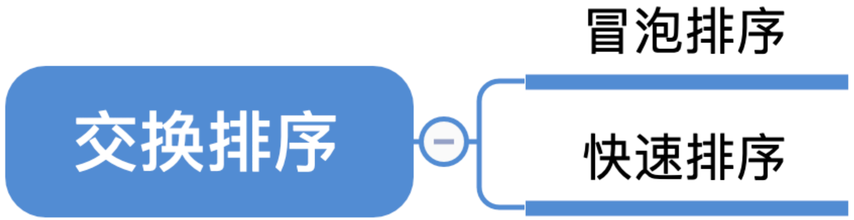
本节内容

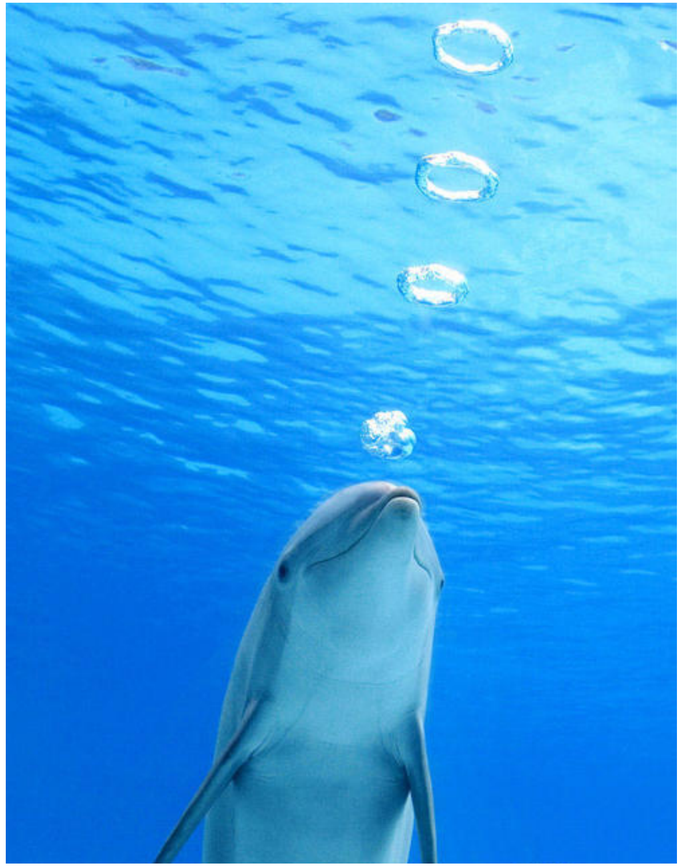
# 冒泡排序

# 知识总览



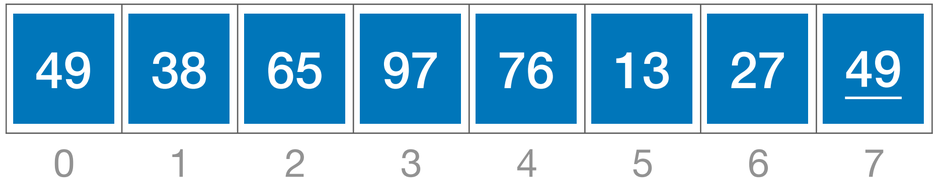

基于“交换”的排序：根据序列中两个元素关键字的比较结果来对换这两个记录在序列中的位置

# 冒泡?



# 冒泡排序

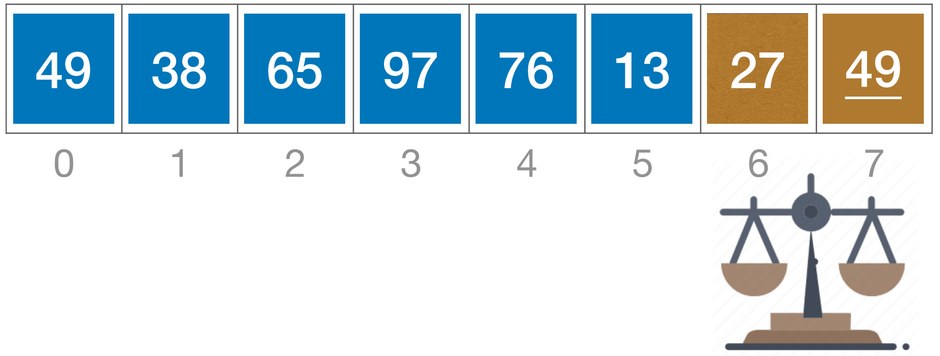
从后往前（或从前往后）两两比较相邻元素的值，若为逆序（即$A[i-1]>A[i]$），则交换它们，直到序列比较完。称这样过程为“一趟”冒泡排序。



# 冒泡排序

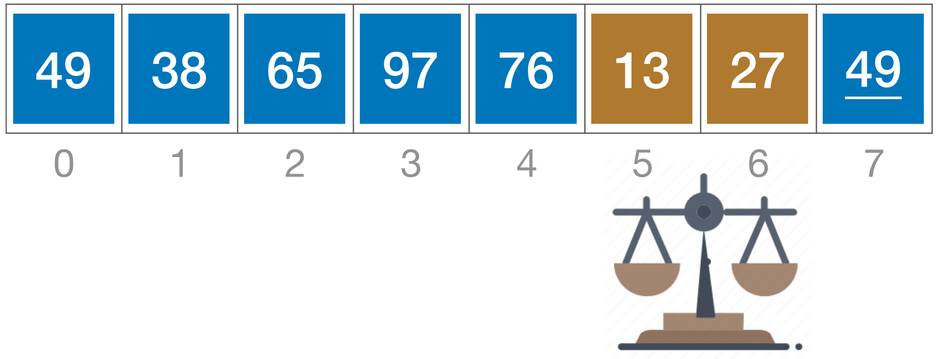
从后往前（或从前往后）两两比较相邻元素的值，若为逆序（即$A[i-1]>A[i]$），则交换它们，直到序列比较完。称这样过程为“一趟”冒泡排序。

