

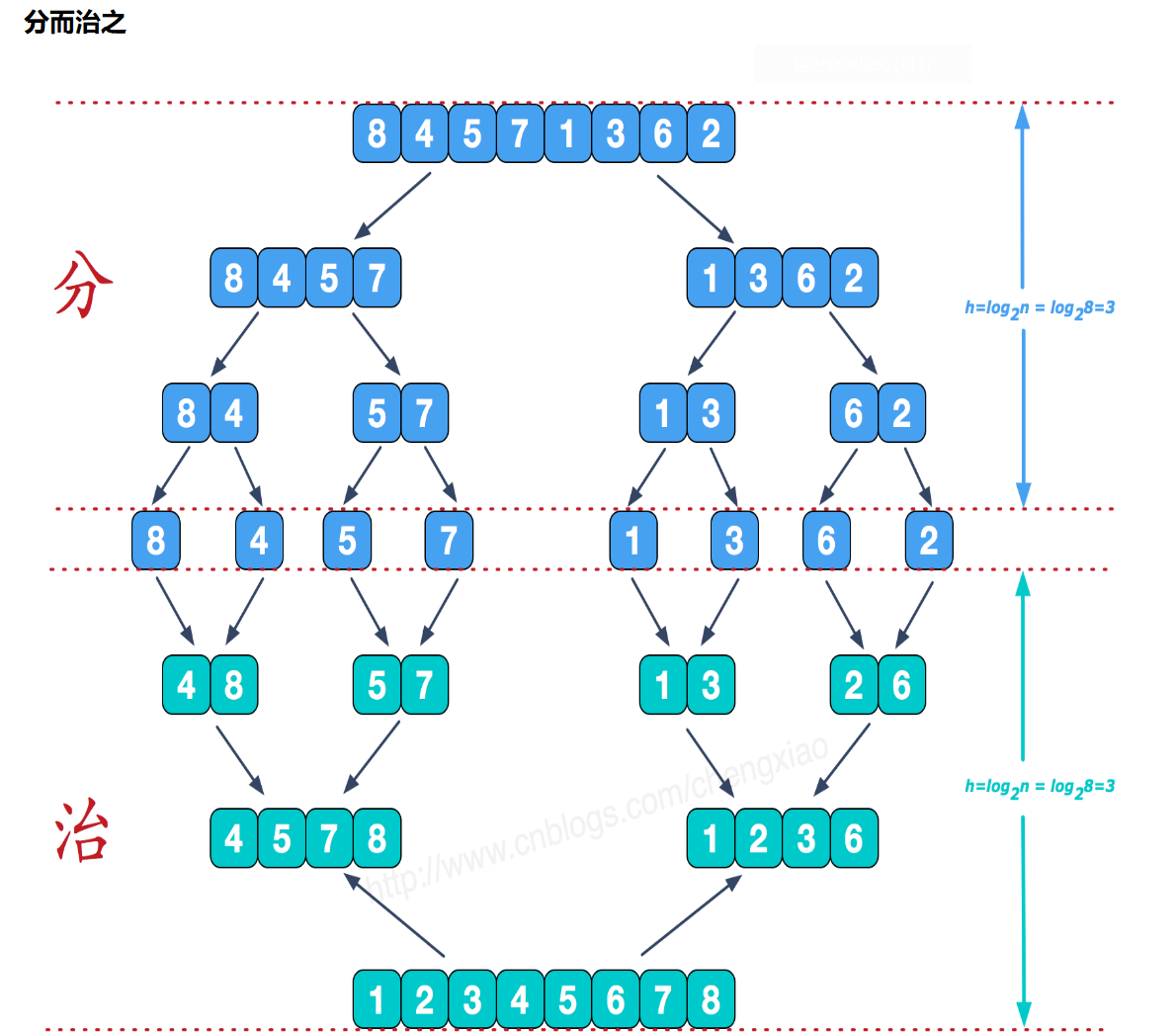
排序算法稳定性↓：

空间复杂度↓：

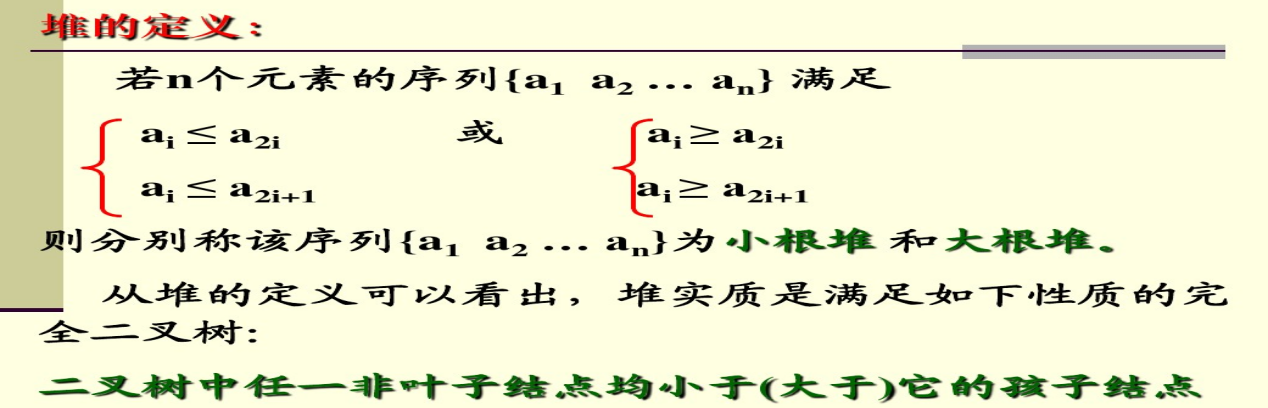
