# 資料結構：作業四

灰字為跟作業三相同的部分

108701009 應數一 許書睿

Python 3.7:

Part A : 定義hash table

(這邊稍微修改了search的部分，原本找不到會報錯，修改後我讓它pass)

https://coderbook.com/@marcus/how-to-create-a-hash-table-from-scratch-in-python/

class HashTable(object):

def \_\_init\_\_(self, length=4):

# Initiate our array with empty values.

self.array = [None] \* length

def hash(self, key):

"""Get the index of our array for a specific string key"""

length = len(self.array)

return hash(key) % length

def add(self, key, value):

"""Add a value to our array by its key"""

index = self.hash(key)

if self.array[index] is not None:

# This index already contain some values.

# This means that this add MIGHT be an update

# to a key that already exist. Instead of just storing

# the value we have to first look if the key exist.

for kvp in self.array[index]:

# If key is found, then update

# its current value to the new value.

if kvp[0] == key:

kvp[1] = value

break

else:

# If no breaks was hit in the for loop, it

# means that no existing key was found,

# so we can simply just add it to the end.

self.array[index].append([key, value])

else:

# This index is empty. We should initiate

# a list and append our key-value-pair to it.

self.array[index] = []

self.array[index].append([key, value])

def get(self, key):

"""Get a value by key"""

index = self.hash(key)

if self.array[index] is None:

#raise KeyError()

pass

else:

# Loop through all key-value-pairs

# and find if our key exist. If it does

# then return its value.

for kvp in self.array[index]:

if kvp[0] == key:

return kvp[1]

# If no return was done during loop,

# it means key didn't exist.

#raise KeyError()

pass

生成隨機數

import numpy as np

def random\_array(k):

return np.random.randint(1,2\*\*k,size=2\*\*k)

以及 時間模塊

import time

Part B : 測試方式

跟作業三差異不大，只是存入的是的key是i ，value也是i的資料，i是符合要求的隨機數

t\_insert = []

t\_search = []

for k in range(10,31):

#insert part

t\_insert\_ =[]

for times in range(10):

insert\_array = random\_array(k)

ht = HashTable(length=2\*\*k)

t = time.perf\_counter()

for i in insert\_array:

#這邊存入key是i value是i資料

ht.add(i,i)

t = time.perf\_counter() - t

t\_insert\_.append(t)

print('\rk=', k,' times:',times+1,' time:',time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()),t\_insert\_,end=' ')

t\_insert.append(np.mean(t\_insert\_))

print('\ninsert: k=',k, t\_insert)

#search part

t\_search\_=[]

for times in range(10):

search\_array = random\_array(17)

t = time.perf\_counter()

for i in search\_array:

ht.get(i)

t = time.perf\_counter() - t

t\_search\_.append(t)

print('\rk=', k,' times:',times+1,' time:',time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()),t\_search\_,end='')

t\_search.append(np.mean(t\_search\_))

print('\nsearch: k=',k, t\_search)

Note:

這邊的 output 長這樣：

…

k= 12 times: 10 time: 2020-06-21 17:30:41 [0.021991199999320088, 0.019774512998992577, 0.021150496000700514, 0.02051666999977897, 0.019767460000366555, 0.019976628000222263, 0.019516948999807937, 0.018582016000436852, 0.022263453000050504, 0.021592855999188032]

insert: k= 12 [0.004908217600132048, 0.009805989599772146, 0.02051322409988643]

k= 12 times: 10 time: 2020-06-21 17:30:48 [0.6767590710005607, 0.6577993680002692, 0.6143646970012924, 0.6445872789990972, 0.6567319780006073, 0.6442201270001533, 0.6737532049992296, 0.6749322819996451, 0.6696228540004086, 0.6553539309988992]

search: k= 12 [0.5356921934004276, 0.7324735537002198, 0.6568124792000163]

…

觀察：如果 time後面的時間距現在時刻超過一個小時，則停止。

這樣做的原因是在如果不幸dead kernel 還能看到跑完的數據

然後令

ht\_insert = t\_insert

ht\_search=t\_search

用於繪圖

Part C : 估計時間

估計方法：假設等比或等差，看（公）差和（公）比，再做估計

1.檢查比（差）:

def checkway(t\_list):

print('比值：')

r=[]

for i in range(len(t\_list)-1):

e = t\_list[i+1]/t\_list[i]

print(e)

r.append(e)

print('平均比值',np.mean(r))

print('差：')

d=[]

for i in range(len(t\_list)-1):

e = t\_list[i+1]-t\_list[i]

print(e)

d.append(e)

print('平均差',np.mean(d))