**第1趟：**



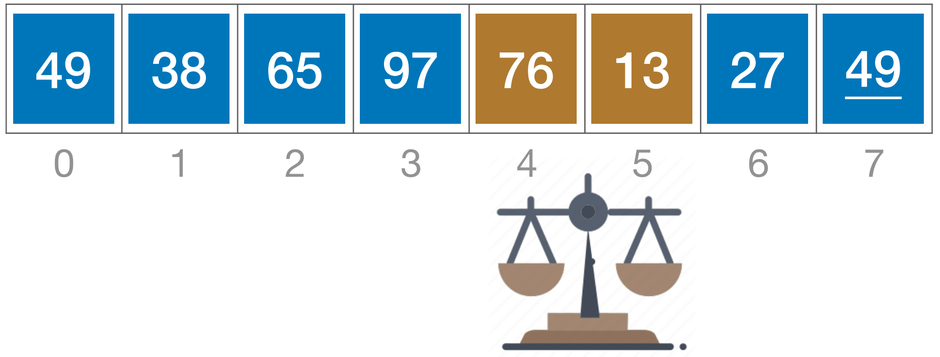
# 冒泡排序

第1趟：



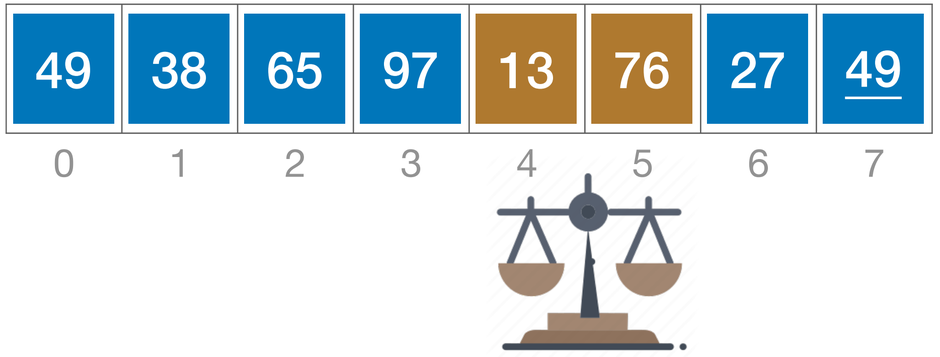
# 冒泡排序

第1趟：



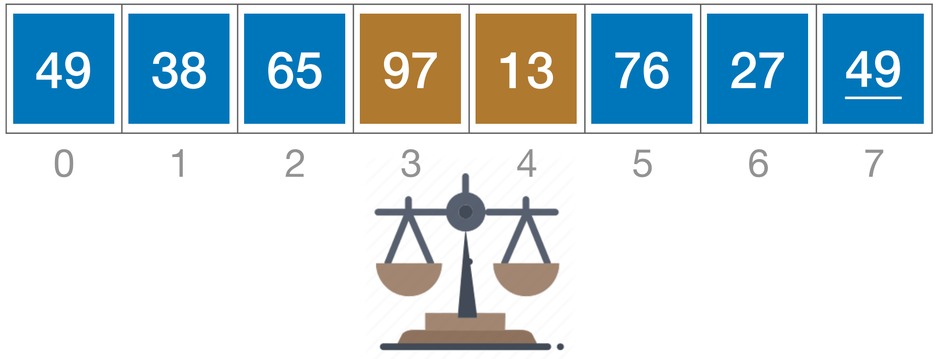
# 冒泡排序

第1趟：



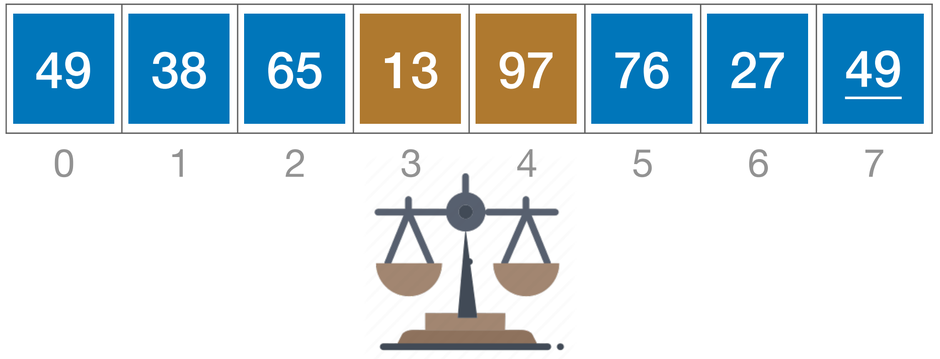
# 冒泡排序

第1趟：



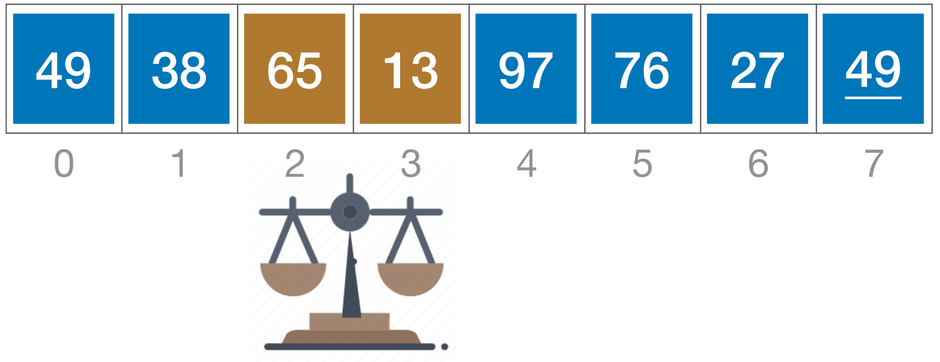
# 冒泡排序

第1趟：



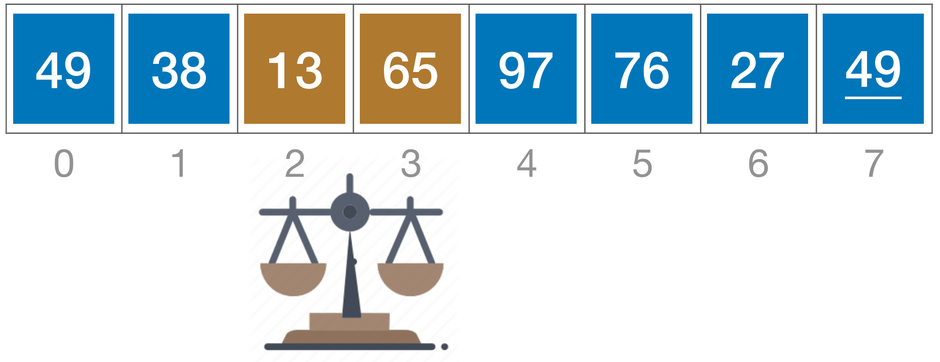
# 冒泡排序

第1趟：



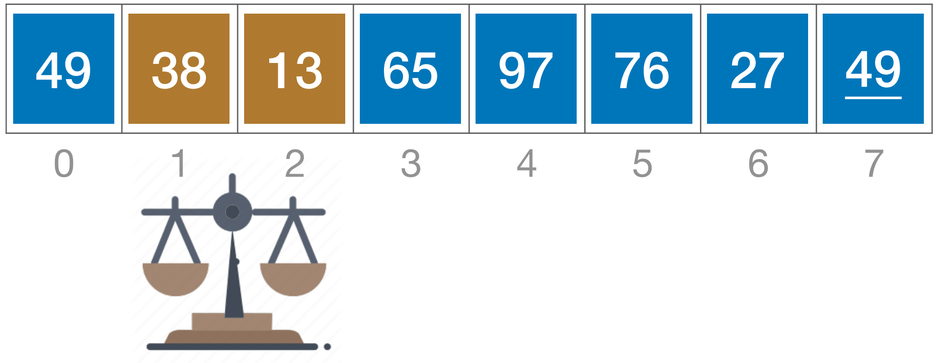
