数据结构与算法（Python版）

冒泡排序和选择排序算法及分析

陈斌 北京大学 gischen@pku.edu.cn

排序：冒泡排序Bubble Sort

- 冒泡排序的算法思路在于对无序表进行多趟比较交换，
- 每趟包括了多次两两相邻比较，并将逆序的数据项互换位置，最终能将本趟的最大项就位
- 经过n-1趟比较交换，实现整表排序
- 每趟的过程类似于“气泡”在水中不断上浮到水面的经过

排序：冒泡排序Bubble Sort

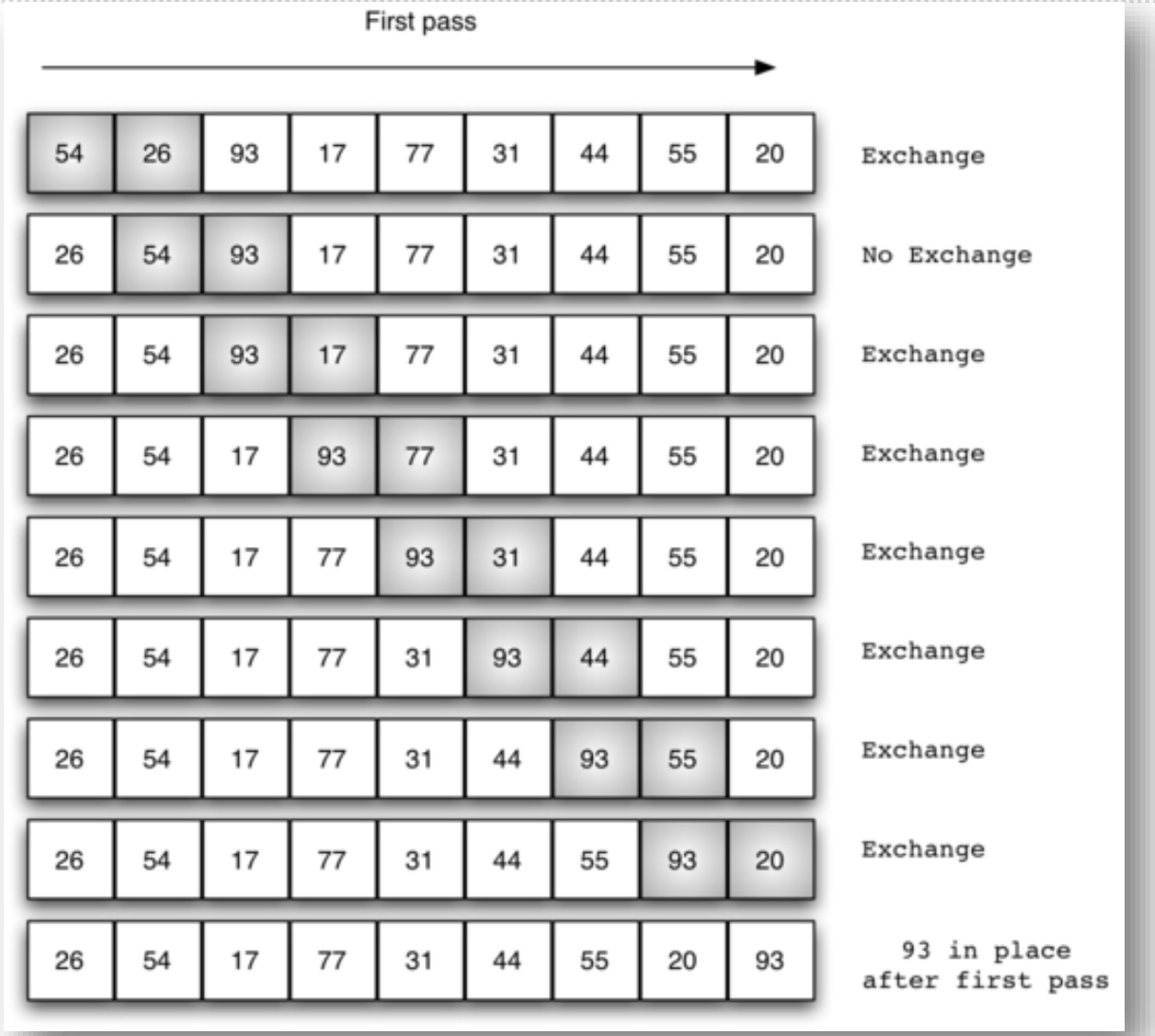
- 第1趟比较交换，共有n-1对相邻数据进行比较

  一旦经过最大项，则最大项会一路交换到达最后一项

- 第2趟比较交换时，最大项已经就位，需要排序的数据减少为n-1，共有n-2对相邻数据进行比较

- 直到第n-1趟完成后，最小项一定在列表首位，就无需再处理了。

冒泡排序：第1趟


First pass
54 26 93 17 77 31 44 55 20 Exchange
26 54 93 17 77 31 44 55 20 No Exchange
26 54 93 17 77 31 44 55 20 Exchange
26 54 17 93 77 31 44 55 20 Exchange
26 54 17 77 93 31 44 55 20 Exchange
26 54 17 77 31 93 44 55 20 Exchange
26 54 17 77 31 44 93 55 20 Exchange
26 54 17 77 31 44 55 93 20 Exchange
26 54 17 77 31 44 55 20 93
93 in place after first pass

冒泡排序：代码


```
def bubbleSort(alist):
    for passnum in range(len(alist)-1,0,-1):
        for i in range(passnum):
