

九章算法基础班

第八讲 排序算法

课程版本：v2.0 张三疯 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

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

- 普通排序算法
- 归并排序 (Merge sort)
- 快速排序 (Quick sort)
- 课程总结

课程回顾

- Python的基本数据结构之一
 - 集合中存储**非重复**的**无序**数据
 - set中的元素不一定是同一类型，非常灵活

```
set_1 = set([12, 15.6, True, 'hello'])  
set_2 = {12, 15.6, True, 'hello'}  
set_3 = set('hello')
```



Bag

- Set的常见操作
 - 增 (Create) : add, update
 - 查 (Read) : 迭代 (iteration) , in
 - 删 (Delete) : remove, clear, discard, pop
 - 其他 : len

- Set的集合间操作：生成新的集合
 - 并集：union, |
 - 交集：intersection , &
 - 差集：difference , -
 - 对称差：symmetric_difference , ^

- Python的基本数据结构之一
 - 字典中存储key非重复的无序的key-value pairs
 - dict可能是最灵活的内置数据结构
 - 别名：index，map

```
dict_1 = {}  
dict_2 = {'spam': 2, 'eggs': 3, 'food': {'ham': 1, 'ice': 2}}  
dict_3 = dict(zip(['spam', 'eggs'], [2, 3]))
```


- Dict的常见操作
 - 增 (Create) : 索引赋值 , update
 - 查 (Read) : 索引 , 迭代 (iteration) , in, get, keys, values, items
 - 改 (Update) : 索引赋值
 - 删 (Delete) : pop, del
 - 其他 : len

- 主要关注插入和查找操作
 - List实现
 - add: $O(n)$ find: $O(n)$
 - 平衡的BST实现
 - add: $O(\log n)$ find: $O(\log n)$
 - Hash table实现
 - add: $O(1)$ find: $O(1)$

- 分治法 (divide and conquer)
 - 将一个大问题分解成多个独立的小问题：分
 - 分别解决每个小问题 (小问题和大问题是同一类问题，可以用递归)
 - 将小问题的解合并，从而得到大问题的解：合

普通排序算法

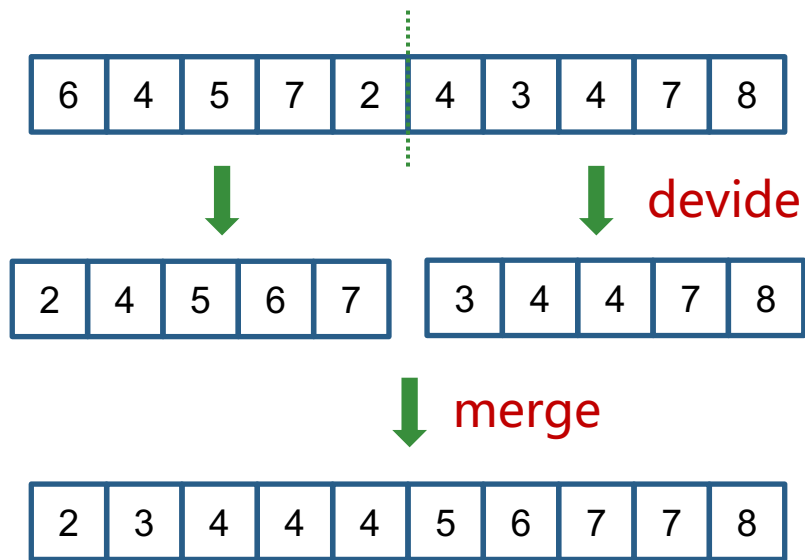
- 选择排序 (Selection sort)
 - http://www.algolist.net/Algorithms/Sorting/Selection_sort
- 插入排序 (Insertion sort)
 - http://www.algolist.net/Algorithms/Sorting/Insertion_sort
- 冒泡排序 (Bubble sort)
 - http://www.algolist.net/Algorithms/Sorting/Bubble_sort

- 复杂度
 - 时间复杂度： $O(n^2)$
 - 空间复杂度： $O(1)$
- 演示动画
 - <http://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html>

归并排序 (Merge sort)