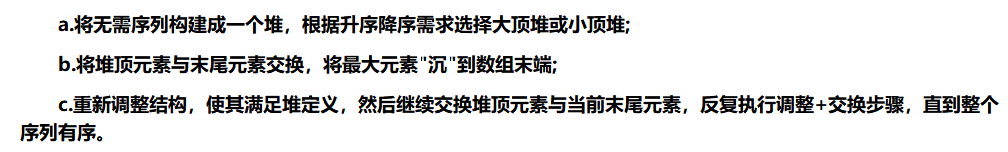
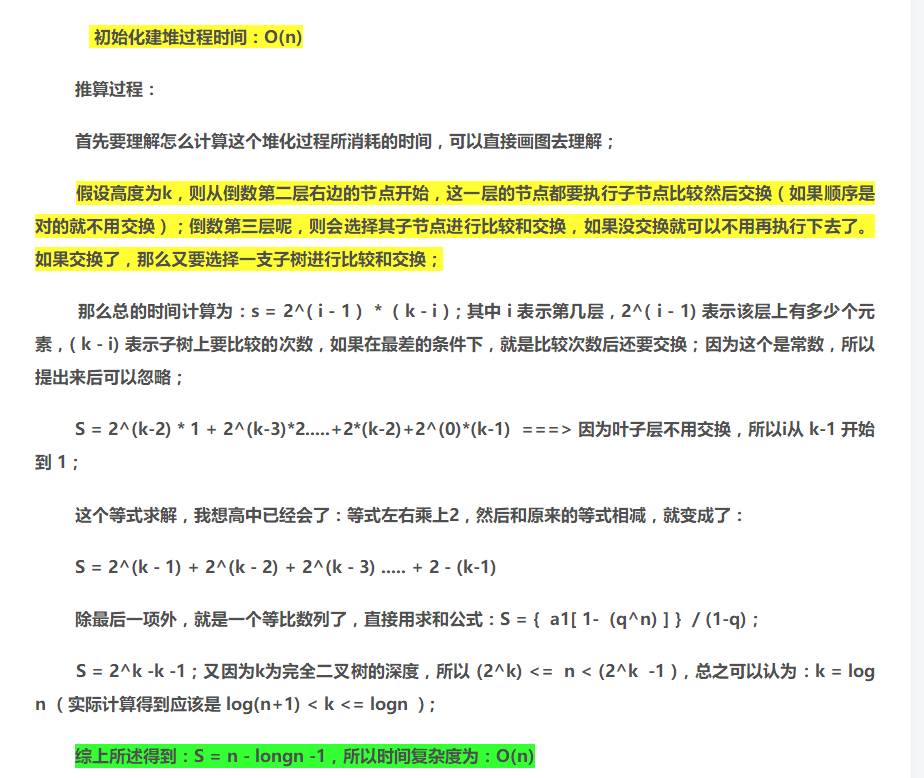
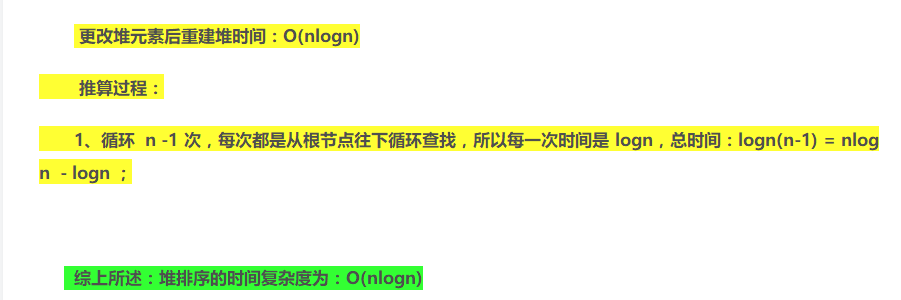
排序分类↓：



