Python

一、参考：

基本数据结构：https://blog.csdn.net/qq\_41560183/article/details/82390337

栈：https://blog.csdn.net/wjl31802/article/details/81985275?utm\_source=blogxgwz0

基本数据结构操作：https://www.cnblogs.com/shengzhongqiu/p/7222003.html

零、简单排序

1.现有字典d={‘a’:24，’g’:52，’l’:12，’k’:33}请按字典中的value值进行排序？

sorted(d.items()，key=lambda x:x[1])

2.请按alist中元素的age由大到小排序:

sorted(alist，key=lambda x:x['age']，reverse=True)

3、重新排序数组中的正值和负值

lists.sort(key=lambda x: (x < 0, abs(x)))

一、数组 基本操作：insert、get、delete、size

0、 通过dicts的value取key：

list(a.keys())[list(a.values()).index('a')]

0、 请写出一段 Python 代码实现删除一个 list 里面的重复元素?

l2 = list(set(l1))

l2 = sorted(set(l1)，key=l1.index) 按原来排序

A、B 中相同元素：print(set(A)&set(B))

A、B 中不同元素：print(set(A)^set(B))

1、寻找数组中第二小的元素：初始化2个最小值：firstmin,secondmin。遍历所有元素,假如当前元素小于firstmin, 那么将更新firstmin,secondmin.如果小于secondmin直接更新secondmin

import numpy as np

arr = [2, 3, 4, 1, 7, 6, 5]

print("# arr中最大的数为{}，位于第{}位".format(np.max(arr), np.argmax(arr)+1))

2、找到数组中第一个不重复出现的整数

def findNoDupMany(datas):

single = 0

Map = {}

for data in datas:

if data in Map:

Map[data ] += 1

else:

Map[data] = 1

for key in Map:

if Map[key] == 1:

single = key

break

return single

3、合并两个有序数据

def merge(self, nums1, m, nums2, n):

nums1[m:m+n] = nums2

return nums1.sort()

二、栈 基本操作：push、pop、isEmpty、top 先进后出， x.append() x.pop() 取出

三、队列 基本操作：Enqueue()  在尾部插入元素、Dequeue()  移除头部的元素、isEmpty() 如果为空返回true、Top()  返回的第一个元素,先进先出: x.pop(0)

队列和堆栈：可以用数组来实现（下标取出），也可以用链表实现（通过head、next取出）。

四、链表：链表就像一个节点链，其中每个节点包含着数据和指向后续节点的指针。 链表还包含一个头指针，它指向链表的第一个元素，但当列表为空时，它指向null或无具体内容。一般用于实现文件系统、哈希表和邻接表。

查找单链接倒数第k个数：

class listNode:

def \_\_init\_\_(self,val,next):

self.val = val

self.next = next

class Solution:

def FindKthToTail(self, head, k):

l = []

while head:

l.append(head)

head=head.next

if k>len(l) or k<1:

return None

return l[-k]

五、数：树形结构是一种层级式的数据结构，由顶点（节点）和连接它们的边组成。 树类似于图，但区分树和图的重要特征是树中不存在环路。

1、二叉排序树又称为二叉查找树。它或者是一颗空树，或者是具有下列性质的二叉树：

若它的左子树不为空，则左子树上所有节点的值均小于它的根结构的值；

若它的右子树不为空，则右子树上所有节点的值均大于它的根结构的值；

它的左、右子树也分别为二叉排序树。

class BSTNode:

"""

定义一个二叉树节点类。

以讨论算法为主，忽略了一些诸如对数据类型进行判断的问题。

"""

def \_\_init\_\_(self, data, left=None, right=None):

"""

初始化

:param data: 节点储存的数据

:param left: 节点左子树

:param right: 节点右子树

"""

self.data = data

self.left = left

self.right = right

class BinarySortTree:

"""

基于BSTNode类的二叉排序树。维护一个根节点的指针。

"""

def \_\_init\_\_(self):

self.\_root = None

def is\_empty(self):

return self.\_root is None

def search(self, key):

"""

关键码检索

:param key: 关键码

:return: 查询节点或None

"""

bt = self.\_root

while bt:

entry = bt.data

if key < entry:

bt = bt.left

elif key > entry:

bt = bt.right

else:

return entry

return None

def \_\_iter\_\_(self):

"""

实现二叉树的中序遍历算法,

展示我们创建的二叉排序树.

直接使用python内置的列表作为一个栈。

:return: data

"""

stack = []

node = self.\_root

while node or stack:

while node:

stack.append(node)

node = node.left

node = stack.pop()

yield node.data

node = node.right

def afterTraverse(root):

'''

后序遍历

'''

if root==None:

return

afterTraverse(root.left)

afterTraverse(root.right)

if \_\_name\_\_ == '\_\_main\_\_':

lis = [62, 58, 88, 48, 73, 99, 35, 51, 93, 29, 37, 49, 56, 36, 50]

bs\_tree = BinarySortTree()

for i in range(len(lis)):

bs\_tree.insert(lis[i])

for i in bs\_tree:

print(i, end=" ")

print("\n", bs\_tree.search(4))

0、最长回文字符串：

class Solution(object):

def longestPalindrome(self, s):

"""