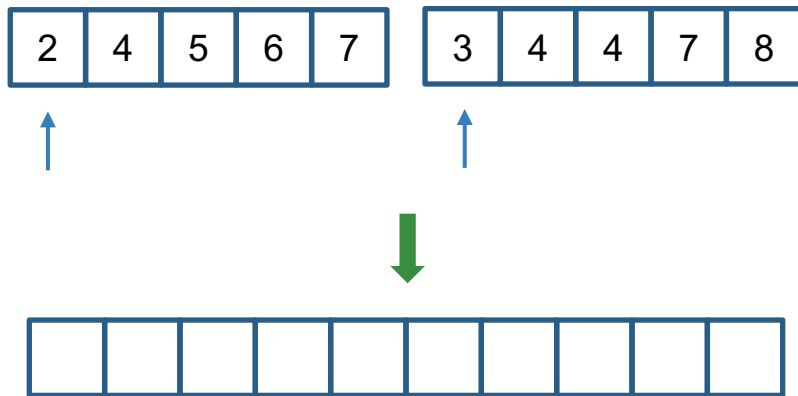
- 归并排序 (merge sort) - 分治
 - 把数组均分成左右两半
 - 将左右两半分别排序 (递归)
 - 将排好序的两半数组合并 (merge)

- 归并排序 (merge sort)

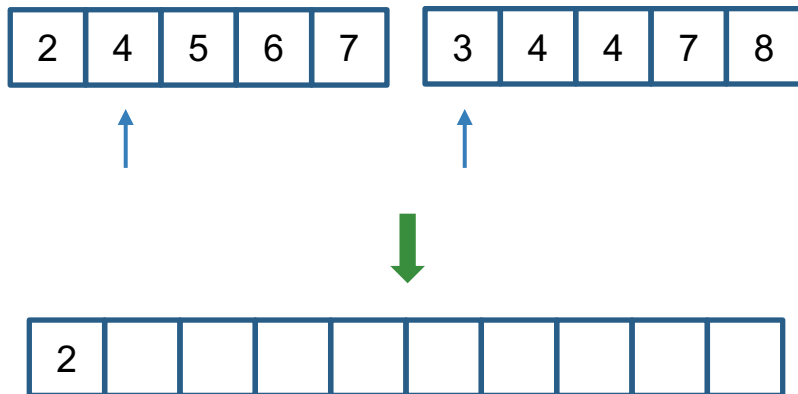


- 如何merge ?
 - 练习一：merge two sorted arrays
 - <https://www.lintcode.com/problem/merge-two-sorted-arrays/>
 - <https://www.jiuzhang.com/solution/merge-two-sorted-arrays/>

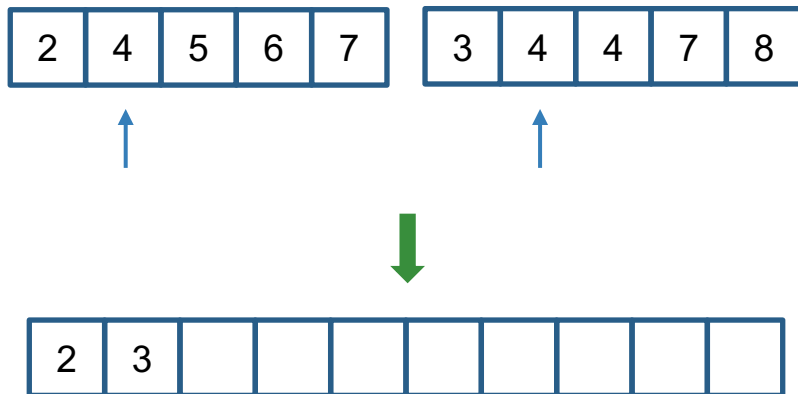
- 如何merge ?



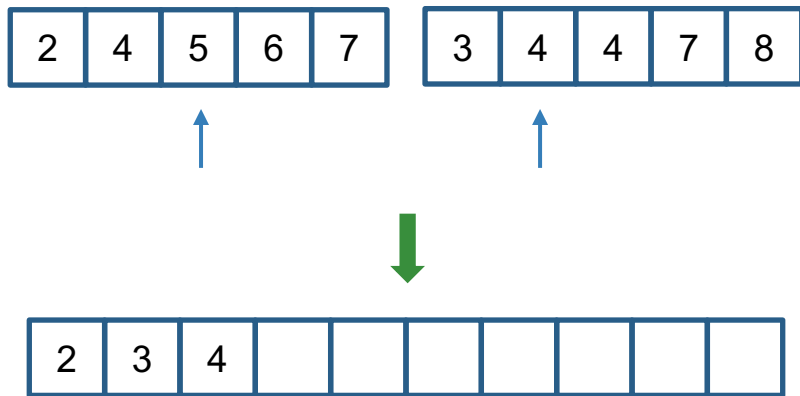
- 如何merge ?



