数据结构与算法（Python版）

归并排序算法及分析

陈斌 北京大学 gischen@pku.edu.cn

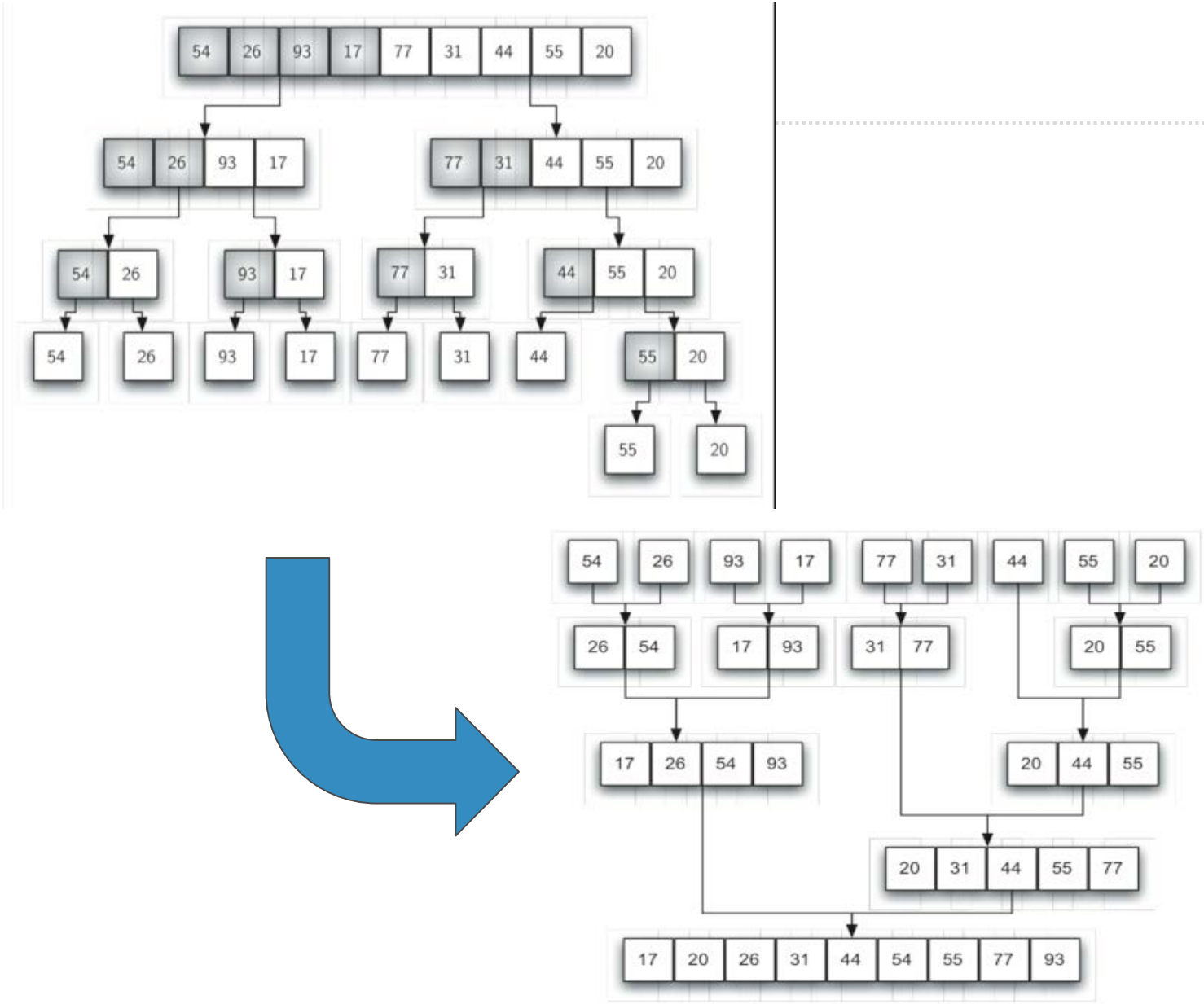
归并排序Merge Sort

- 下面我们来看看**分治策略**在排序中的应用
- 归并排序是递归算法，思路是将数据表持续分裂为两半，对两半分别进行归并排序

  递归的基本结束条件是：数据表仅有1个数据项，自然是排好序的；

  缩小规模：将数据表分裂为相等的两半，规模减为原来的二分之一；

  调用自身：将两半分别调用自身排序，然后将分别排好序的两半进行归并，得到排好序的数据表


54 26 93 17 77 31 44 55 20
54 26 93 17
77 31 44 55 20
54 26
93 17
77 31
44 55 20
54
26
93
17
77
31
44
55 20
55
20
54 26 93 17 77 31 44 55 20
26 54
17 93
31 77
20 55
17 26 54 93
20 44 55
20 31 44 55 77
17 20 26 31 44 54 55 77 93

```
def mergeSort(alist):
##    print("Splitting ",alist)
    if len(alist)>1:
        mid = len(alist)//2
        lefthalf = alist[:mid]
        righthalf = alist[mid:]

        mergeSort(lefthalf)
        mergeSort(righthalf)

        i= j= k= 0
        while i<len(lefthalf) and j<len(righthalf):
            if lefthalf[i]<righthalf[j]:
                alist[k]=lefthalf[i]
                i=i+1
            else:
                alist[k]=righthalf[j]
                j=j+1
            k=k+1

        while i<len(lefthalf):
            alist[k]=lefthalf[i]
            i=i+1
            k=k+1

        while j<len(righthalf):
            alist[k]=righthalf[j]
            j=j+1
            k=k+1
##    print("Merging ",alist)
```

基本结束条件

递归调用

拉链式交错把左右半部
从小到大归并到结果列表中

归并左半部剩余项

归并右半部剩余项

另一个归并排序代码（更Pythonic）


```python
# merge sort
# 归并排序


def merge_sort(lst):
    # 递归结束条件
    if len(lst) <= 1:
        return lst

    # 分解问题，并递归调用
    middle = len(lst) // 2
    left = merge_sort(lst[:middle])  # 左半部排好序
    right = merge_sort(lst[middle:])  # 右半部排好序

    # 合并左右半部，完成排序
    merged = []
    while left and right:
        if left[0] <= right[0]:
            merged.append(left.pop(0))
        else:
            merged.append(right.pop(0))

    merged.extend(right if right else left)
    return merged
```

归并排序：算法分析

- 将归并排序分为两个过程来分析：**分裂**和**归并**
- 分裂的过程，借鉴二分查找中的分析结果，是对数复杂度，时间复杂度为O(log n)
- 归并的过程，相对于分裂的每个部分，其所有数据项都会被比较和放置一次，所以是线性复杂度，其时间复杂度是O(n)

  综合考虑，每次分裂的部分都进行一次O(n)的数据项归并，总的时间复杂度是O(nlog n)

归并排序：算法分析

- 最后，我们还是注意到两个切片操作

  为了时间复杂度分析精确起见，

  可以通过取消切片操作，改为传递两个分裂部分的起始点和终止点，也是没问题的，

  只是算法可读性稍微牺牲一点点。

- 我们注意到归并排序算法使用了**额外1倍**的存储空间用于归并

- 这个特性在对特大数据集进行排序的时候要考虑进去