2.選定方法和平均值作代表，然後套用:

def estim(t\_list,way,r):

if way=='r':

while len(t\_list)<21:

t\_list.append(t\_list[-1]\*r)

print(t\_list)

elif way=='d':

while len(t\_list)<21:

t\_list.append(t\_list[-1]+r)

print(t\_list)

else:

print('error')

Part D: 結果

紅色：binary tree

綠色：skip list

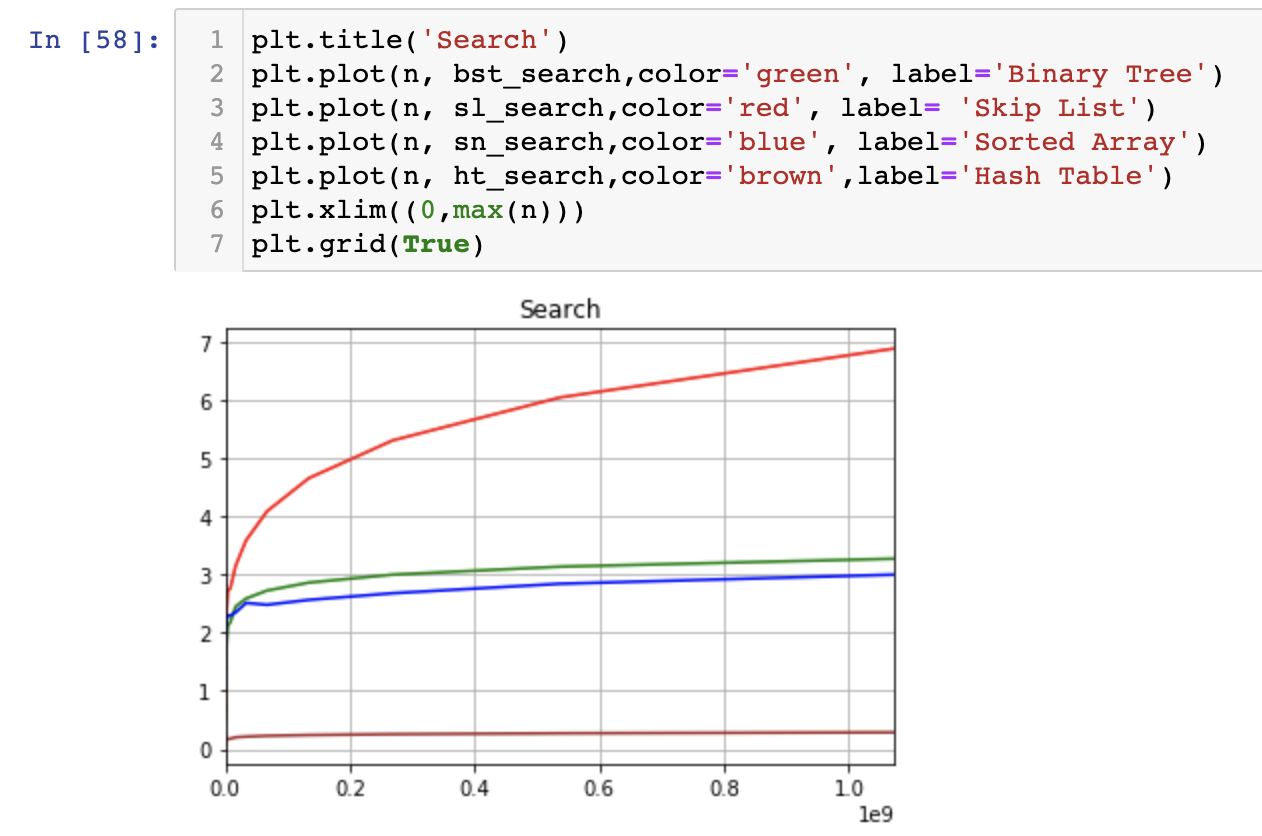
藍色：sorted array

棕色：hash table

1. Insert:



2.Search:



Part E: 其他

作業系統：macOS Ctalina 10.15.3，CPU: i5-8259U，RAM:8GB