Python入门与算法

第六讲 常用排序算法原理与应用
Principle and Application of Sorting Algorithm

林平之 老师

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
知乎专栏: http://zhuanlan.zhihu.com/jiuzhang
微博: http://www.weibo.com/ninechapter
官网: www.jiuzhang.com

版权声明

九章课程不提供视频，也严禁录制视频的侵权行为

否则将追求法律责任和经济赔偿

请不要缺课

本节重点

归并排序 Merge Sort

快速排序 Quick Sort

在Python中使用排序

你会掌握

如何用分治法(Divide and Conquer)解决排序问题

如何分析分治算法的时间和空间复杂度

如何有效避免快速排序的最坏情况

学习目标

自学内容，以下三种排序时间复杂度均为O(n^2)

选择排序 Selection Sort

插入排序 Insertion Sort

冒泡排序 Bubble Sort

演示界面

：http://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html

归并排序和快速排序 Merge sort & Quick sort

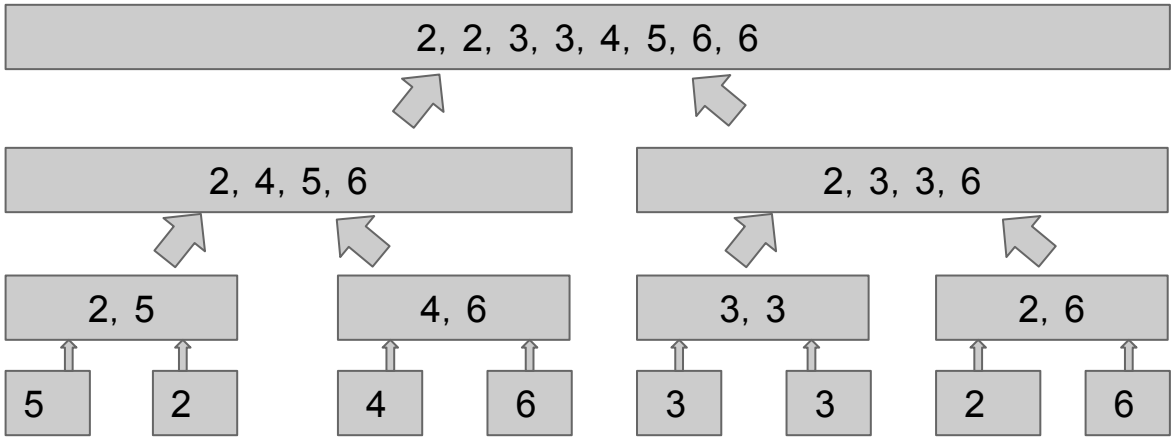
- Merge sort
  - 最坏时间复杂度O(nlogn)
  - 稳定的排序算法
- Quick sort
  - 均摊复杂度(亦或者是平均复杂度)O(nlogn)
  - 不是稳定的排序算法

