# 資料結構：作業二

108701009 應數一 許書睿

Python 3.7:

Part A : 定義function

1. insert\_sort:

<https://www.geeksforgeeks.org/python-program-for-insertion-sort/>

def insertionSort(arr):

# Traverse through 1 to len(arr)

for i in range(1, len(arr)):

key = arr[i]

# Move elements of arr[0..i-1], that are

# greater than key, to one position ahead

# of their current position

j = i-1

while j >=0 and key < arr[j] :

arr[j+1] = arr[j]

j -= 1

arr[j+1] = key

1. merge sort:

https://www.educative.io/edpresso/merge-sort-in-python

def MergeSort(myList):

if len(myList) > 1:

mid = len(myList) // 2

left = myList[:mid]

right = myList[mid:]

# Recursive call on each half

MergeSort(left)

MergeSort(right)

# Two iterators for traversing the two halves

i = 0

j = 0

# Iterator for the main list

k = 0

while i < len(left) and j < len(right):

if left[i] < right[j]:

# The value from the left half has been used

myList[k] = left[i]

# Move the iterator forward

i += 1

else:

myList[k] = right[j]

j += 1

# Move to the next slot

k += 1

# For all the remaining values

while i < len(left):

myList[k] = left[i]

i += 1

k += 1

while j < len(right):

myList[k]=right[j]

j += 1

k += 1

3.randomized quick sort:

<https://www.codementor.io/@garethdwyer/quicksort-tutorial-python-implementation-with-line-by-line-explanation-p9h7jd3r6>

def partition(xs, start, end):

follower = leader = start

while leader < end:

if xs[leader] <= xs[end]:

xs[follower], xs[leader] = xs[leader], xs[follower]

follower += 1

leader += 1

xs[follower], xs[end] = xs[end], xs[follower]

return follower

def \_quicksort(xs, start, end):

if start >= end:

return

p = partition(xs, start, end)

\_quicksort(xs, start, p-1)

\_quicksort(xs, p+1, end)

def quicksort(xs):

\_quicksort(xs, 0, len(xs)-1)

4.counting sort:

<https://en.wikibooks.org/wiki/Algorithm_Implementation/Sorting/Counting_sort>

def counting\_sort(array, maxval):

"""in-place counting sort"""

m = maxval + 1

count = [0] \* m # init with zeros

for a in array:

count[a] += 1 # count occurences

i = 0

for a in range(m): # emit

for c in range(count[a]): # - emit 'count[a]' copies of 'a'

array[i] = a

i += 1

return (array,count)

5.O(nlogn)的sort

這邊選用的是heap sort:

<https://www.programiz.com/dsa/heap-sort>

def heapify(arr, n, i):

# Find largest among root and children

largest = i

l = 2 \* i + 1

r = 2 \* i + 2

if l < n and arr[i] < arr[l]:

largest = l

if r < n and arr[largest] < arr[r]:

largest = r

# If root is not largest, swap with largest and continue heapifying

if largest != i:

arr[i], arr[largest] = arr[largest], arr[i]

heapify(arr, n, largest)

def heapSort(arr):

n = len(arr)

# Build max heap

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

heapify(arr, n, i)

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

# Swap

arr[i], arr[0] = arr[0], arr[i]

生成Input Array （IA)

* Uniformly Randomly : IA1
* Almost Sorted : IA2

(選取不重複位置 參考[http://kailotus.blogspot.com/2018/07/python29numpy.html及https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.choice.html](http://kailotus.blogspot.com/2018/07/python29numpy.html%E5%8F%8Ahttps:/docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.choice.html" \t "_blank) ）

import numpy as np

def IA1(k):

return np.random.randint(1,1001,size=2\*\*k)

def IA2(k):

l = list(range(1,2\*\*k+1,1)) #加上list參考https://blog.csdn.net/wanglin\_lin/article/details/50819657

loc = np.random.choice(2\*\*k,100,replace=False) #選取位置

r\_numbers = np.random.randint(1,1001,100) #生成隨機數

for i in range(100):

l[loc[i]]=r\_numbers[i]

return l

以及 時間模塊

import time

Part B : 測試方式

（以insertion sort 為例）

t\_list =[]

for k in range(10,31):

k\_t\_list =[]

for times in range(10):

input\_array = IA1(k)

t = time.perf\_counter()

insertionSort(input\_array)

t = time.perf\_counter() - t

k\_t\_list.append(t)

print('\rInput Array Size:2^', k,' times:',times,' time:',time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()),end='')

t\_list.append(np.mean(k\_t\_list))