:type s: str

:rtype: str

"""

if len(s) <= 1:

return s

for j in range(len(s),0,-1):

for i in range(0,len(s)-length+1):

now\_s=s[i:i+j]

if now\_s==now\_s[::-1]:

return now\_s

1、字符串的所有组合问题，输入三个字符a,b,c，则它们的组合有a,b,c,ab,ac,bc,abc。

2、海量日志数据，提取出某日访问次数最多的那个IP：

思路如下：

1.对于海量数据中的每一个ip，使用hash函数计算hash(ip)%1000,输出到1000个文件中

2.对于这1000个文件，分别找出出现最多的ip。

3.使用外部排序，对找出来的1000个ip在进行排序。

from collections import Counter

if \_\_name\_\_ == '\_\_main\_\_':

ip\_list = ["192.168.1.2","192.168.1.3", "192.168.1.3", "192.168.1.4", "192.168.1.2","192.168.1.2"]

ip\_counter = Counter(ip\_list) # 使用python内置的列表元素计数函数，进行统计

print(ip\_counter.most\_common()) #[('192.168.1.2', 3), ('192.168.1.3', 2), ('192.168.1.4', 1)]

print(ip\_counter.most\_common()[0][0])

3、动态规划：

# 这里使用了图解中的吉他，音箱，电脑，手机做的测试，数据保持一致

w = [0, 1, 4, 3, 1] #n个物体的重量(w[0]无用)

p = [0, 1500, 3000, 2000, 2000] #n个物体的价值(p[0]无用)

n = len(w) - 1 #计算n的个数

m = 4 #背包的载重量

x = [] #装入背包的物体，元素为True时，对应物体被装入(x[0]无用)

v = 0

#optp[i][j]表示在前i个物体中，能够装入载重量为j的背包中的物体的最大价值

optp = [[0 for col in range(m + 1)] for raw in range(n + 1)]

#optp 相当于做了一个n\*m的全零矩阵的赶脚，n行为物件，m列为自背包载重量

def knapsack\_dynamic(w, p, n, m, x):

#计算optp[i][j]

for i in range(1, n + 1): # 物品一件件来

for j in range(1, m + 1): # j为子背包的载重量，寻找能够承载物品的子背包

if (j >= w[i]): # 当物品的重量小于背包能够承受的载重量的时候，才考虑能不能放进去

optp[i][j] = max(optp[i - 1][j], optp[i - 1][j - w[i]] + p[i]) # optp[i - 1][j]是上一个单元的值， optp[i - 1][j - w[i]]为剩余空间的价值

else:

optp[i][j] = optp[i - 1][j]

#递推装入背包的物体,寻找跳变的地方，从最后结果开始逆推

j = m

for i in range(n, 0, -1):

if optp[i][j] > optp[i - 1][j]:

x.append(i)

j = j - w[i]

#返回最大价值，即表格中最后一行最后一列的值

v = optp[n][m]

return v

print '最大值为：' + str(knapsack\_dynamic(w, p, n, m, x))

print '物品的索引：',x

#最大值为：4000

#物品的索引： [4, 3]

4、"驼峰"和"下划线"字符串之间的相互转换

通过对re.sub函数的深入了解，现在应该可以轻松写出"驼峰"和"下划线"字符串相互转换的代码了。直接上代码：

# coding:utf-8

import re

def hump2underline(hunp\_str):

'''

驼峰形式字符串转成下划线形式

'''

# 匹配正则，匹配小写字母和大写字母的分界位置

p = re.compile(r'([a-z]|\d)([A-Z])')

# 这里第二个参数使用了正则分组的后向引用

sub = re.sub(p, r'\1\_\2', hunp\_str).lower()

return sub

def underline2hump(underline\_str):

'''

下划线形式字符串转成驼峰形式

'''

# 这里re.sub()函数第二个替换参数用到了一个匿名回调函数，回调函数的参数x为一个匹配对象，返回值为一个处理后的字符串

sub = re.sub(r'(\_\w)',lambda x:x.group(1)[1].upper(),underline\_str)

return sub

5、查找第一个只出现一次的字符。

def methond3(str):

for s in str:

if str.count(s) == 1:

return s,str.index(s)

6、现在要处理一个大小为 10G 的文件，但是内存只有 4G.

def get\_lines():

l = []

with open('file.txt', 'rb') as f:

data = f.readlines(60000)

l.append(data)

yield l

要考虑到的问题有：内存只有 4G 无法一次性读入 10G 的文件，需要分批读入。分批读入数据要记录每次读入数据的位置。分批每次读入数据的大小，太小就会在读取操作上花费过多时间。

read、readline 和 readlines 的区别?

read:读取整个文件。

readline：读取下一行，使用生成器方法。

readlines：读取整个文件到一个迭代器以供我们遍历。

经过测试发现参数为"rb"时的效率是"r"的6倍。

r： 以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。

rb：以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式。

7、整形数组中除了两个数外,其它的都出现了两次。

class Solution:

# 返回[a,b] 其中ab是出现一次的两个数字

def FindNumsAppearOnce(self, array):

temp = []

for a in array:

if a in temp:

temp.remove(a)

else:

temp.append(a)

return temp

8、全排列组合

def str\_sort(s=''):

if len(s) <= 1:

return [s]

str\_list = []

for i in range(len(s)):

for j in str\_sort(s[0:i] + s[i + 1:]):

str\_list.append(s[i] + j)

return str\_list

9、快排

def quickSort(listx):

if len(listx)<=1:

return listx

pivot = listx[len(listx)//2] #取列表中中间的元素为被比较数pivot

listl = [x for x in listx if x < pivot] #<pivot的放在一个列表

listm = [x for x in listx if x ==pivot] #=pivot的放在一个列表

listr = [x for x in listx if x > pivot] #>pivot的放在一个列表

left = quickSort(listl) #递归进行该函数

right = quickSort(listr) #递归进行该函数

return left + listm + right #整合

print(quickSort([9,3, 6, 8, 9, 19, 1, 5])) #[1, 3, 5, 6, 8, 9, 9, 19]