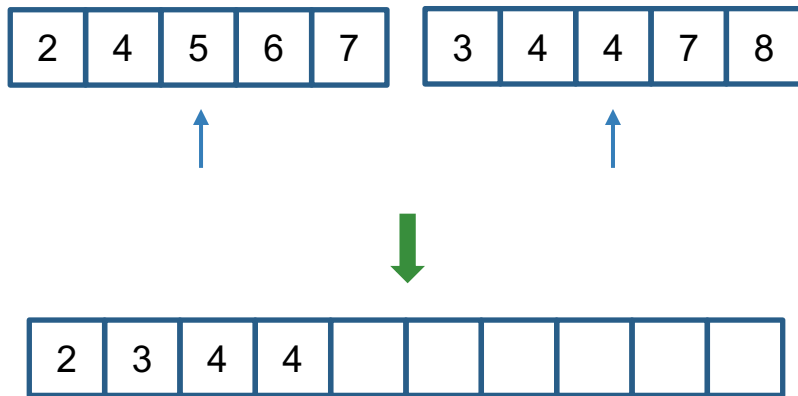
- 如何merge ?



- 如何merge ?

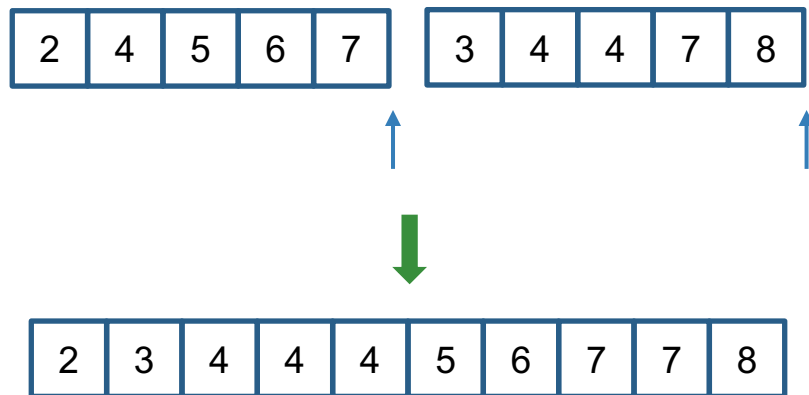


- 如何merge ?



以此类推

- 如何merge ?
 - 结果



- 分治法的代码思考方式

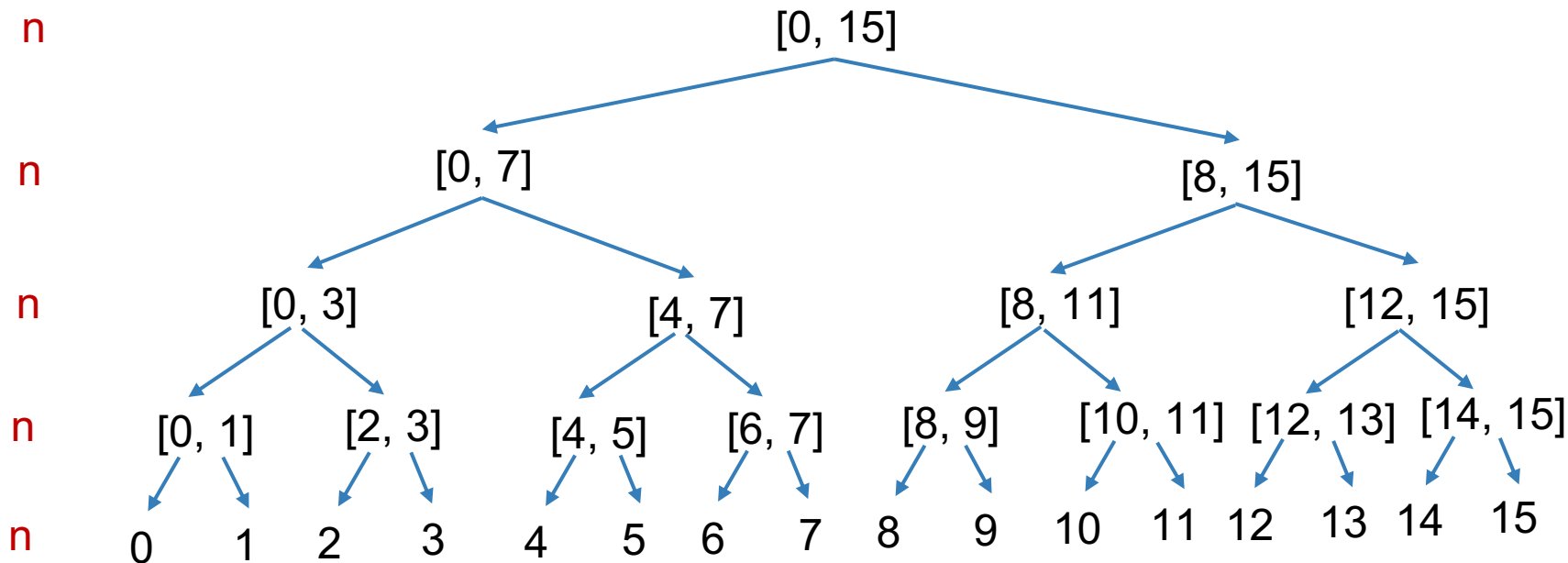
```
8  def merge_sort_helper(array, left, right):
9      if left >= right:
10         return
11
12     mid = (left + right) // 2
13     merge_sort_helper(array, left, mid)
14     merge_sort_helper(array, mid + 1, right)
```

