Python语法基础
Introduction to Python II

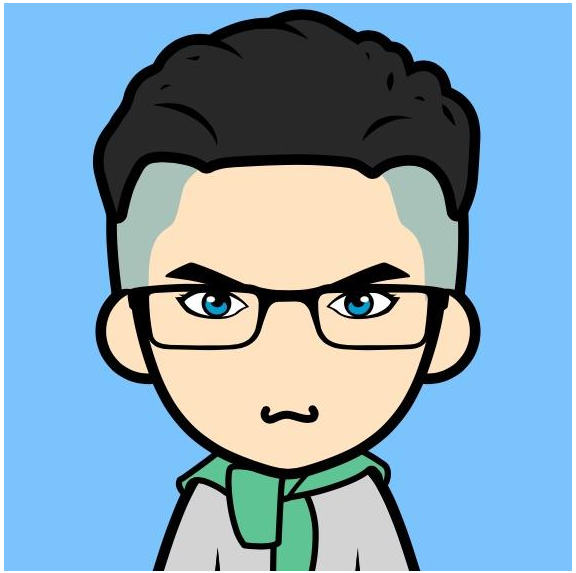
林平之老师

扫描二维码关注微信/小程序，获取最新面试题及权威解答

微信扫一扫，使用小程序

微信扫一扫，关注微信公众号

自我介绍



林老师

全国算法竞赛一等奖

国内TOP2名校毕业

参加国家信息学竞赛NOI

前FLAG工程师

拥有丰富的面试经验

大纲

- Python基础类型之
  - Dict & Set
- Python实现快速排序和归并排序
- Python中如何使用排序方法
- Lambda, Map, Filter, Reduce函数的使用
- Q & A

Dict

Python的基础类型Dict

- dict全称dictionary，其他语言中也称为map或者hash，使用键-值（key-value）存储

- 存储映射关系
-

```
names = ['Alice', 'Bob', 'linpz']
scores = [99, 97, 98]

d = {'Alice': 99, 'Bob': 97, 'linpz': 98}
```

Python的基础类型Dict

- 提问：dict为什么那么快？

- key-value存储方式：Insert放进去的时候，根据key计算value的存放位置，这样，取的时候再根据key就可以得到value存放的位置
-

```
names = dict()  # names = {}
names['ta'] = '96'
print(names['ta'])
```

- 访问不存在的key时会出现错误

```
print(names['linghc'])

KeyError: 'linghc'
```

- 判断key是否存在的两种方是:

```
'ta' in names

names.get('linpz')  # return None if key doesn't exist
```

Python的基础类型Dict

从Dict中获取key对应的值，当key不存在的时候范围一个default的值

```
# P:
linpz = {'yanzhi': 5, 'name': 8}
linpz['yanzhi'] = linpz.get('yanzhi', 0) + 1

# linpz = {'yanzhi': 6, 'nice': 8}

#NP:
linpz = {'yanzhi': 5, 'name': 8}
if 'yanzhi' in linpz:
    linpz['yanzhi'] += 1
else:
    linpz['yanzhi'] = 1
# linpz = {'yanzhi': 6, 'nice': 8}
```

dict的get(key,default)方法用于获取字典中key的值，若不存在该key，则默认值default

Python的基础类型Dict

- dict内部存放的顺序和key放入的顺序是没有关系的

- 使用OrderedDict 如果你希望按照key排序

```
from collections import OrderedDict

d = OrderedDict()
```

Python的基础类型Dict

对两组List配对组成Dict
keys = ['name', 'yanzhi', 'nice']
values = ['linpz', 5, 8]

{'name': 'linpz', 'yanzhi': 5, 'nice': 8}

```
keys = ['name', 'yanzhi', 'nice']
values = ['linpz', 5, 8]

# P
result = dict(zip(keys, values))
# {'name': 'linpz', 'yanzhi': 5, 'nice': 8}


# NP:
result = {}
for index, key in enumerate(keys):
    result[key] = values[index]
```

- dict的特点，与list对比

Dict 特点

1、查找和插入的速度快，不会随着key的增加而变慢

2、占用大量的内存，内存浪费多

List 特点

1、查找和插入的时间随着元素的增加而增加，超多的元素就会巨慢

2、浪费的内存很少

Set

Python的基础类型Set

- set和dict非常类似，但是只有key，没有value
- set就是key和value相等情况下的特殊dict
- set中没有重复的key
- ```
  s = set([1, 2, 3, 4])
  print(s)
  ```
  ```

- set添加元素 - add

