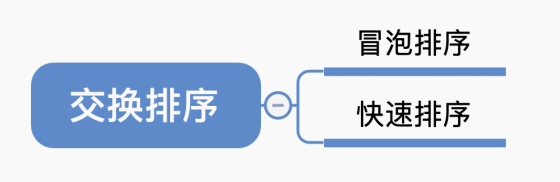
交换排序

基于“交换”的排序：

根据序列中两个元素关键字的⽐较结果来**对换**这两个记录在序列中的**位置**



1. 冒泡排序

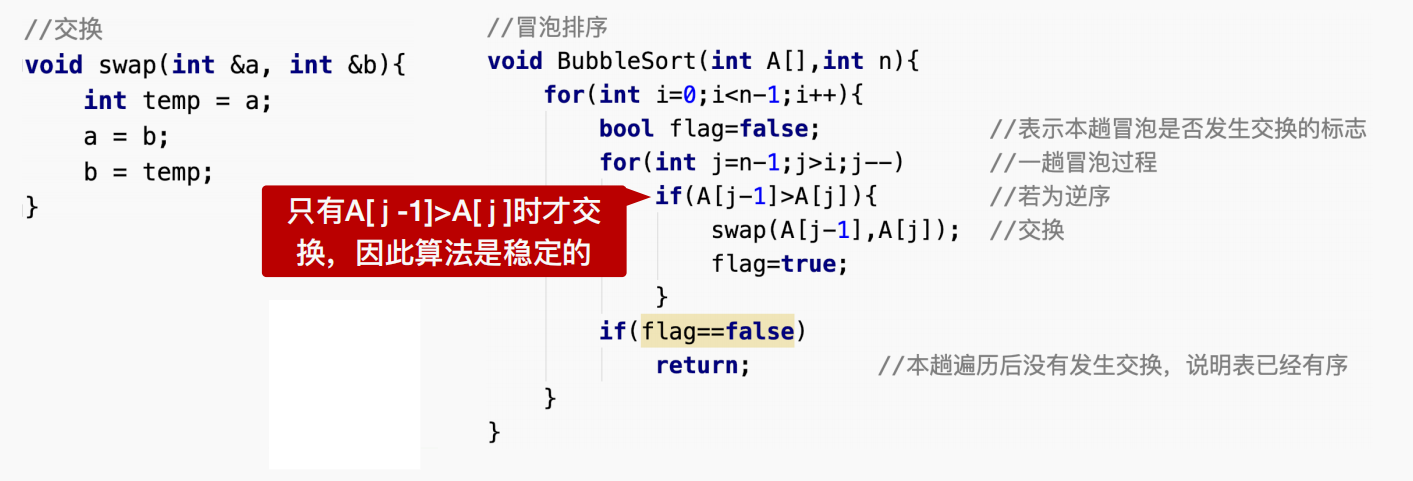
思路：

**从后往前**（或从前往后）**两两⽐较相邻元素**的值，若为**逆序**（即A[i-1]>A[i]），则**交换**它们，直到序列⽐较完。称这样过程为“⼀趟”冒泡排序。

**每一趟排序**都可以**确定一个最小值**在最前面

若某⼀趟排序**没有发⽣“交换”**，说明此时已经**整体有序**

算法实现：

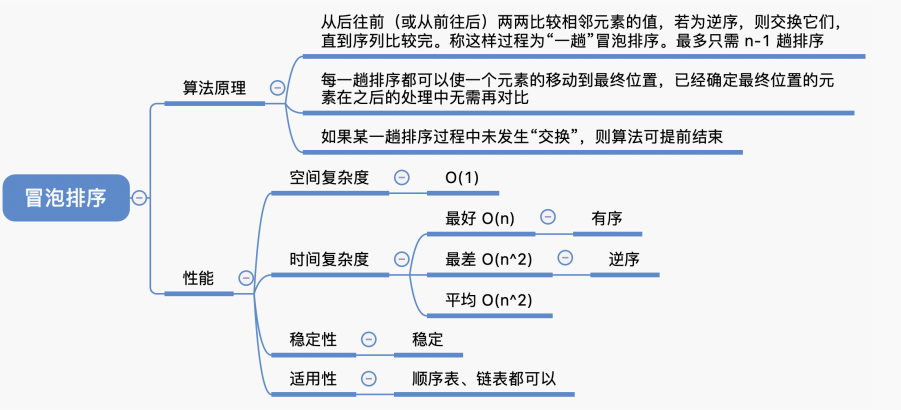


算法性能分析：

空间复杂度：O(1)