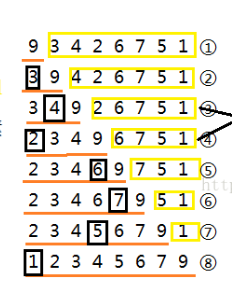
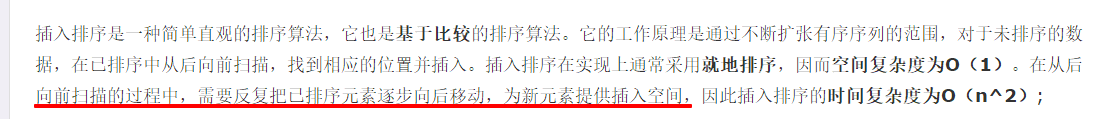
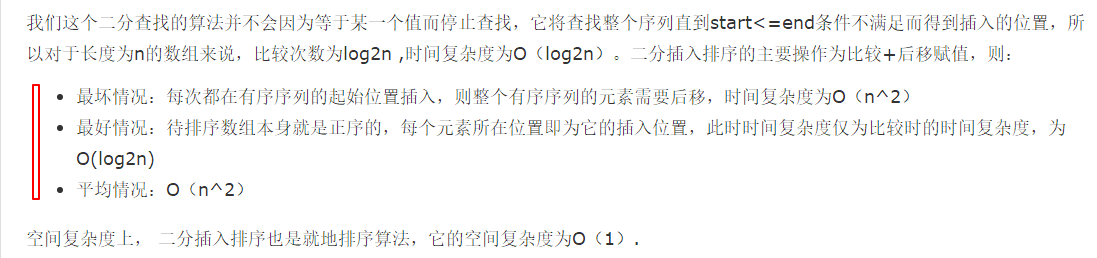
***合并算法***

* java中Arrays.mergeSort方法使用的就是合并排序，方法中的分块大小默认是7（Arrays.INSERTIONSORT\_THRESHOLD），当被分成长度小于7的小片段时，再使用类似插入排序，对小片段中的6个元素进行插入排序

***堆排序*** 与 ***最大堆***

* 大根堆，小根堆
* 堆排序也是选择排序的一种
* Java实现堆排序：<http://blog.csdn.net/yuzhihui_no1/article/details/44258297>
* 堆排序中对初始化的时间复杂度：
* 更改堆元素后重建堆的时间复杂度：