然後再去查看t\_list

Note:

這邊的 output 長這樣：

Input Array Size:2^ 17 times: 9 time: 2020-04-29 19:58:16

觀察：如果 time後面的時間距現在時刻超過一個小時，則停止。

然後令

Insert\_sort\_t = t\_list

用於繪圖

Part C : 估計時間

發現時間大致上呈等比

1.判斷公比:

esti=[]

for i in range(len(insert\_sort\_t1)-1):

e=insert\_sort\_t1[i+1]/insert\_sort\_t1[i]

print(e)

esti.append(e)

print('\n',np.mean(esti))

output:

3.90033292353372

3.87381117437975

3.9643606681168997

4.02509465316946

4.217586183564448

4.005217751489279

4.3351319658340595

4.045933617155374

2.添加到list去，用於繪圖:

### 每一項大約是前一項的4.04倍時間，由此估計

while len(insert\_sort\_t1) <21:

insert\_sort\_t1.append(insert\_sort\_t1[-1]\*4.04)

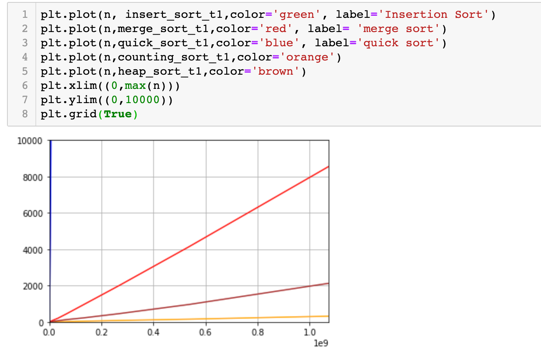
print(insert\_sort\_t1)

output:

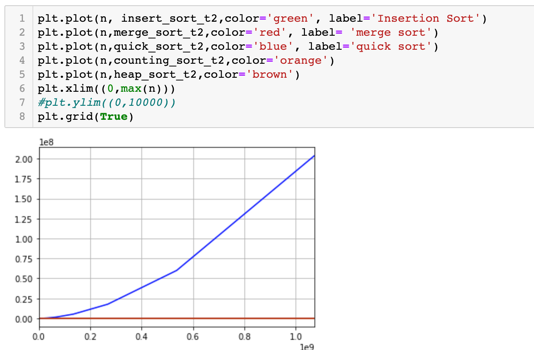
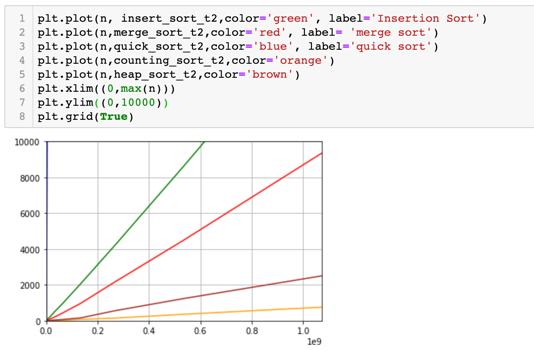
[0.11104481229995145, 0.4331117374011228, 1.6777930880994973, 6.651376927700039, 26.77242170790014, 112.91499589580053, 452.2491459711993, 1960.5597292209, 7920.661306052436, 31999.47167645184, 129277.86557286544, 522282.5769143764, 2110021.610734081, 8524487.307365688, 34438928.721757375, 139133272.0358998, 562098419.0250351, 2270877612.861142, 9174345555.959015, 37064356046.07442, 149739998426.14066]

Part D: 結果

1. Uniformly Randomly:



1. Almost Sorted:



Part E: 其他

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

Code在：<https://nbviewer.jupyter.org/github/ivan1003hsu/python_work/tree/master/for_DS/>

因多次dead kernel 所以分成多個檔案，並對程式碼做了些微更改：

t\_list =[]

for k in range(10,31):

k\_t\_list =[]

……

print('\rInput Array Size:2^', k,' times:',times,' time:',time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()),t\_list,k\_t\_list,end='')

……

確保變數的結果在dead kernal後還能看到