什么是稳定的排序算法：简单的来说，就是数组里有两个相同的数，那么不管排序前 还是排序后，原来在前面的一定 还是在前面。

Merge Sort

实例分析

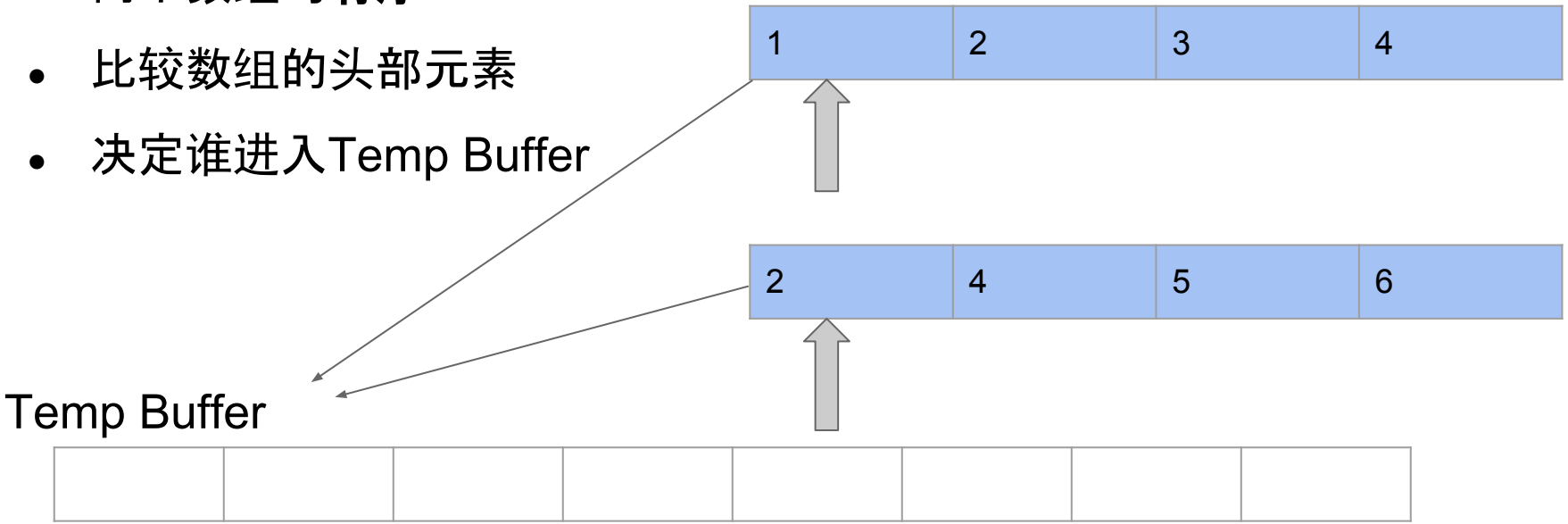
Array = [5, 2, 4, 6, 3, 3, 2, 6] 不断递归，然后向上合并：



Array=[5, 2, 4, 6, 3, 3, 2, 6]

合并两个有序数组 Merge Two Sorted Arrays

- 两个数组均**有序**
- 比较数组的头部元素
- 决定谁进入Temp Buffer



Merge Two Sorted Arrays

http://www.lintcode.com/en/problem/merge-two-sorted-arrays/

http://www.jiuzhang.com/solutions/merge-two-sorted-arrays/

Sort Integer II

http://www.lintcode.com/en/problem/sort-integers-ii/

http://www.jiuzhang.com/solutions/sort-integers-ii/

归并排序 Merge Sort

归并排序(Merge Sort)思想 - 分治

- 把数组均分成左右两半
- 将左右两半分别排序（递归）
- 将排好序的两半数组合并（merge）

归并排序 Merge Sort

- 时间复杂度
  - O(nlogn)
- 空间复杂度
  - O(n)

Python语言实现参考：

http://www.jiuzhang.com/solutions/merge-sort

Quick Sort

快速排序 Quick Sort

快速排序(quick sort)思想

- 把数组分为两边，使得：

  数组的左边小于等于数组的右边

这个过程叫做：Partition

- （也就是意味着：左边的最大数小于等于右边的最小数）
- 对左右两部分数组分别继续排序（递归）

快速排序 Quick Sort

如何选基准数Pivot

- 选当前数组的第一个数
  - pivot = Array[0]
- 在当前数组中随机选一个数
  - pivot = Array[rand.nextInt(end - start + 1) + start]
  - 生成start ~ end之间的一个随机数

Partition Array

http://www.lintcode.com/en/problem/partition-array/

http://www.jiuzhang.com/solutions/partition-array/

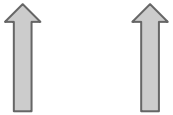
Partition

Partition Array: [50, 70, 40, 30, 40, 80, 90, 10]

Pivot = 50 选第一个元素

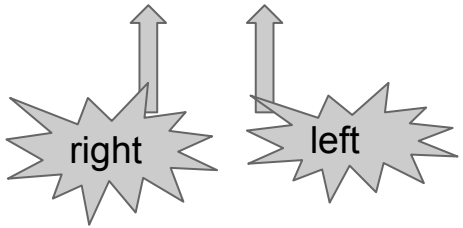
First round: [10, 70, 40, 30, 40, 80, 90, 50]

Second round: [10, 40, 40, 30, 70, 80, 90, 50]



Partition

Third round: [10, 40, 40, 30, 70, 80, 90, 50]