```
s = set([])
s.add(5)
```

- set删除元素 - remove

```
s = set([1, 2, 3])
s.remove(2)
```

- 两个元素的交集

```
s1 & s2
```

- 两个元素的并

```
s1 | s2
```

- 判断元素是否在集合内

```
ele in set
```

- 对比两份代码的时间效率

```
import time
l = [0 for _ in range(100000)]

start = time.time()
for i in range(5000):
    if i in l:
        pass

print(time.time() - start)
```

```
import time
l = set([0 for _ in range(100000)])

start = time.time()
for i in range(5000):
    if i in l:
        pass

print(time.time() - start)
```

排序Sort

以下三种排序时间复杂度均为O(n^2)，可以通过轻松自学

选择排序 Selection Sort

插入排序 Insertion Sort

冒泡排序 Bubble Sort

演示界面

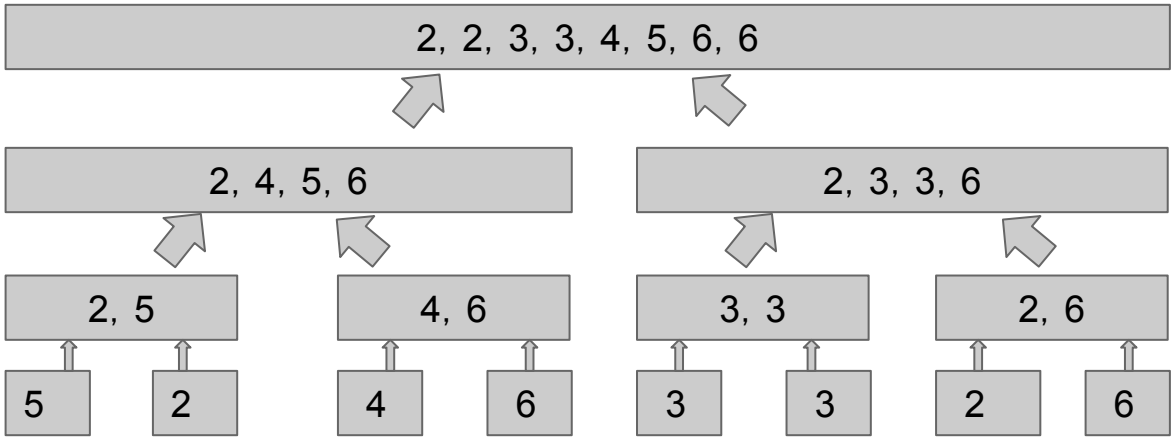
:http://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html

- 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。

归并排序Merge Sort

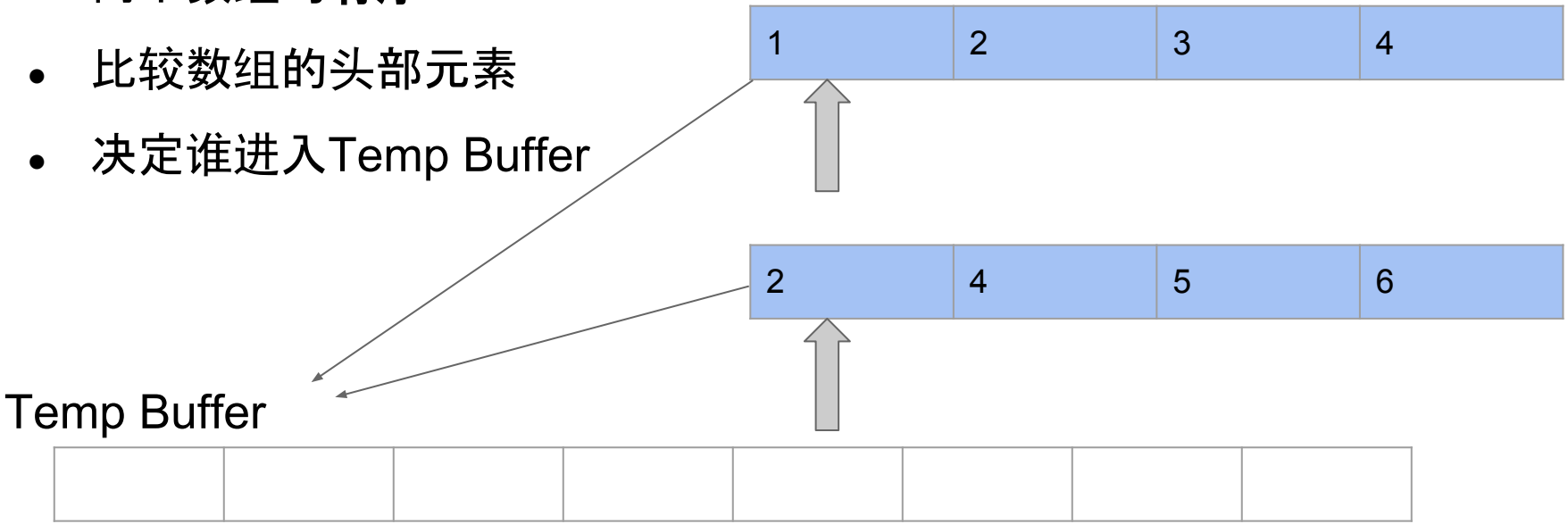
Array = [5, 2, 4, 6, 3, 3, 2, 6] 不断递归，然后向上合并：