# 冒泡排序

第1趟：



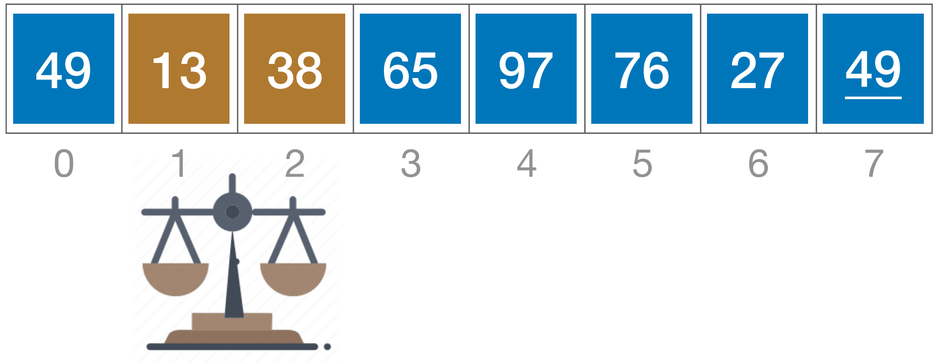
# 冒泡排序

第1趟：



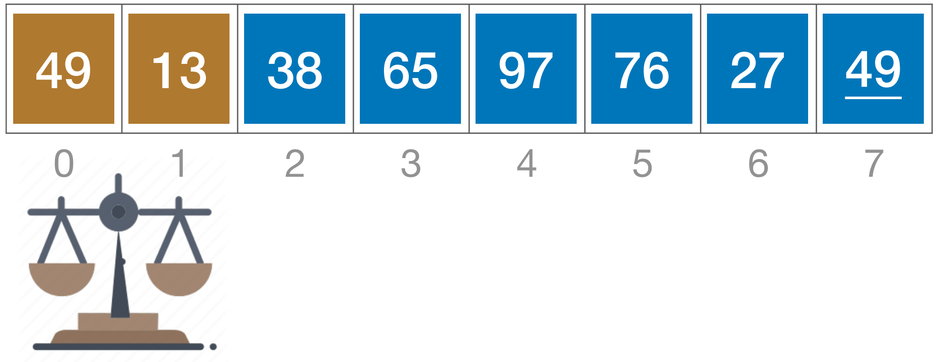
# 冒泡排序

第1趟：



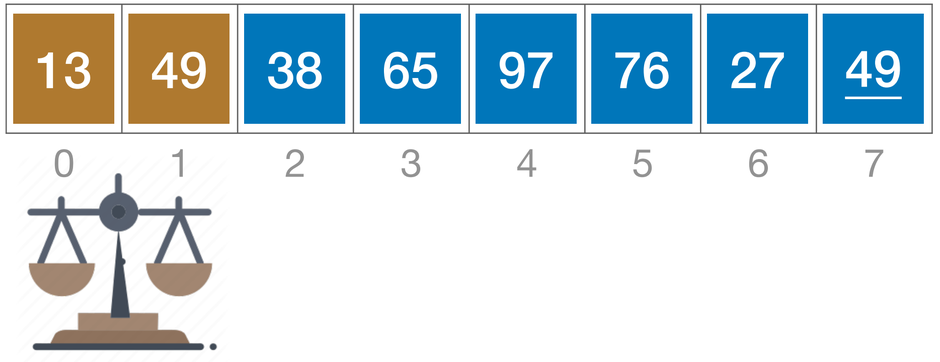
# 冒泡排序

第1趟：



# 冒泡排序

第1趟：



# 冒泡排序

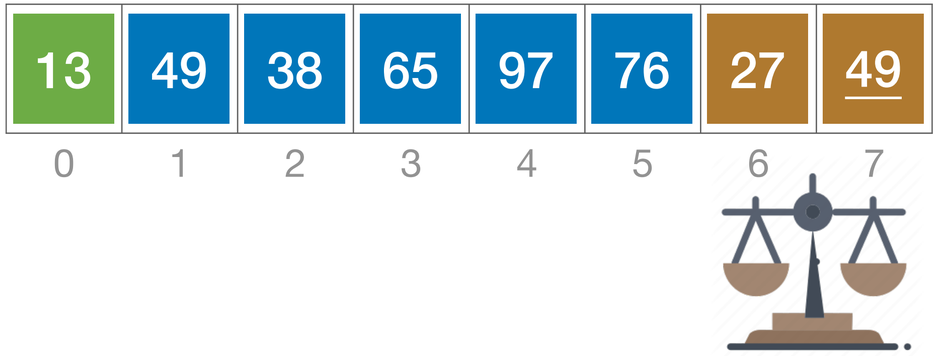
第1趟结束：


第一趟排序使关键字值最小的一个元素“冒”到最前面

# 冒泡排序

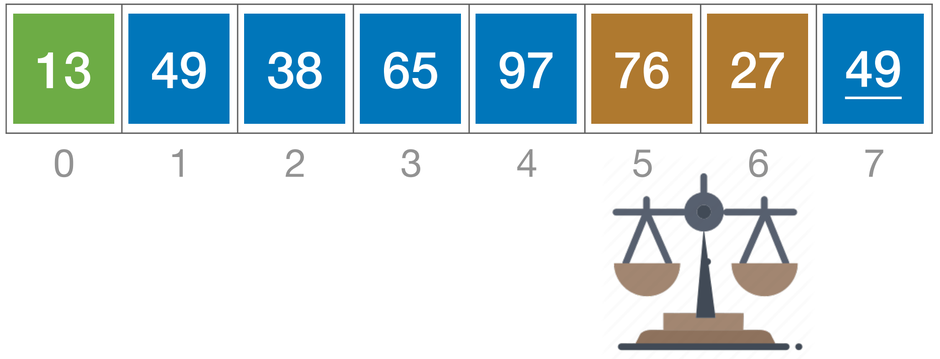
从后往前（或从前往后）两两比较相邻元素的值，若为逆序（即$A[i-1]>A[i]$），则交换它们，直到序列比较完。称这样过程为“一趟”冒泡排序。