Quick Sort中Partition总结

如何把数组分为两部分（partition）

1. 两个指针，分别指向当前数组的头和尾
2. 如果当前数小于Pivot，左指针继续向右移动，直到左边指针指向的数大于等于基准数
3. 如果当前数大于Pivot，右指针继续向左移动，直到右边指针指向的数小于等于基准数
4. 交换两个指针指向的数，然后两个指针分别移动一位
5. 回到第2步，直到left >= right为止

提问：为什么Quick Sort中我们需要使用小于和大于就移动，不轻易使用小于等于和大于等于？

距离说明Partition不适用小于等于的原因

如数组Array = [3, 4, 4, 5, 6, 7, 4]

Pivot = 3

这会发生什么？

快速排序 Quick Sort

如何确定继续递归的左右两边的边界

Pivot: 4

Partition之前

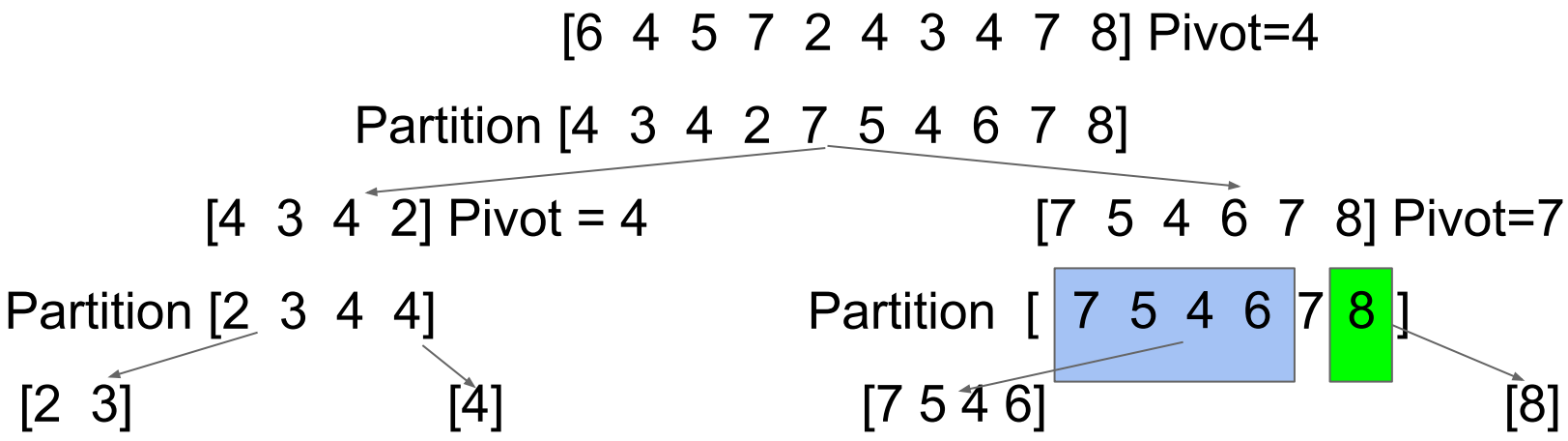
[6 4 5 7 2 4 3 4 7 8]

Partition之后

[4 3 4 2 7 5 4 6 7 8]

原区间为【start, end】
继续需要递归的左侧区间为：
【start, right】
继续需要递归的右侧区间为：
【left, end】

Quick Sort 过程



……

最后的结果为:[2 3 4 4 4 5 6 7 7 8]

Coding 实现 Quick sort

快速排序 Quick Sort

九章算法

- 时间复杂度
  - O(nlogn)—平均情况
  - O(n^2)—最坏情况
- 空间复杂度
  - O(logn)