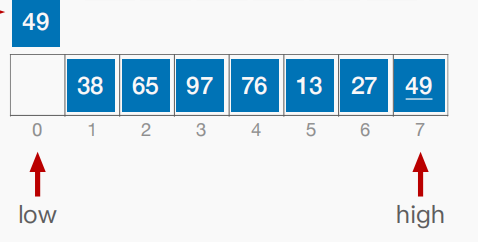
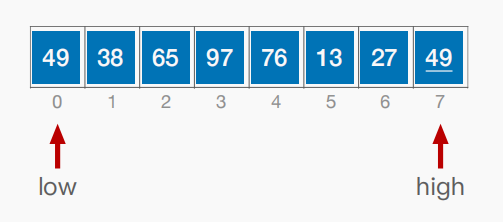
稳定性：稳定

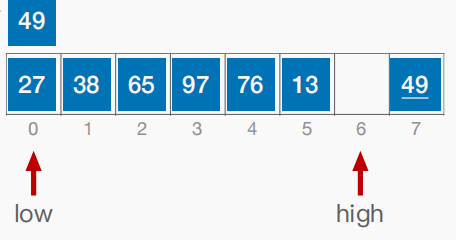
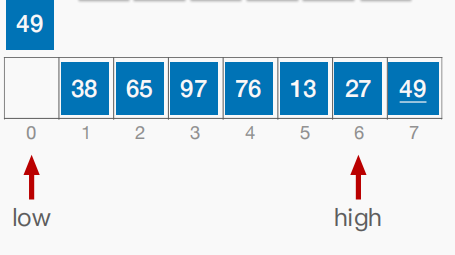


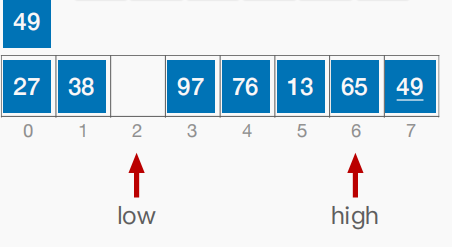
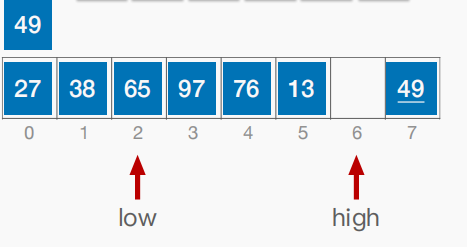
1. 快速排序

算法思想：

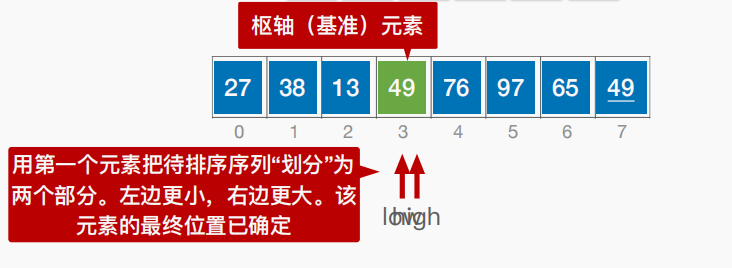
在待排序表L[1…n]中任取⼀个元素**pivot**作为**枢轴**（或基准，通常取⾸元素），通过⼀趟排序将待排序表划分为独⽴的两部分L[1…k-1]和L[k+1…n]，使得**L[1…k-1]中的所有元素⼩于pivot**，**L[k+1…n]中的所有元素⼤于等于pivot**，则**pivot放在了其最终位置L(k)上**，这个过程称为⼀次“**划分**”。然后分别**递归地对两个⼦表**重复上述过程，**直⾄每部分内只有⼀个元素或空为⽌**，即所有元素放在了其最终位置上。