**第2趟：**



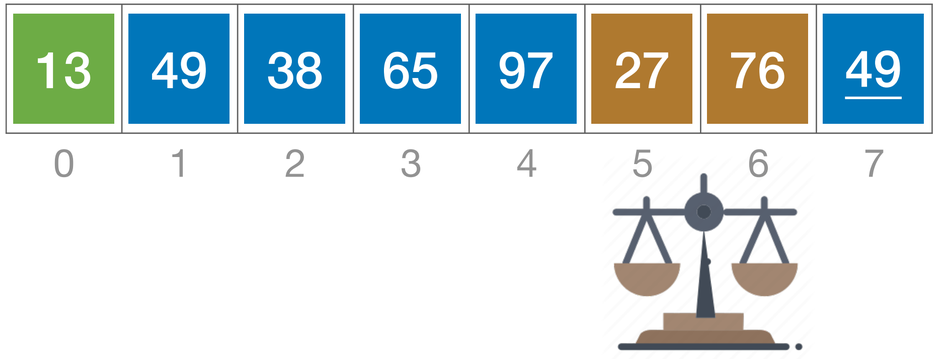
# 冒泡排序

第2趟：



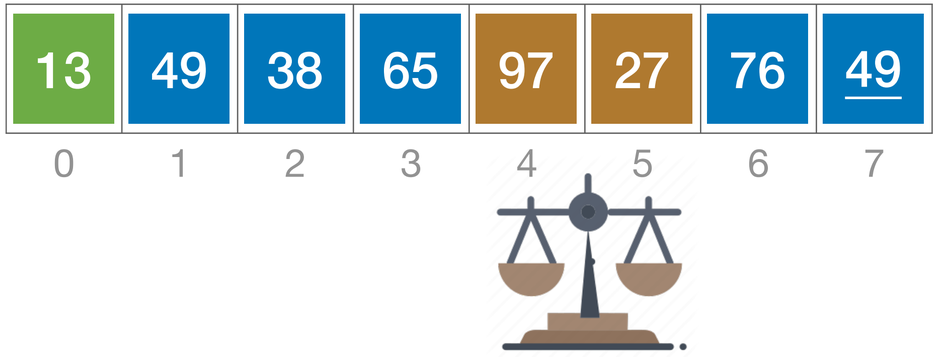
# 冒泡排序

第2趟：



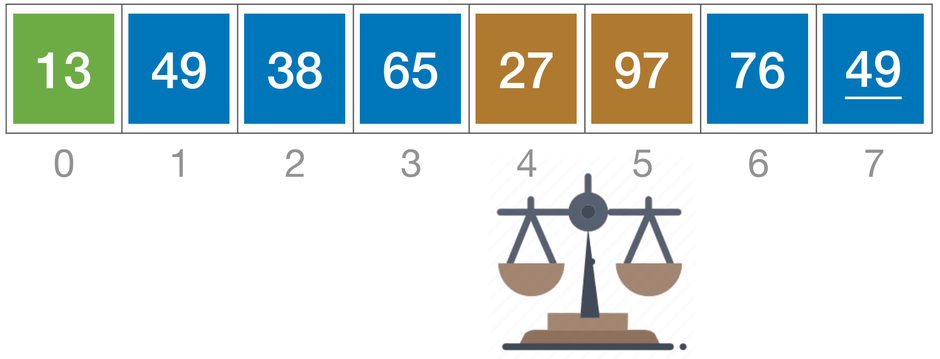
# 冒泡排序

第2趟：



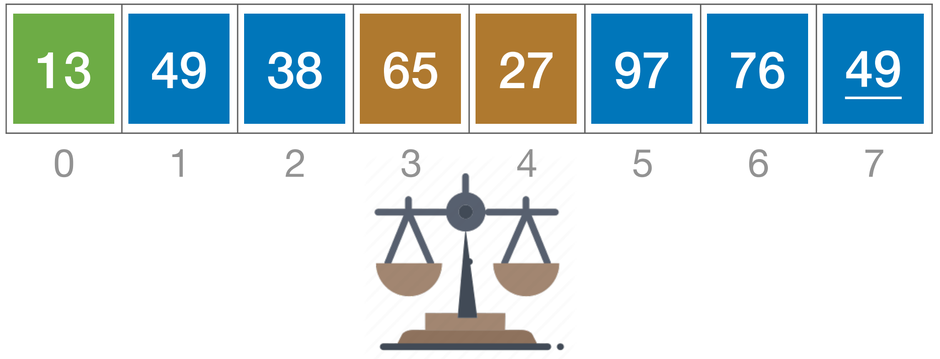
# 冒泡排序

第2趟：



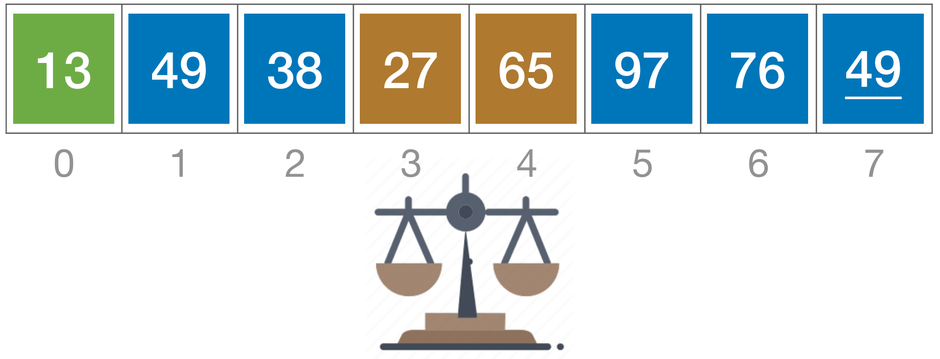
# 冒泡排序

第2趟：



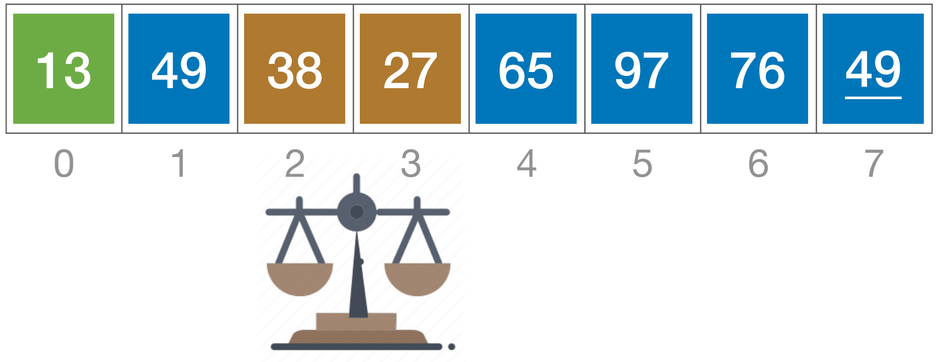
# 冒泡排序

第2趟：



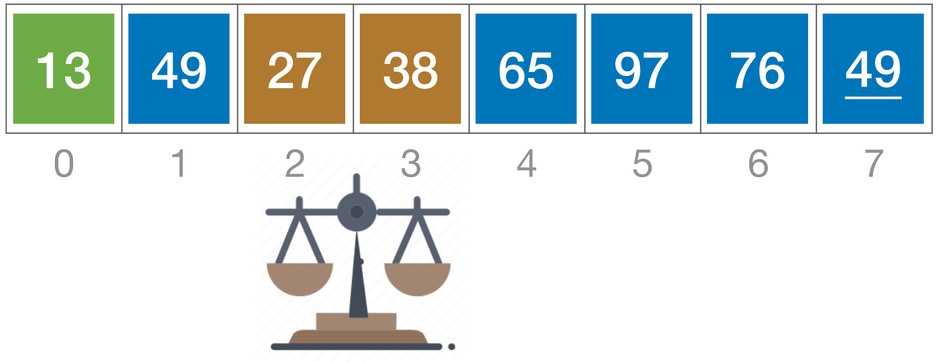
# 冒泡排序

第2趟：



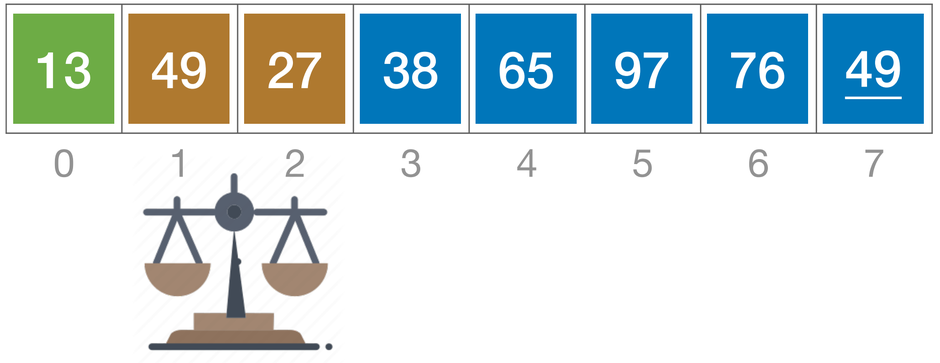
# 冒泡排序

第2趟：



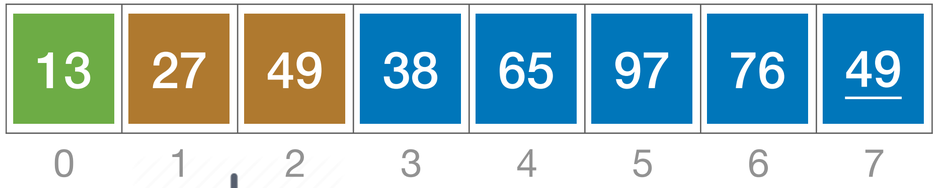
# 冒泡排序

第2趟：



# 冒泡排序

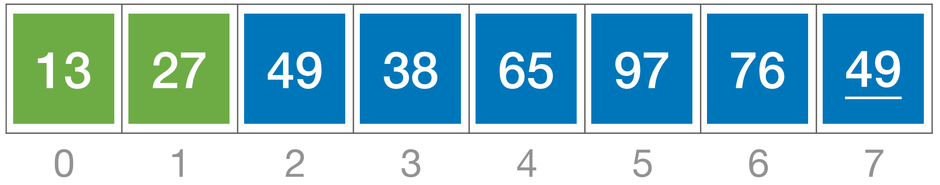
第2趟：




前边已经确定最终位置的元素不用再对比

# 冒泡排序

第2趟结束：

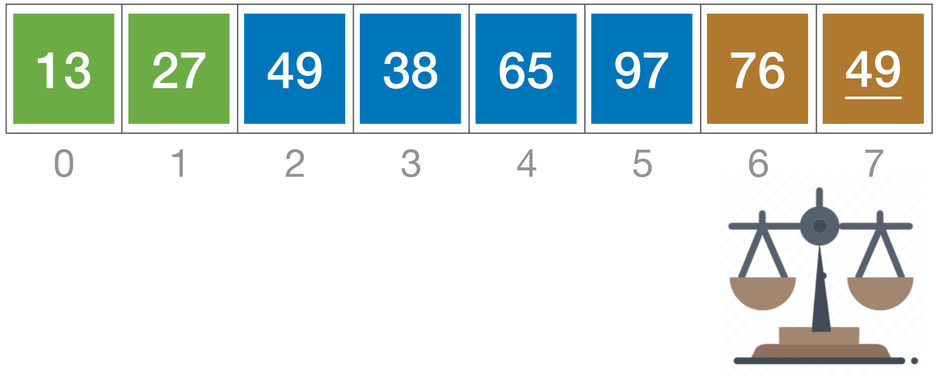



第2趟结束后，最小的两个元素会“冒”到最前边

# 冒泡排序

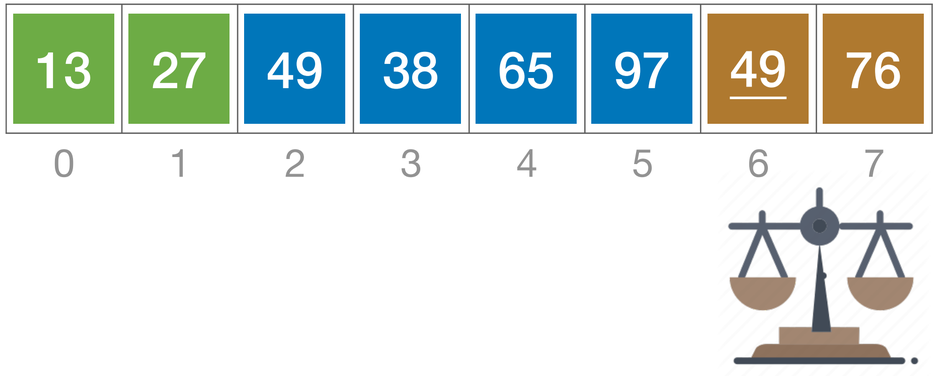
从后往前（或从前往后）两两比较相邻元素的值，若为逆序（即$A[i-1]>A[i]$），则交换它们，直到序列比较完。称这样过程为“一趟”冒泡排序。