***插入排序***

* 插入排序的空间复杂度为O(1)
* 
* ***二分插入排序***：是对直接插入算法的优化

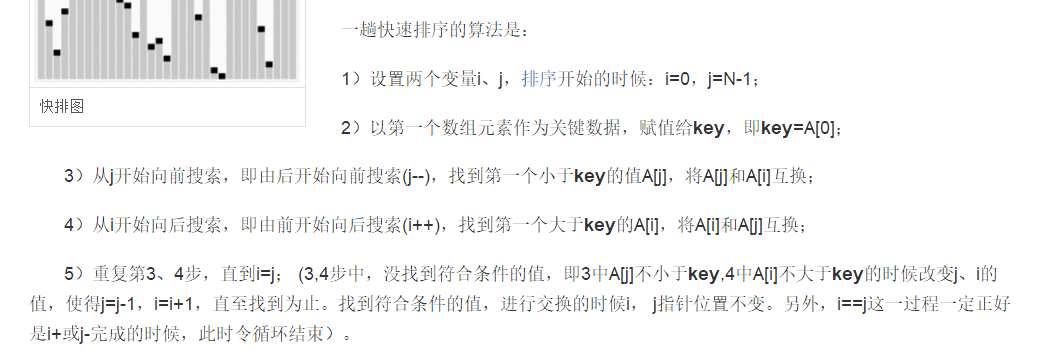
***希尔排序***

* 
* <https://www.cnblogs.com/chengxiao/p/6104371.html>

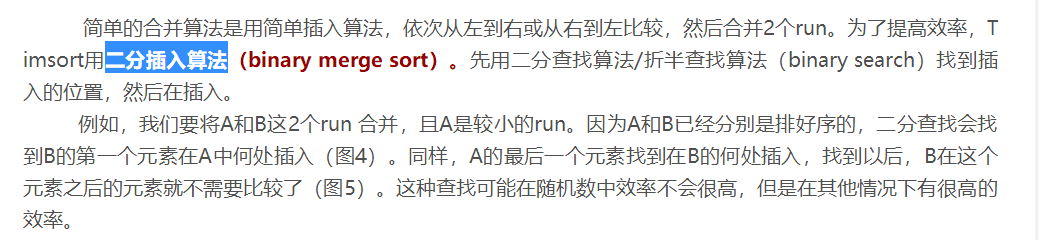
***计数排序***

* 计数排序

***快速排序***

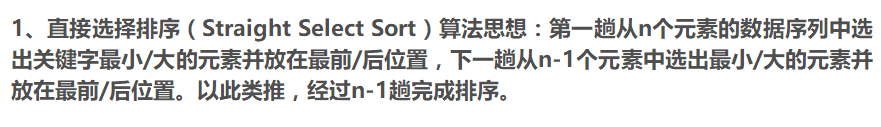
* *快速排序*：首先任意选取一个数据（通常选用数组的第一个数）作为关键数据，然后将所有比它小的数都放到它前面，所有比它大的数都放到它后面，这个过程称为一趟快速排序 ; 快速排序最小空间占用 ↓***值得注意***的是，快速排序不是一种稳定的[排序算法](https://baike.baidu.com/item/%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95" \t "https://baike.baidu.com/item/%E5%BF%AB%E9%80%9F%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95/_blank)，也就是说，多个相同的值的相对位置也许会在算法结束时产生变动。

***TimSort***

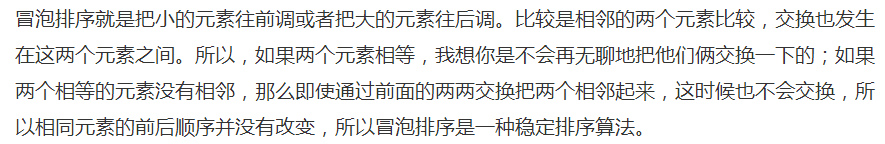
* TimSort排序：<http://blog.csdn.net/yangzhongblog/article/details/8184707>
* 
* Arrays类中的timsort流程：

1. 当长度小于32（MIN\_MERGE）时，先选出前n个有序的元素，并以增序形式输出，剩下的元素采用2分插入（binarySort方法）的方式，添加到有序数组中
2. 当长度大于32时，分段（分成许多个runs） → 合并runs（mergeCollapse） → 省略掉run1的前和run2的后面部分元素（忽略首位片段，通过gallopLeft和gallopRight计算出来的） → 对剩下的元素进行排序（mergeHi或mergeLo） → mergeXX中关键代码在循环代码（outer:while (true)），这个循环中会对不同的情况再采取直接排序或者继续采用忽略首位片段的方式尽心快速排序