- 分治法的代码思考方式
 - 先假设小的任务已经完成（实际上未完成）
 - 在此基础上完成大的任务，此时原来小的任务也就一并完成了

```
8  def merge_sort_helper(array, left, right):
9      if left >= right:
10         return
11
12     mid = (left + right) // 2
13     merge_sort_helper(array, left, mid)
14     merge_sort_helper(array, mid + 1, right)
15     merge(array, left, right)
```

归并排序

- 时间复杂度： $O(n\log n)$



- 空间复杂度： $O(n)$
 - 栈空间： $O(\log n)$
 - 堆空间： $O(n \log n) \rightarrow O(n)$
- 代码参考
 - <https://www.jiuzhang.com/solutions/merge-sort>

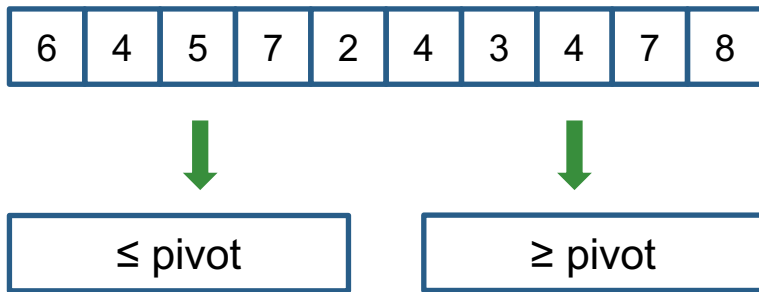
- 练习二：面试真题
 - Reverse Pairs
 - <https://www.lintcode.com/problem/reverse-pairs/>
 - <https://www.jiuzhang.com/solution/reverse-pairs/>

快速排序 (Quick sort)

- 快速排序 (quick sort)
 - 20世纪以来十大经典算法
 - <https://www.quora.com/What-are-the-top-10-algorithms-of-the-20th-century>

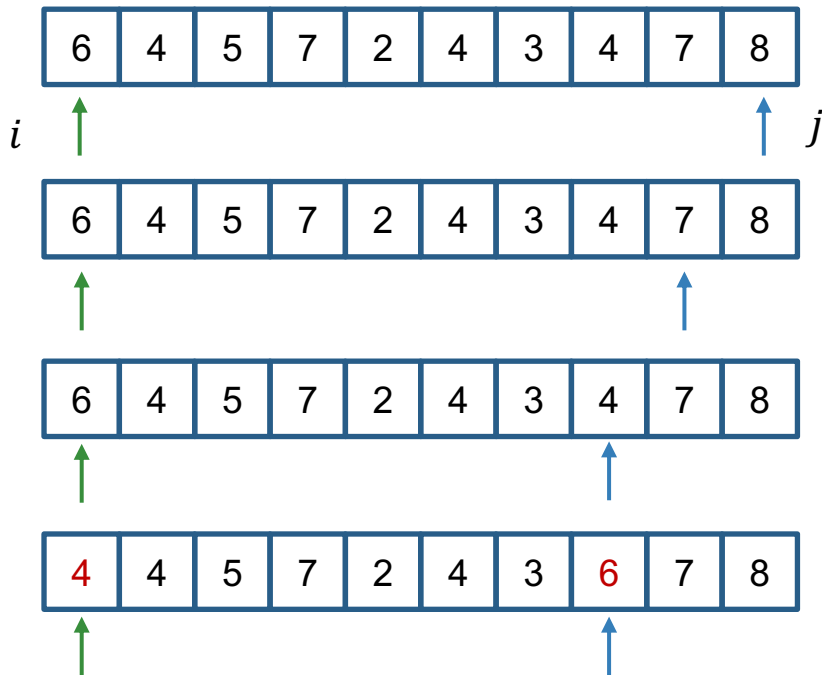
- 快速排序 (quick sort) 思想
 - 把数组分为两边，使得：数组的左边小于等于数组的右边（左右两边长度不一定相等）
 - 对左右两部分数组分别排序（递归）
- 先整体后局部