**第3趟：**



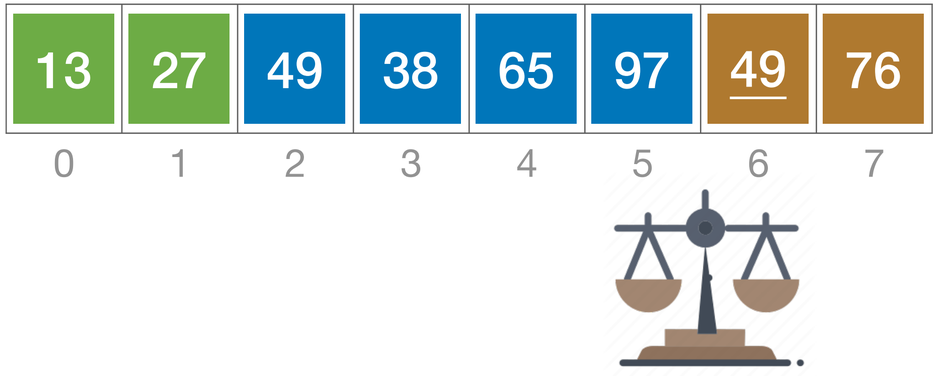
# 冒泡排序

第3趟：



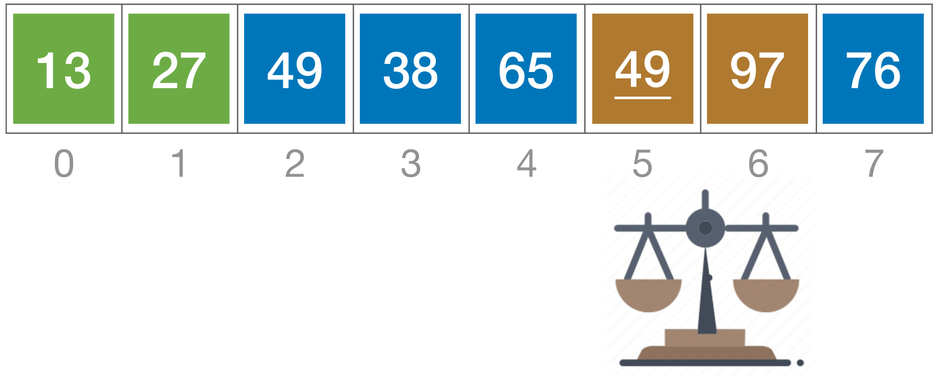
# 冒泡排序

第3趟：



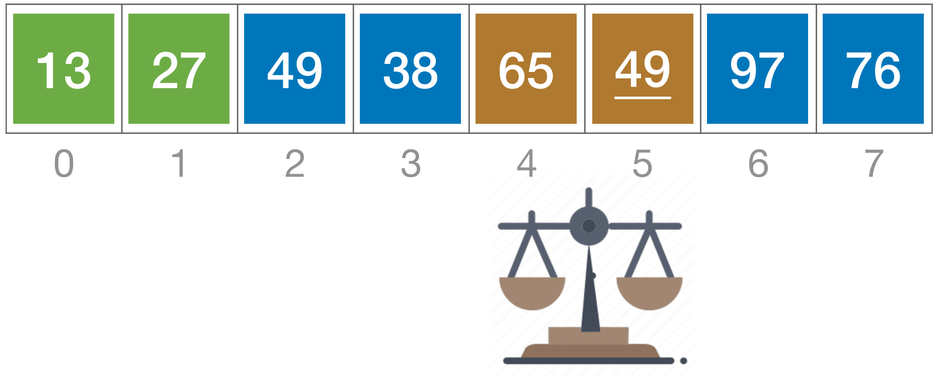
# 冒泡排序

第3趟：



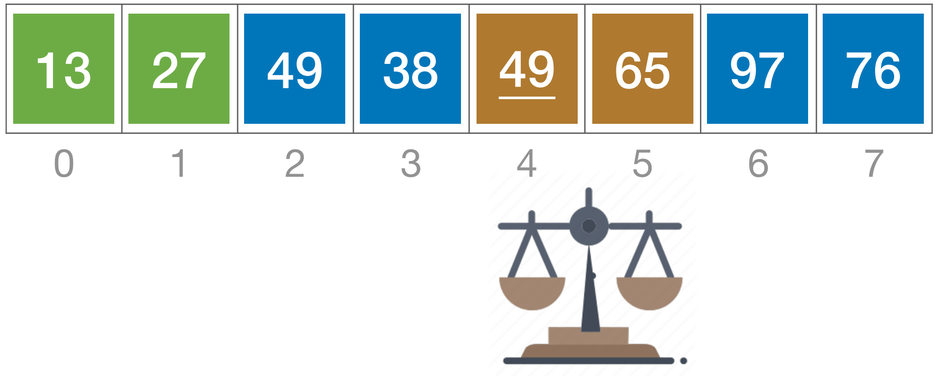
# 冒泡排序

第3趟：



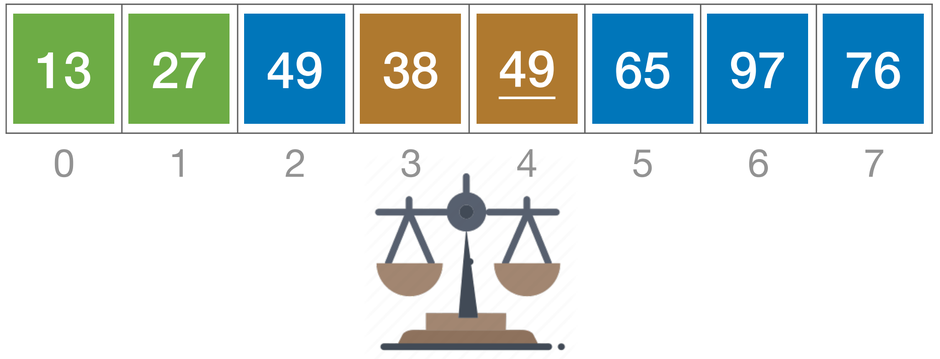
# 冒泡排序

第3趟：



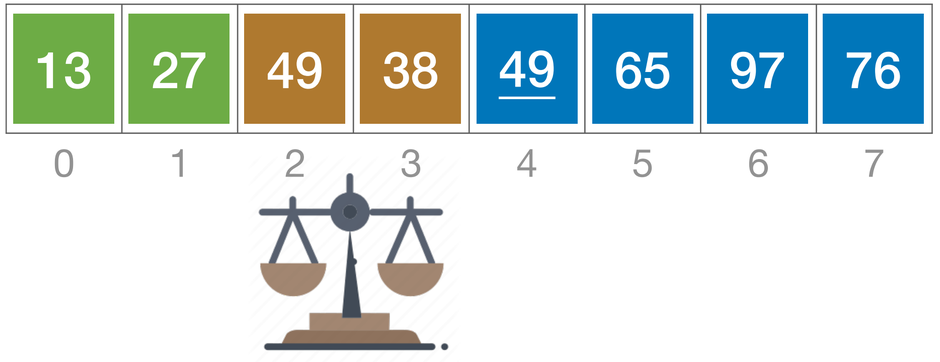
# 冒泡排序

第3趟：



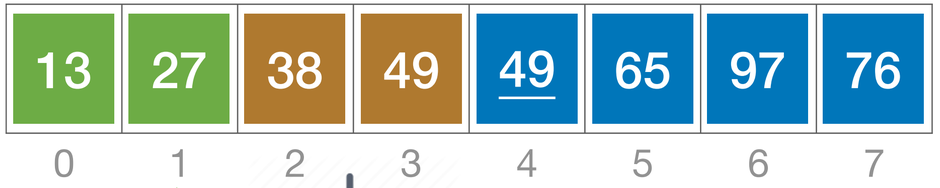
# 冒泡排序

第3趟：



# 冒泡排序

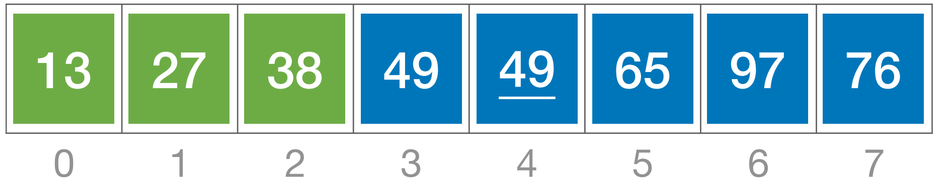
第3趟：




前边已经确定最终位置的元素不用再对比

# 冒泡排序

第3趟结束：

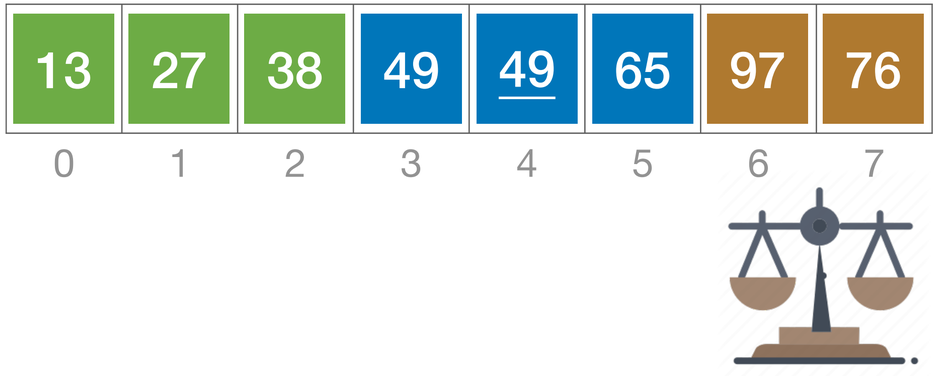



第3趟结束后，最小的3个元素会“冒”到最前边

# 冒泡排序

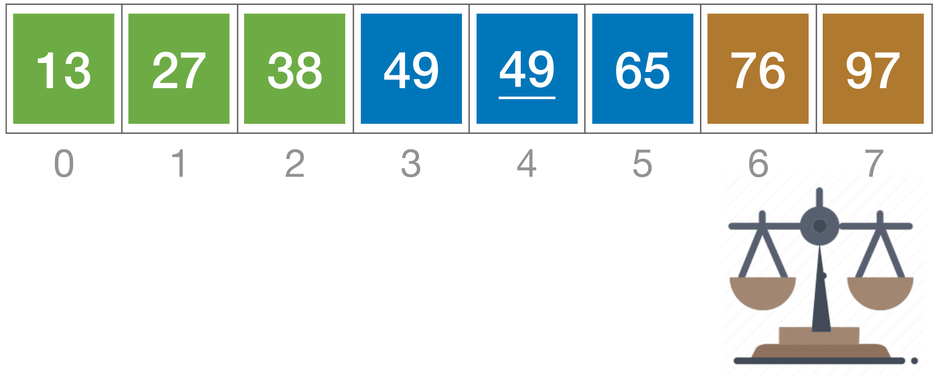
从后往前（或从前往后）两两比较相邻元素的值，若为逆序（即$A[i-1]>A[i]$），则交换它们，直到序列比较完。称这样过程为“一趟”冒泡排序。

