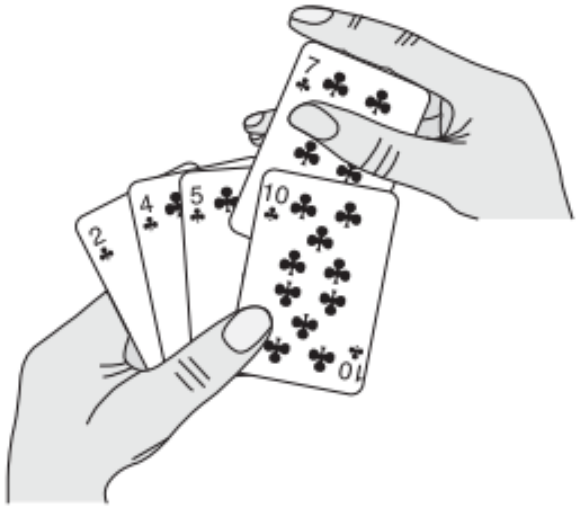
数据结构与算法（Python版）

插入排序算法及分析

陈斌 北京大学 gischen@pku.edu.cn

插入排序Insertion Sort

- 插入排序时间复杂度仍然是$O(n^2)$，但算法思路与冒泡排序、选择排序不同
- 插入排序维持一个已排好序的子列表，其位置始终在列表的前部，然后逐步扩大这个子列表直到全表

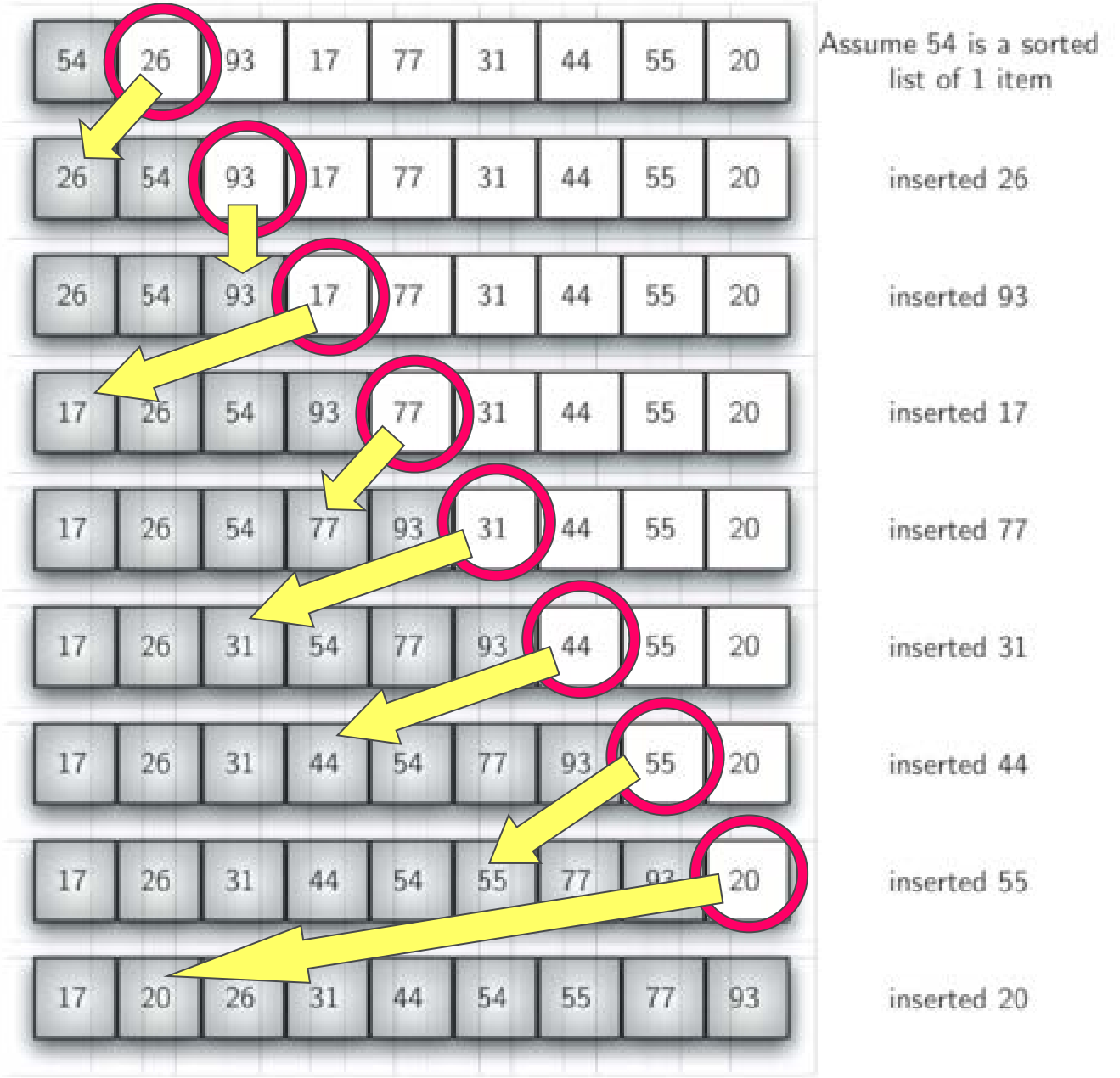


插入排序Insertion Sort

- 第1趟，子列表仅包含第1个数据项，将第2个数据项作为“新项”插入到子列表的合适位置中，这样已排序的子列表就包含了2个数据项
- 第2趟，再继续将第3个数据项跟前2个数据项比对，并移动比自身大的数据项，空出位置来，以便加入到子列表中
- 经过n-1趟比对和插入，子列表扩展到全表，排序完成

插入排序Insertion Sort

- 插入排序的比对主要用来寻找“**新项**”的**插入位置**
- 最差情况是每趟都与子列表中所有项进行比对，总比对次数与冒泡排序相同，数量级仍是$O(n^2)$
- 最好情况，列表已经排好序的时候，每趟仅需1次比对，总次数是O(n)


54 26 93 17 77 31 44 55 20
Assume 54 is a sorted list of 1 item
26 54 93 17 77 31 44 55 20
inserted 26
26 54 93 17 77 31 44 55 20
inserted 93
17 26 54 93 77 31 44 55 20
inserted 17
17 26 54 77 93 31 44 55 20
inserted 77
17 26 31 54 77 93 44 55 20
inserted 31
17 26 31 44 54 77 93 55 20
inserted 44
17 26 31 44 54 55 77 93 20
inserted 55
17 20 26 31 44 54 55 77 93
inserted 20

插入排序：思路


17 26 54 77 93 31 44 55 20
Need to insert 31 back into the sorted list
17 26 54 77 93 44 55 20
93>31 so shift it to the right
17 26 54 77 93 44 55 20
77>31 so shift it to the right
17 26 54 77 93 44 55 20
54>31 so shift it to the right
17 26 31 54 77 93 44 55 20
26<31 so insert 31 in this position
比对，移动所有
比“新项”大的数据项

插入排序：代码


```
def insertionSort(alist):
    for index in range(1,len(alist)):

        currentvalue = alist[index]
        position = index

        while position>0 and alist[position-1]>currentvalue:
            alist[position]=alist[position-1]
            position = position-1

        alist[position]=currentvalue
```

新项/插入项

比对、移动

插入新项

由于移动操作仅包含1次赋值，是交换操作的1/3，所以插入排序性能会较好一些。