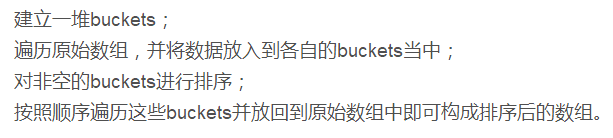
***选择排序***

* 选择排序和冒泡排序相似：

***冒泡排序***

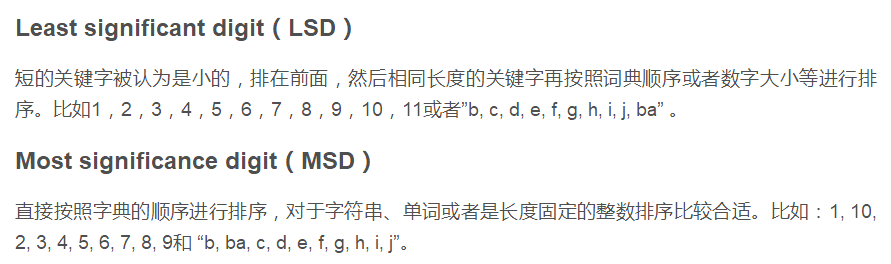
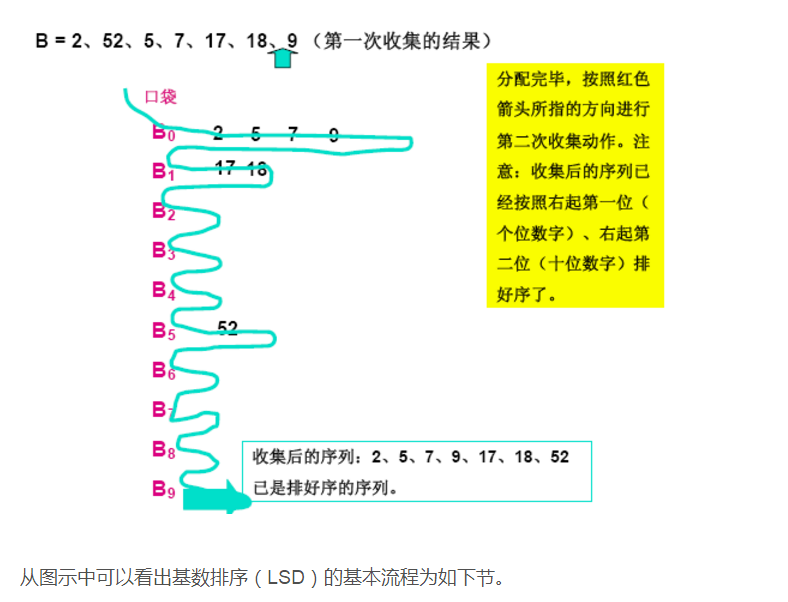
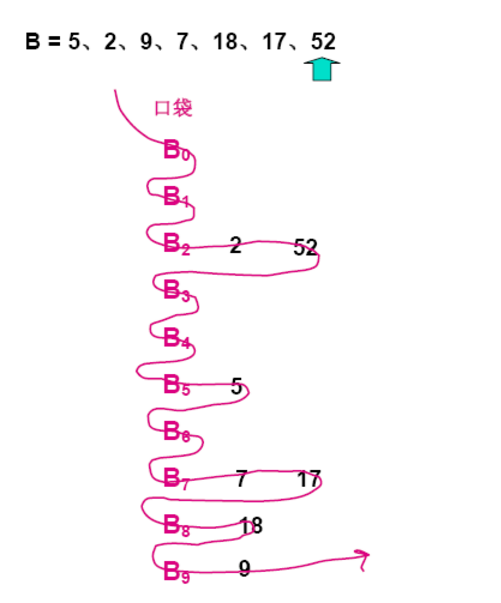
* 冒泡排序为什么是稳定的：

***桶排序***

* 流程：

***基数排序***

什么是基数

* 2种基数培训：
* 简单流程图：

***Other***

* 一个神奇的算法：如果n是2的幂，则返回MIN\_MERGE的1/2否则，返回[MIN\_MERGE/2, MIN\_MERGE]