**第4趟：**



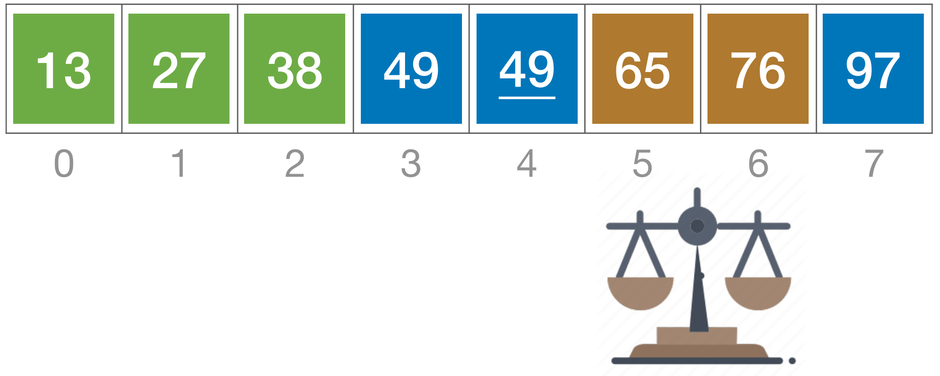
# 冒泡排序

第4趟：



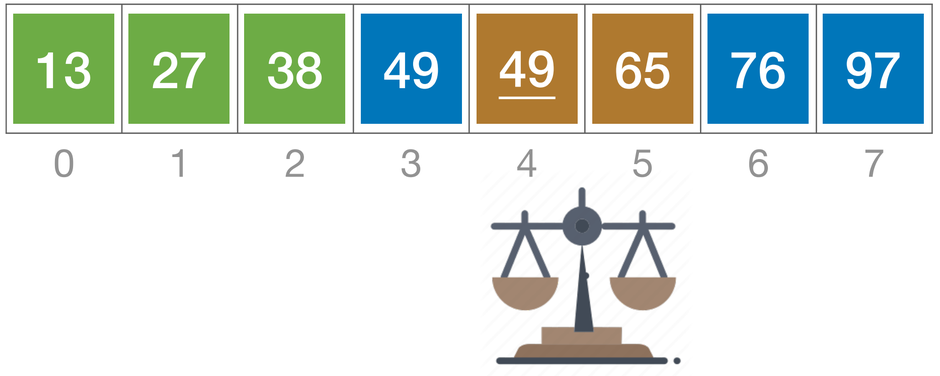
# 冒泡排序

第4趟：



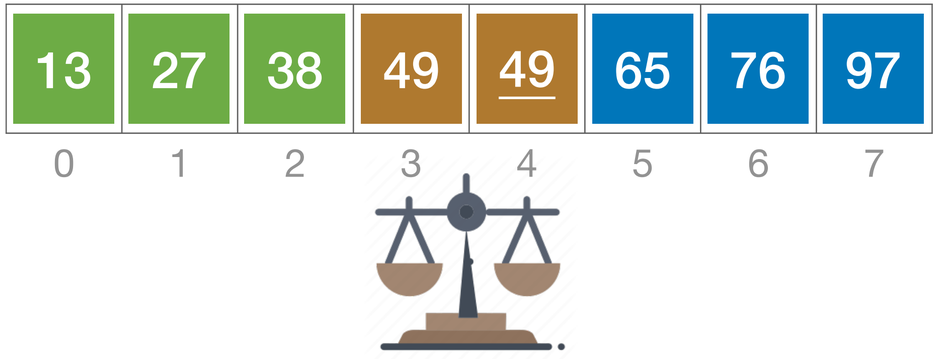
# 冒泡排序

第4趟：



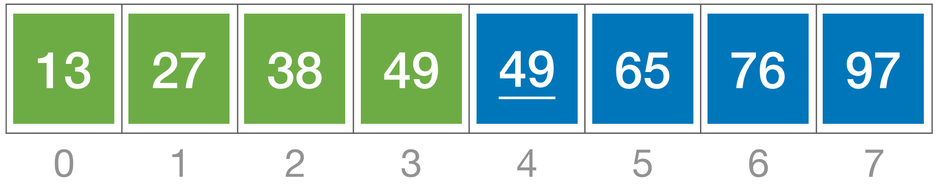
# 冒泡排序

第4趟：



# 冒泡排序

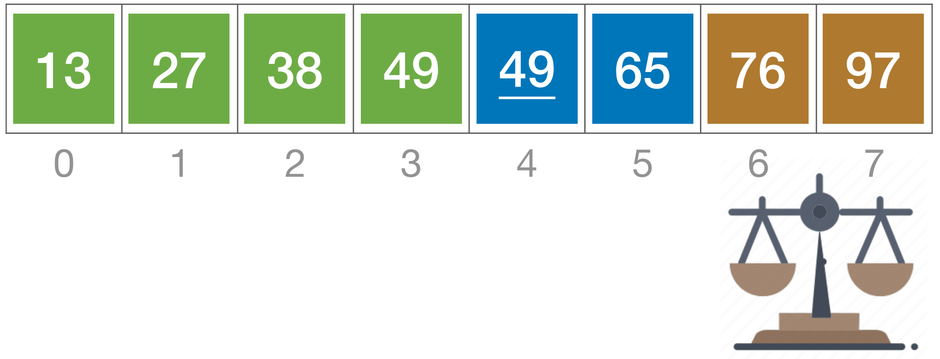
第4趟结束：



# 冒泡排序

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。

**第5趟：**



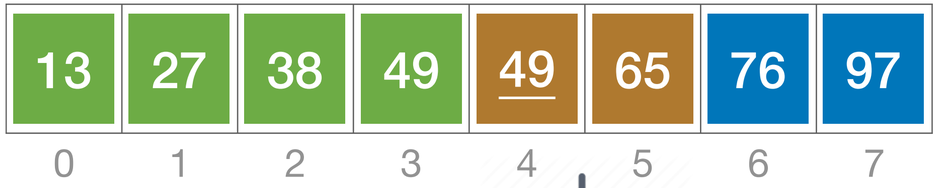
# 冒泡排序

第5趟：

# 冒泡排序

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。总共需进行 n–1 趟冒泡。

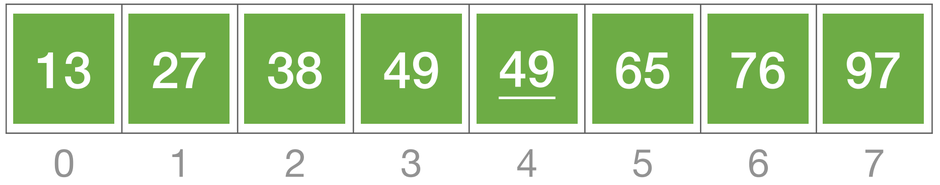
**第5趟：**




若某一趟排序没有发生“交换”，说明此时已经整体有序。

# 冒泡排序

第5趟结束：

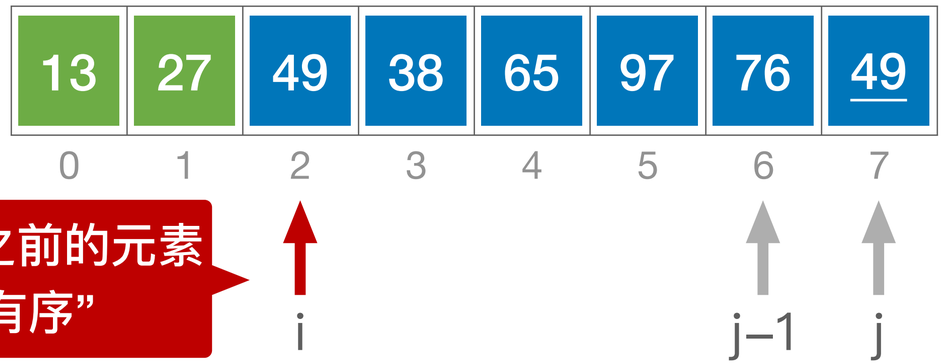


# 算法实现

```
//交换
void swap(int &a, int &b){
    int temp = a;
    a = b;
    b = temp;
}
```

```
//冒泡排序
void BubbleSort(int A[],int n){
    for(int i=0;i<n-1;i++){
        bool flag=false;                //表示本趟冒泡是否发生交换的标志
        for(int j=n-1;j>i;j--)          //一趟冒泡过程
            if(A[j-1]>A[j]){            //若为逆序
                swap(A[j-1],A[j]);      //交换
                flag=true;
            }
        if(flag==false)
            return;                     //本趟遍历后没有发生交换，说明表已经有序
    }
}
```

只有A[j-1]>A[j]时才交换，因此算法是稳定的


i 所指位置之前的元素都已“有序”

# 算法性能分析

空间复杂度：O(1)

最好情况（有序）：


比较次数=n−1；交换次数=0

最好时间复杂度=O(n)