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

Python中的排序 - 合并两个有序数组

- 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)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）

- 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

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[random.randint(start, end)] # import random
  - 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/

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？

如数组Array = [3, 4, 4, 5, 6, 7, 4]

Pivot = 3

这会发生什么？

如何确定继续递归的左右两边的边界

Pivot: 4

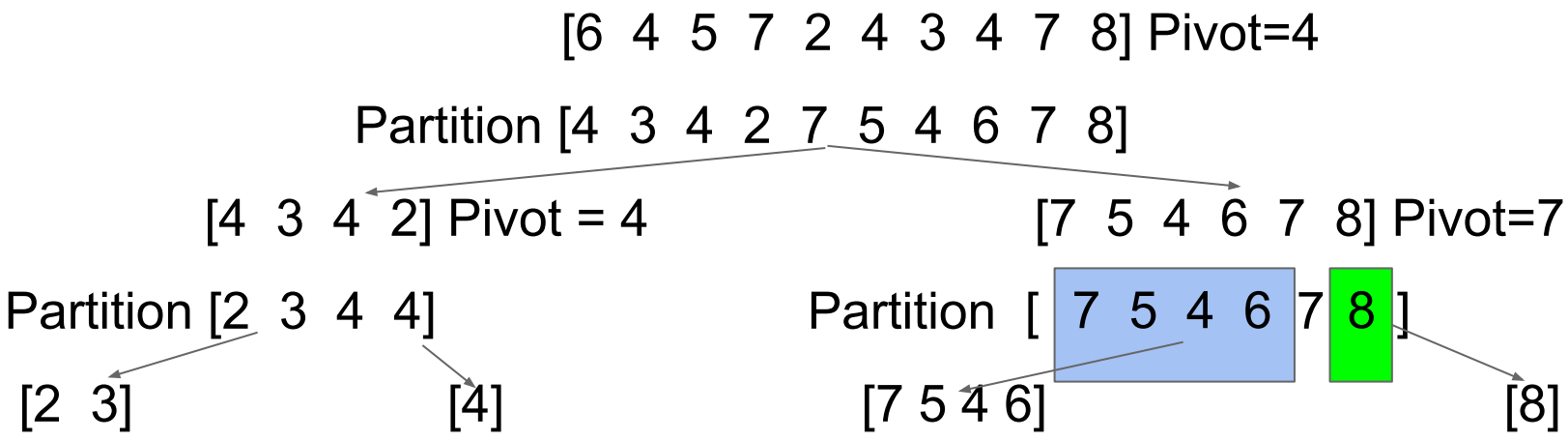
Partition之前

[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】



.....

最后的结果为:[2 3 4 4 4 5 6 7 7 8]

- 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

- 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
  - 利用重写__cmp__函数来覆盖object的比较方法

Python中的排序 - 对List进行排序