            if alist[i]>alist[i+1]:
                temp = alist[i]
                alist[i] = alist[i+1]
                alist[i+1] = temp

alist = [54,26,93,17,77,31,44,55,20]
bubbleSort(alist)
print(alist)
```

n-1趟

序错，交换

Python支持直接交换

```
alist[i],alist[i+1]=alist[i+1],alist[i]
```

https://zh.visualgo.net/sorting

冒泡排序：算法分析

- 无序表初始数据项的排列状况对冒泡排序没有影响
- 算法过程总需要n-1趟，随着趟数的增加，**比对**次数逐步从n-1减少到1，并包括可能发生的数据项**交换**。
- 比对次数是1～n-1的累加：

$$\frac{1}{2}n^2 - \frac{1}{2}n$$

- 比对的时间复杂度是$O(n^2)$

冒泡排序：算法分析

- 关于交换次数，时间复杂度也是$O(n^2)$，通常每次交换包括3次赋值
- **最好**的情况是列表在排序前已经有序，交换次数为0
- **最差**的情况是每次比对都要进行交换，交换次数等于比对次数
- **平均**情况则是最差情况的一半

冒泡排序：算法分析

- 冒泡排序通常作为时间效率较差的排序算法，来作为其它算法的对比基准。
- 其效率主要差在每个数据项在找到其最终位置之前，
- 必须要经过**多次**比对和交换，其中大部分的操作是**无效**的。
- 但有一点优势，就是**无需**任何额外的存储空间开销。

冒泡排序：性能改进

- 另外，通过监测每趟比对是否发生过交换，可以提前确定排序是否完成
- 这也是其它多数排序算法无法做到的
- 如果某趟比对没有发生任何交换，说明列表已经排好序，可以提前结束算法