- 快速排序
 - 选取基准数 (pivot)
 - 将数组分割为两部分，长度不一定相等 (partition)
 - 递归处理子问题



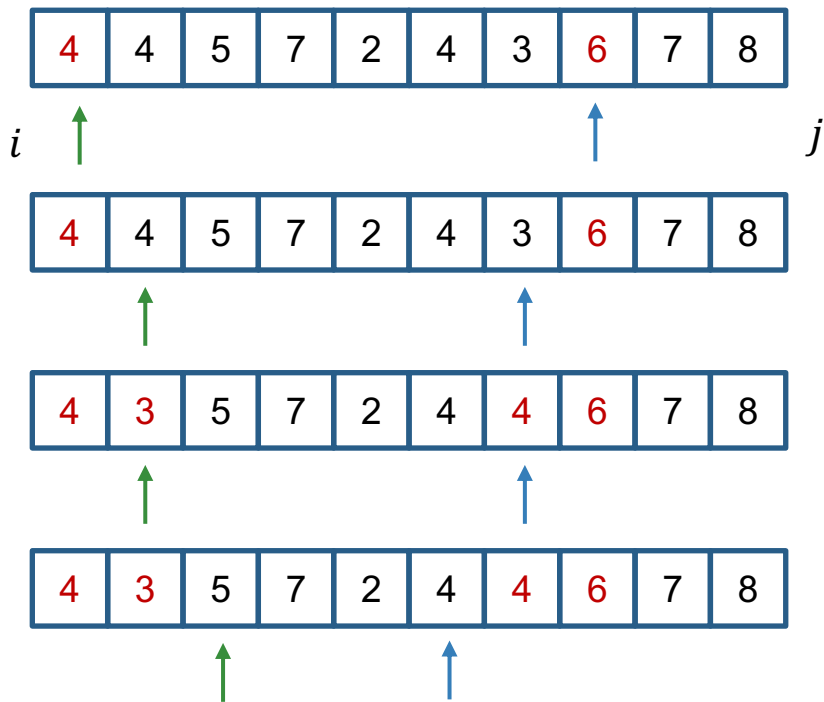
- 如何把数组分为两部分 (partition)
 - 两个指针，分别指向当前数组的头和尾
 - 移动左边的指针，直到左边指针指向的数大于等于基准数
 - 移动右边的指针，直到右边指针指向的数小于等于基准数
 - 交换两个指针指向的数
 - 回到第2步，直到两个指针相遇