更⼩的元素都交换到左边，更⼤的元素都交换到右边



算法实现：



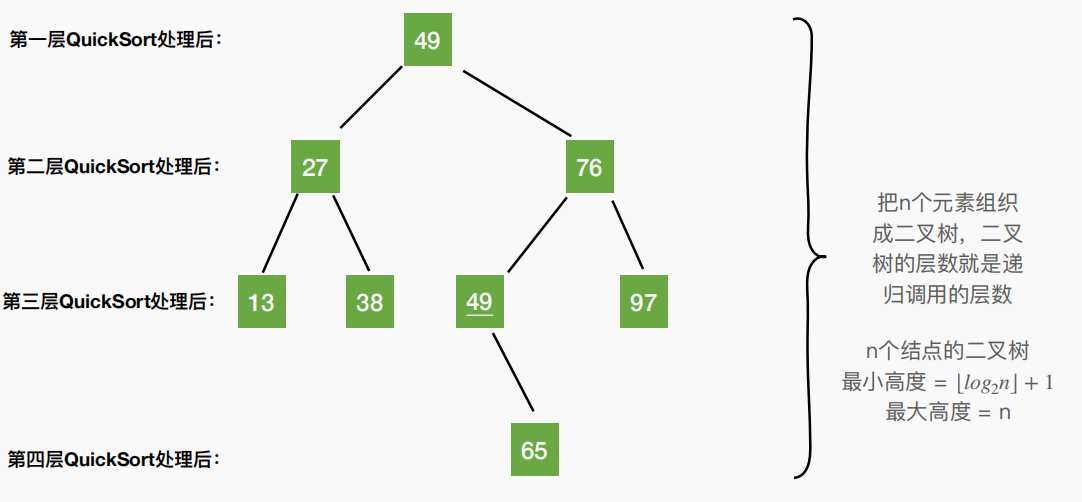
算法效率分析：

空间复杂度=**O(递归层数**)



时间复杂度=**O(n\*递归层数)**

平均时间复杂度=**O(nlog2n)**



最好情况

若每⼀次选中的“枢轴”将待排序序列**划分为均匀的两个部分**，则**递归深度最⼩**，**算法效率最⾼**

最坏情况

若每⼀次选中的“枢轴”将待排序序列**划分为很不均匀的两个部分**，则会导致**递归深度增加**，**算法效率变低**

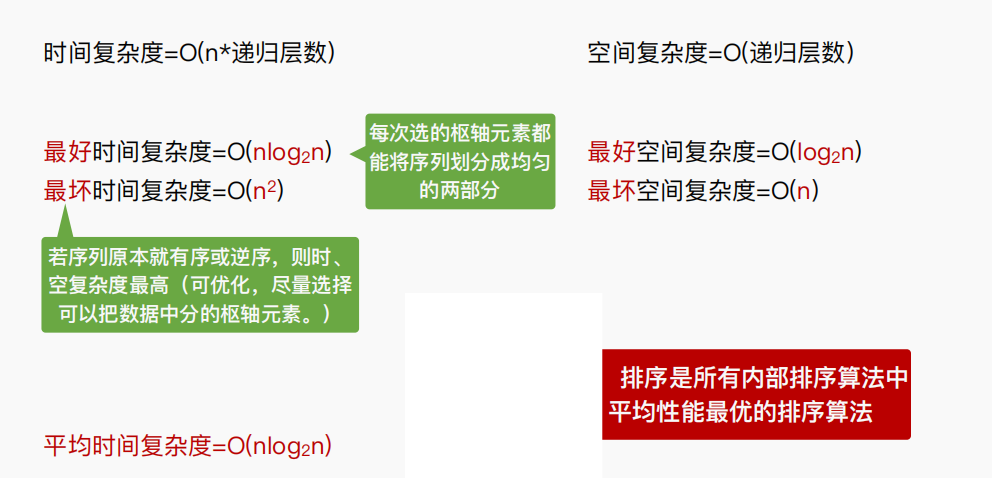
若初始序列**有序或逆序**，则快速排序的**性能最差**（因为每次选择的都是最靠边的元素）

快速排序算法优化思路：

尽量选择可以把数据中分的枢轴元素。

①选头、中、尾三个位置的元素，取中间值作为枢轴元素；

②随机选⼀个元素作为枢轴元素



稳定新：**不稳定**