冒泡排序：性能改进

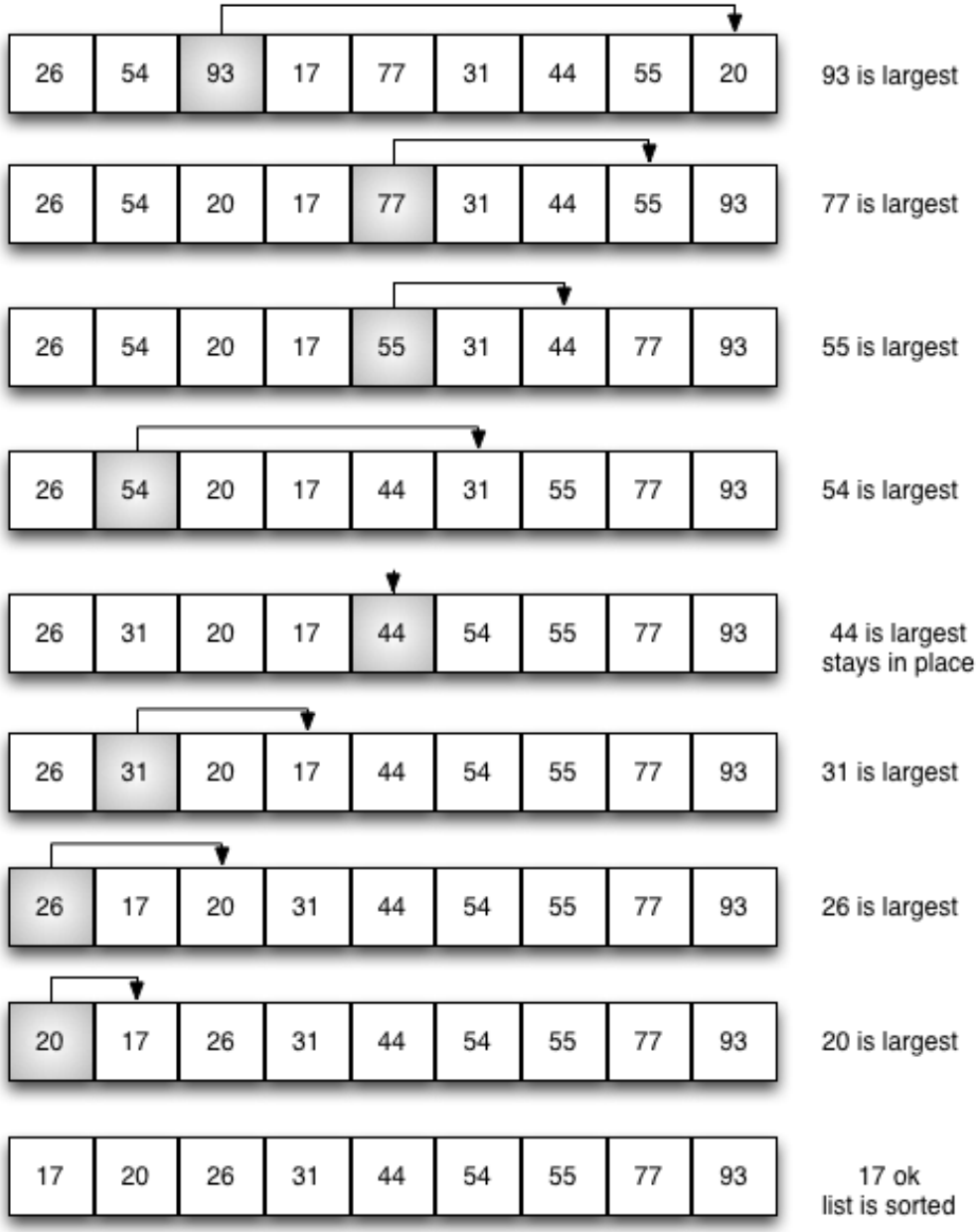

```
def shortBubbleSort(alist):
    exchanges = True
    passnum = len(alist)-1
    while passnum > 0 and exchanges:
        exchanges = False
        for i in range(passnum):
            if alist[i]>alist[i+1]:
                exchanges = True
                temp = alist[i]
                alist[i] = alist[i+1]
                alist[i+1] = temp
        passnum = passnum-1

alist=[20,30,40,90,50,60,70,80,100,110]
shortBubbleSort(alist)
print(alist)
```

选择排序Selection Sort

- 选择排序对冒泡排序进行了改进，保留了其基本的多趟比对思路，每趟都使当前最大项就位。
- 但选择排序对交换进行了削减，相比起冒泡排序进行多次交换，每趟仅进行1次交换，记录最大项的所在位置，最后再跟本趟最后一项交换
- 选择排序的时间复杂度比冒泡排序稍优

  比对次数不变，还是$O(n^2)$

  交换次数则减少为$O(n)$


26 54 93 17 77 31 44 55 20
93 is largest
26 54 20 17 77 31 44 55 93
77 is largest
26 54 20 17 55 31 44 77 93
55 is largest
26 54 20 17 44 31 55 77 93
54 is largest
26 31 20 17 44 54 55 77 93
44 is largest
stays in place
26 31 20 17 44 54 55 77 93
31 is largest
26 17 20 31 44 54 55 77 93
26 is largest
20 17 26 31 44 54 55 77 93
20 is largest
17 20 26 31 44 54 55 77 93
17 ok
list is sorted

选择排序：代码


```
def selectionSort(alist):
    for fillslot in range(len(alist)-1,0,-1):
        positionOfMax=0
        for location in range(1,fillslot+1):
            if alist[location]>alist[positionOfMax]:
                positionOfMax = location

        temp = alist[fillslot]
        alist[fillslot] = alist[positionOfMax]
        alist[positionOfMax] = temp
```