- 如何做Partition ?
 - 当 i 指向的值小于pivot右移
 - 当 j 指向的值大于pivot左移
 - 否则，交换值



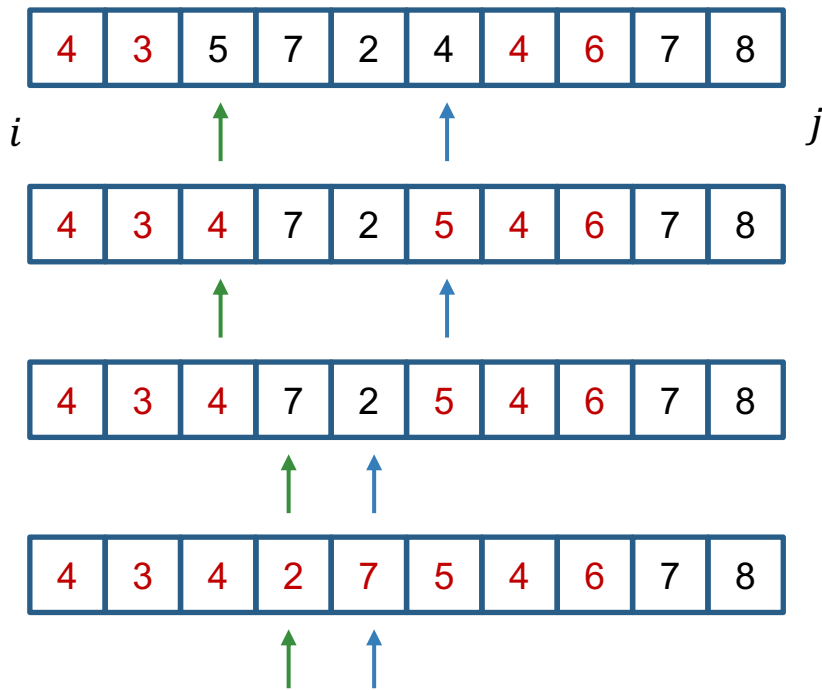
假设pivot取值为4

- 如何做Partition ?
 - 当 i 指向的值小于pivot右移
 - 当 j 指向的值大于pivot左移
 - 否则，交换值



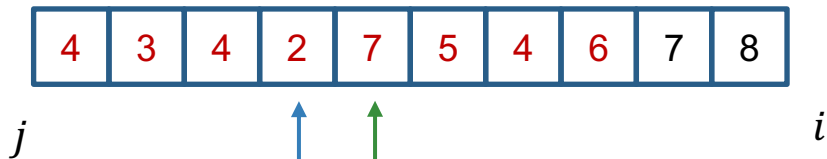
假设pivot取值为4

- 如何做Partition ?
 - 当 i 指向的值小于pivot右移
 - 当 j 指向的值大于pivot左移
 - 否则，交换值



假设pivot取值为4

- 两个子问题的边界
 - $[left, j]$
 - $[i, right]$



- 递归

假设pivot取值为4

- 代码演示
 - 先理解
 - 再记忆
 - 再当成模板

- 如何选基准数
 - 选当前数组的第一个数？
 - 在当前数组中随机选一个数：`random.randint(start, end)`
- 退化
 - $O(n^2)$

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

- 如何确定子问题左右两边的边界

- pivot = 2

| | | | |
|---|---|---|---|
| 3 | 1 | 1 | 2 |
|---|---|---|---|

- pivot = 2

| | | | | |
|---|---|---|---|---|
| 3 | 4 | 2 | 2 | 1 |
|---|---|---|---|---|

- 一定是 [left, j] 和 [i, right]

- 为什么当值等于pivot时也交换呢？
 - 保证子问题规模一定小于原问题
 - 使子问题规模尽量相等，降低时间复杂度



- 时间复杂度
 - $O(n \log n)$ - 平均情况
 - $O(n^2)$ - 最坏情况
- 空间复杂度
 - $O(\log n)$
- 代码参考
 - <https://www.jiuzhang.com/solutions/quick-sort>

- 练习三
 - Partition Array
 - <https://www.lintcode.com/problem/partition-array/>
 - <https://www.jiuzhang.com/solution/partition-array/>

- 练习四
 - Sort Integers II
 - <https://www.lintcode.com/problem/sort-integers-ii/>
 - <https://www.jiuzhang.com/solution/sort-integers-ii/>

- 对list进行排序
 - 利用list的成员函数sort()排序：原地排序
 - 利用内置函数sorted()进行排序：生成新的list
 - 重写__lt__和__gt__函数来定义object的比较方法

课程总结

- Python language
 - Dynamically typed language
 - Variable model
 - Control flow
 - Function : copy address, execute

- CS basics
 - Unicode
 - OOP
 - Reference
 - Memory model : heap and stack
 - Time/ space complexity analysis
 - Test case

- Data structure
 - List, Tuple, String
 - LinkedList
 - Stack, Queue
 - Binary tree, BST
 - Set, Dictionary

- Algorithm
 - Enumeration
 - DFS, BFS
 - Divide and conquer
 - Sort

- <https://www.jiuzhang.com/course/13/questionnaire/>



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](#)

微博: <http://www.weibo.com/ninechapter>

官网: www.jiuzhang.com



谢谢大家