最坏情况（逆序）：


比较次数=(n−1)+(n−2)+…+1 = $\frac{n(n-1)}{2}$ =交换次数

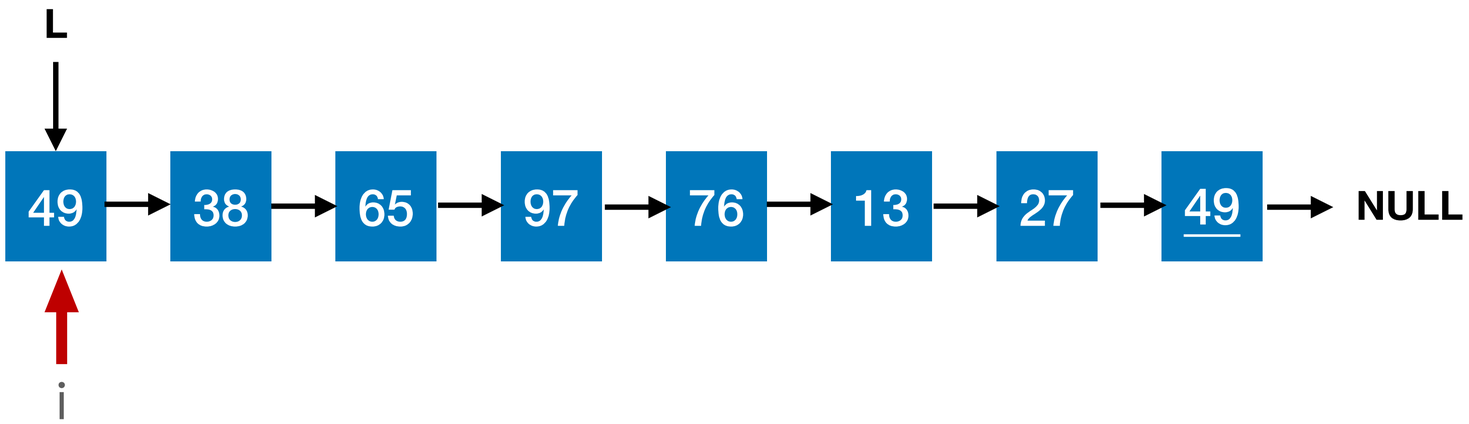
最坏时间复杂度=O($n^2$)

平均时间复杂度=O($n^2$)

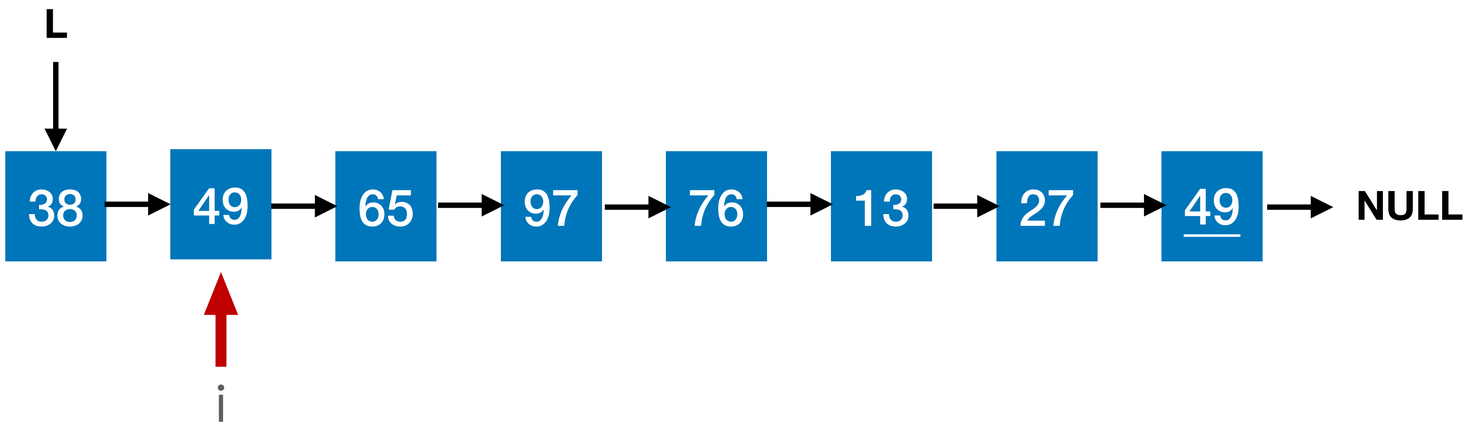
每次交换都需要移动元素3次

```
//交换
void swap(int &a, int &b){
    int temp = a;
    a = b;
    b = temp;
}
```

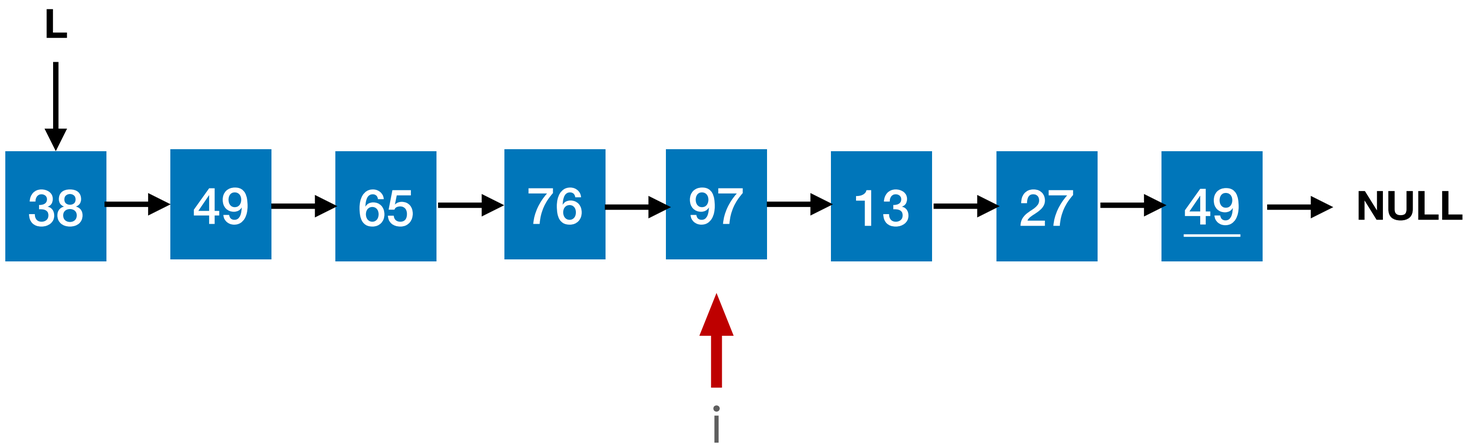
# 冒泡排序是否适用于链表?



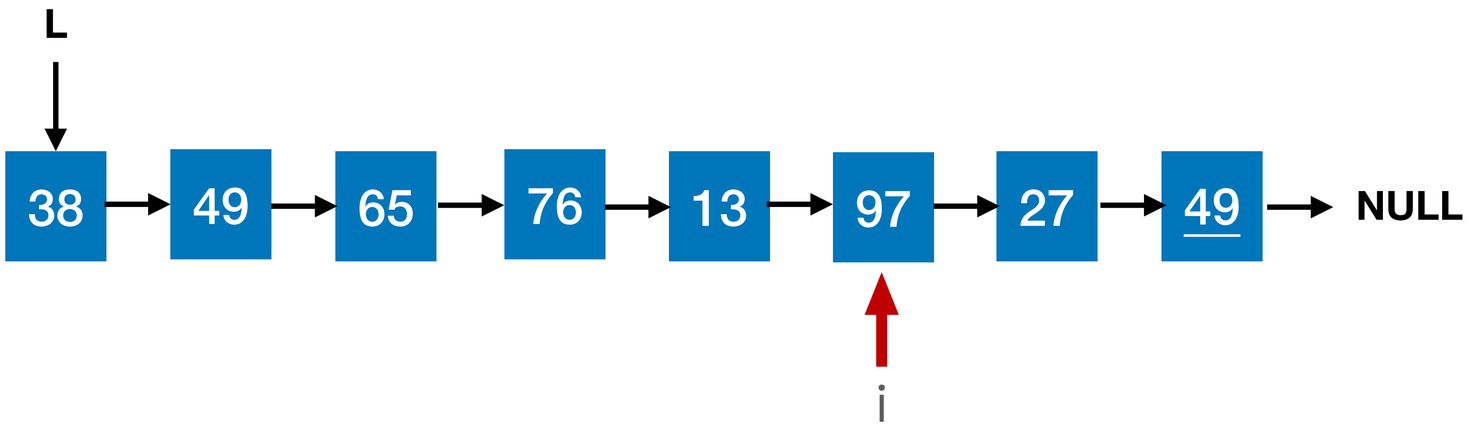
# 冒泡排序是否适用于链表?