Python语言参考程序：

http://www.jiuzhang.com/solutions/quick-sort

Python中使用Sort

在Python中使用排序

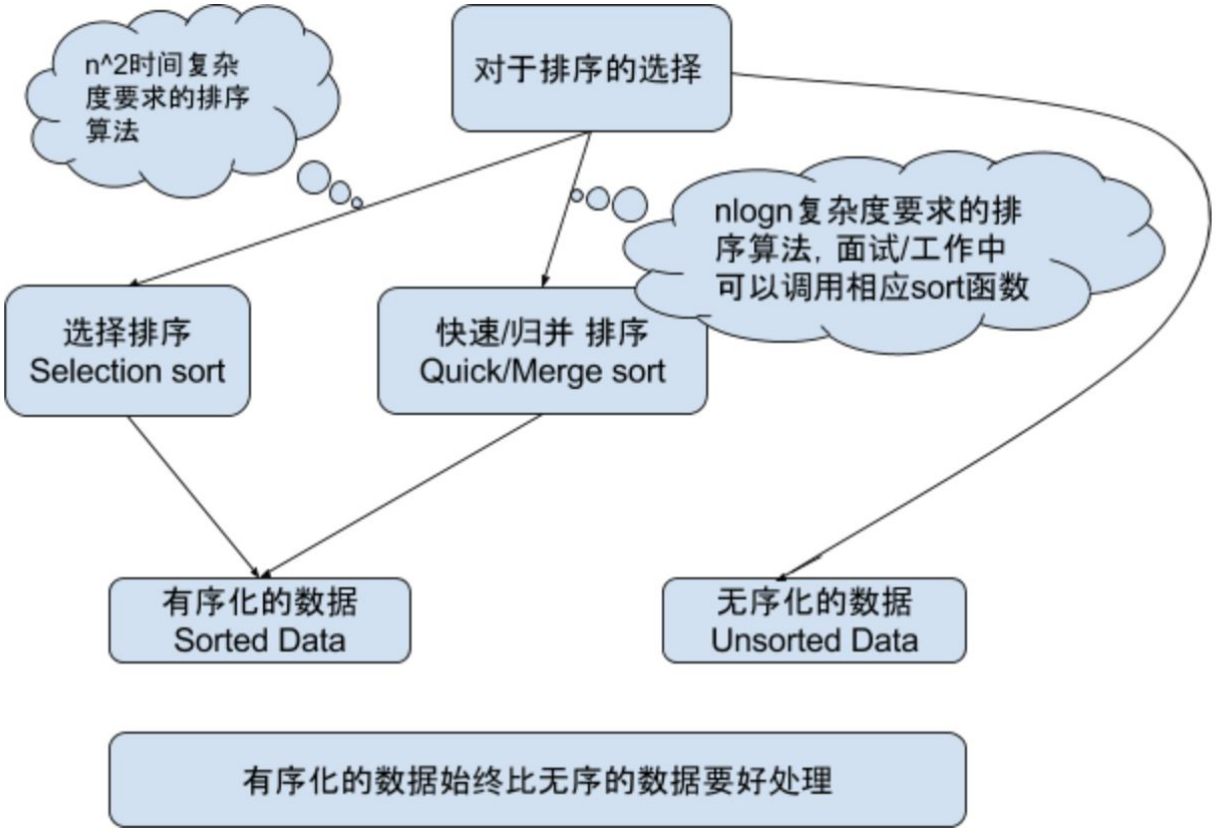
- 对list进行排序
  - 利用list的成员函数sort()排序
  - 利用内置函数sorted()进行排序

对List排序


```
1  nums = [2, 4, 5, 1, 10]
2
3  nums.sort()
4
5  nums = sorted(nums)
6
7
8
```

- sort() 对list进行排序,改变list的值
- sorted() 产生一个新的list，不改变原list的值

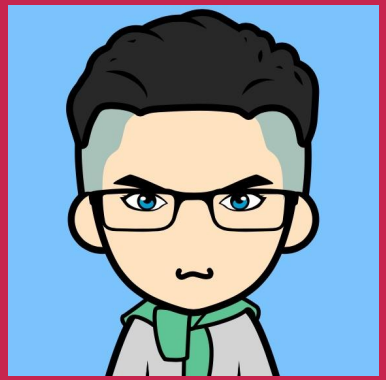
排序的选择


n^2时间复杂度要求的排序算法
对于排序的选择
nlogn复杂度要求的排序算法，面试/工作中可以调用相应sort函数
选择排序
Selection sort
快速/归并 排序
Quick/Merge sort
有序化的数据
Sorted Data
无序化的数据
Unsorted Data
有序化的数据始终比无序的数据要好处理

排序的应用

1. 加速检索，可以在有序数据上进行二分查找，比如sstable
2. 更好的进行数据整理与合并

Q & A



谢谢大家