```
nums = [1, 2, 5, 4, 3]

nums.sort()

nums_copy = sorted(nums)
```

- 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

Python中的排序 - 对List进行排序

- 参数sort中参数key

```
nums = [(1, 2), (3, 4), (1, 6)]

nums.sort(key=lambda x:x[1])

print(nums)
```

Python中的排序 - 对List进行排序

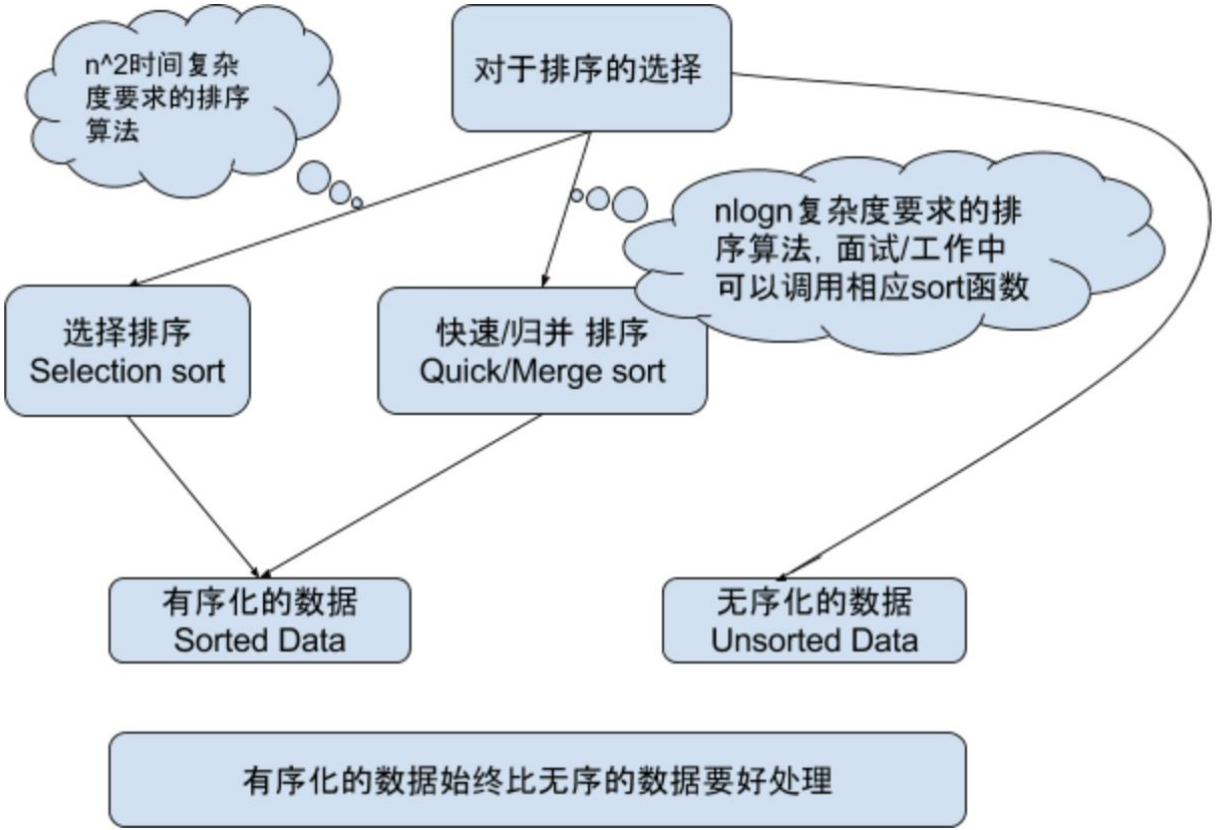

```
from functools import cmp_to_key

def cmp(a, b):
    return a - b

nums = [1, 2, 5, 3, 4]
nums.sort(key=cmp_to_key(cmp))
print(nums)
```

- Python3中sort中的参数不再有cmp
- In Py3, the cmp parameter was removed entirely (as part of a larger effort to simplify and unify the language, eliminating the conflict between rich comparisons and the __cmp__() magic method

Python中的排序的选择



Lambda使用

Python - 匿名函数lambda


```
f = lambda x: 2 ** x
print(f(3))
```

1. 使用lambda的地方都可以使用正常的函数
2. 更像一个表达式
3. lambda函数是否可以有多个参数？

Python - filter函数


```
a = filter(lambda x: x % 2 == 1, [1, 2, 3, 4, 5])
print(a)
print(list(a))
```

filter过滤出满足表达式的元素

Python - map函数


```
b = map(lambda x: 2 ** x, [1, 2, 3, 4, 5])
print(b)
print(list(b))
```

对每一个元素做变换，从x映射成为2 ** x

```
from functools import reduce
reduce(lambda x, y: y ** x, [1, 2, 3, 4])
```

不断的对这个list内的元素进行迭代计算，reduce有范围值

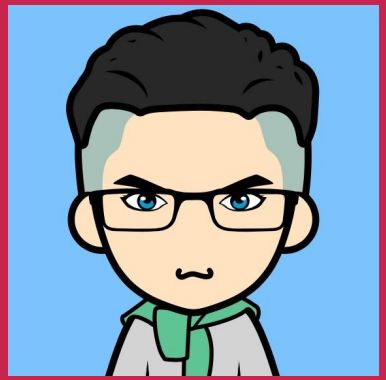
x, y = 1, 2　返回 2 ** 1 = 2

x, y = 2, 3　返回 3 ** 2 = 9

x, y = 9, 4　范围 4 ** 9 = 262144

Python - 使用lambda排序


```
list_c = [('a', 4), ('c', 5), ('b', 3), ('z', 100)]
list_c.sort(key=lambda x: x[1])
```



谢谢大家