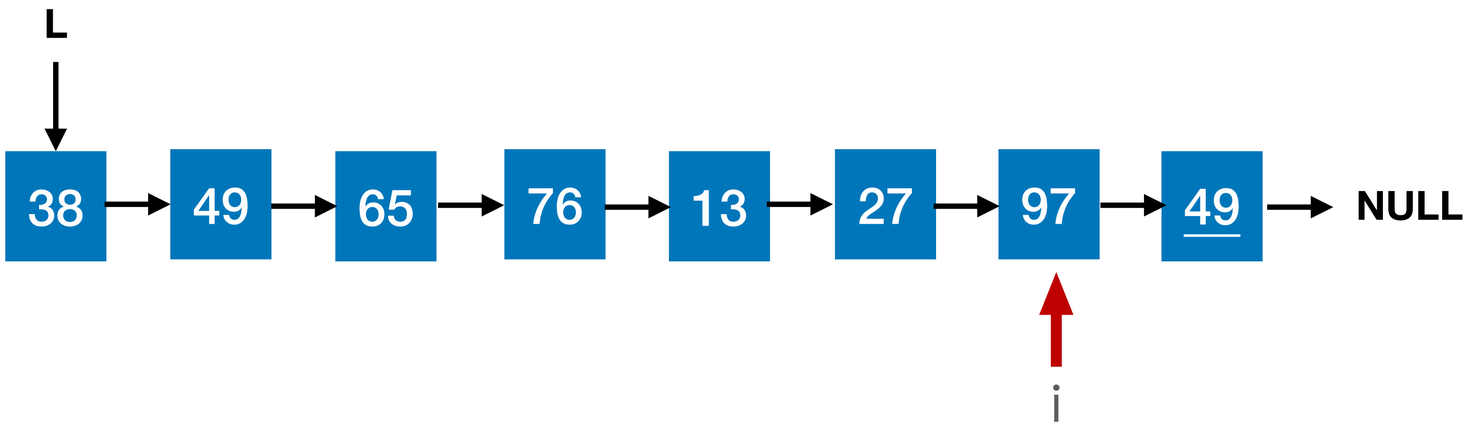
# 冒泡排序是否适用于链表?



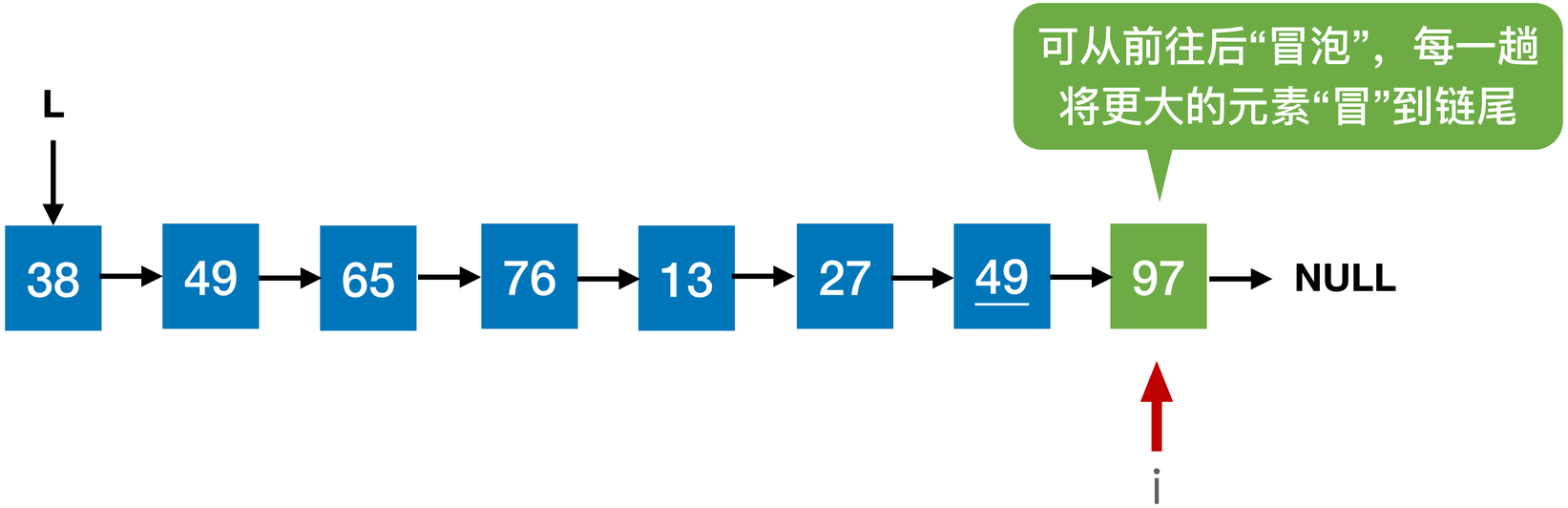
# 冒泡排序是否适用于链表?



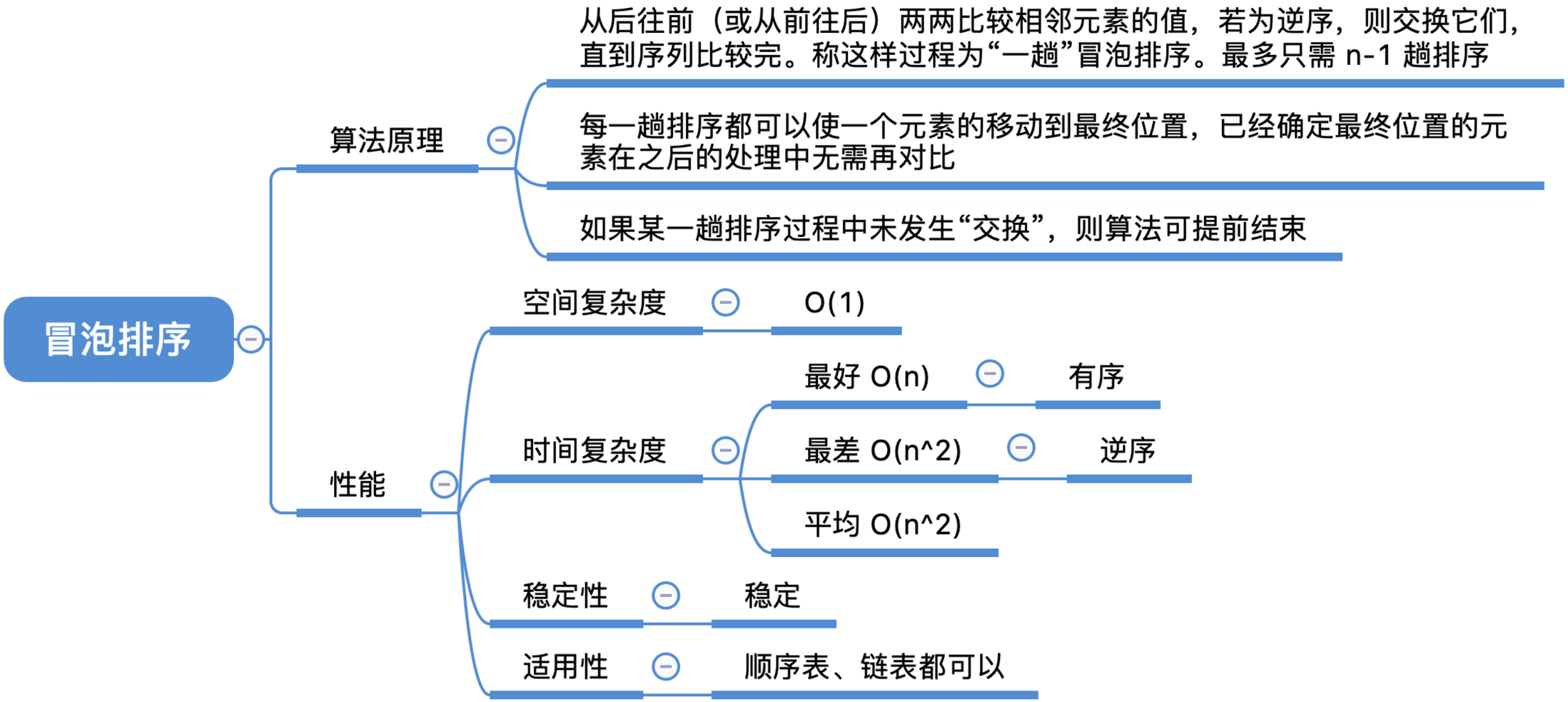
# 冒泡排序是否适用于链表?



# 冒泡排序是否适用于链表?



# 知识回顾与重要考点



# 欢迎大家对本节视频进行评价~

**学员评分：8.3.1 冒泡排序**

扫一扫二维码打开或分享给好友

– 腾讯文档 –

可多人实时在线编辑，权限安全可控

公众号：王道在线

b站：王道计算机教育

抖音：王道计算机考研