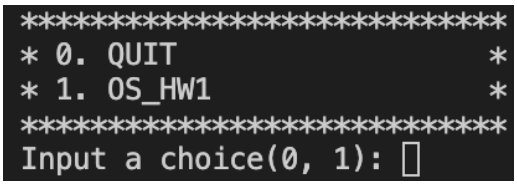
作業系統作業一　* 有做流程圖

資訊三乙 10833230 李郁含

1.開發環境：MacBook air 2020 M1 / Visual Studio Code /　使用語言：Python

2.實作方法和流程: 開始執行會出現以下畫面



輸入 1 能不斷重複執行流程

直到輸入 0 退出

便開始整個作業的輸入流程，首先輸入檔案名稱，接著輸入要切成幾份，最後輸入方法編號，

我整個 bubblesort、mergesort、方法 function 皆是包在一個名為 Do 的 class，因此以上

使用者輸入的檔案名稱、k 份數量和方法編號皆會儲存在 Do 這個 class 裡，方便 function

取用。

以使用者輸入的 method 來決定要執行的 function：

Method1: 是將 N 個數目字直接進行 BubbleSort,並顯示 CPU 執行之時間

我首先一行一行先將檔案讀進列表裡，接著宣告一個最大容量為 list 長度的 queue(先進先出)，然後將前面讀完檔的 list 跟 queue 丟進 BubbleSort function 作 BubbleSort，並在執行的過程中也同樣紀錄 CPU time，最後將回傳回來的 queue 與 time 寫檔。

Method2: 是將 N 個數目字切成 k 份,在一個 process 內對 K 份資料進行 BubbleSort 之後,再用同一個 process 作 MergeSort,並顯示 CPU 執行之時間

首先一行一行將檔案讀進列表並切成 k 份，然後宣告一個共享的 queue.Queue 對象並返回它的代理，讓這個管理器可以在不同進程中共享數據，然後宣告一個工作內容是 bubble_merge 的 process，代表在同一個 process 進行 BubbleSort 和 MergeSort，工作同時進行的過程也同樣紀錄 CPU time，最後將回傳回來的 queue 與 time 寫檔。

Method3: 是將 N 個數目字切成 k 份,並由 K 個 processes 分別進行 BubbleSort 之後,再用

process(es)作 MergeSort,並顯示 CPU 執行之時間
同樣地，首先一行一行將檔案讀進列表並切成 k 份，然後宣告一個共享的 queue.Queue 對象並返回它的代理，讓這個管理器可以在不同進程中共享數據，jobs:然後宣告一個以 queue 為資料結構 （上限為 k）的工作容器裝作 BubbleSort 的 process，後面用一個迴圈建立 k 個 processes。
mjobs:和一個以 queue 為資料結構（上限為 k-1）的工作容器裝作 MergeSort 的 process。
一個迴圈邊一個執行 jobs 邊建立 k-1 個 mjobs 並同時執行判斷直到所有工作都 start 才退出迴圈，接著兩個迴圈以判斷所有工作皆完成，工作同時進行的過程也同樣紀錄 CPU time，最後將回傳回來的 queue 與 time 寫檔。

Method4: 將 N 個數目字切成 k 份,並由 K 個 threads 分別進行 BubbleSort 之後,再用 thread(s) 作 MergeSort,並顯示 CPU 執行之時間
同樣地，首先一行一行將檔案讀進列表並切成 k 份，接著宣告一個最大容量為 list 長度的 queue(先進先出)， * thread 不用設定要共用 q，他們本來就會共用 *
*後面基本上與 method3 差不多 *
jobs:然後宣告一個以 queue 為資料結構 （上限為 k)的工作容器裝作 BubbleSort 的 thread，後面用一個迴圈建立 k 個 threads。
mjobs:和一個以 queue 為資料結構（上限為 k-1）的工作容器裝作 MergeSort 的 thread。
一個迴圈邊一個執行 jobs 邊建立 k-1 個 mjobs 並同時執行判斷直到所有工作都 start 才退出迴圈，接著兩個迴圈以判斷所有工作皆完成，工作同時進行的過程也同樣紀錄 CPU time，最後將回傳回來的 queue 與 time 寫檔。

將檔案讀進列表並切成 k 份：先（step）向下取整除法 （切完）每份小 list 的數量 = 大 list 的數量/要切幾份，以 step 數量為一組切成 k 或 k+1 個（因為不一定會剛好切整數），判斷如果是 k+1 個就把 k+1 那個 list 裡面的所有數字增加進第 k 個 list，並刪掉最後一個（多餘的那個 k+1）

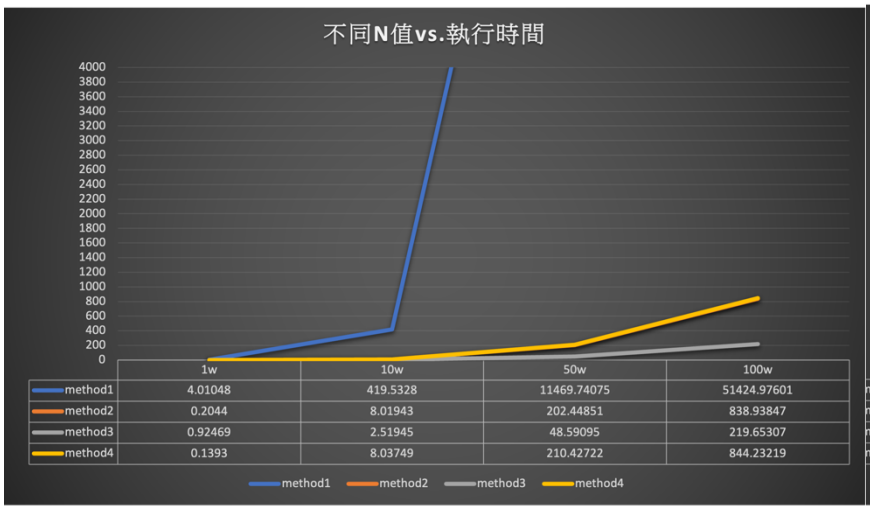
寫檔的部分：用前面計算的 CPU time 以 time 函式庫寫檔
以 datetime 函式庫取得程式執行完寫檔當下的時間(時區為 UTC+8)

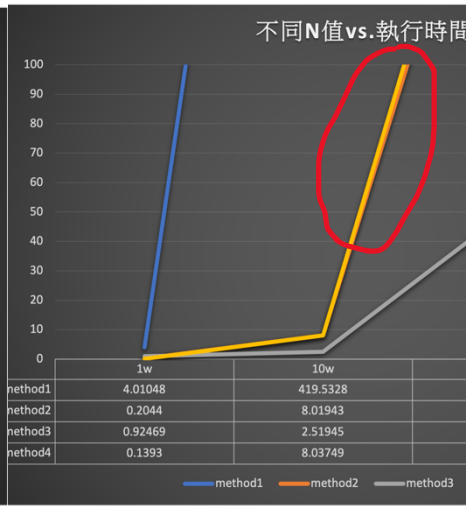
3.特殊機制考量與設計: 一開始按執行會出現一個使用者介面（上面有圖片），輸入 1 會開始這個作業的流程直到使用者輸入 0 才會退出，可以讓使用者不斷重複作業的流程，不用重新開啟。

4.分析結果和原因:

不同 N 值 vs.執行時間(都是 K=50)

	1w	10w	50w	100w
method1	4.01048(s)	419.53280 (s)	11469.74075(s)	51424.97601 (s)
method2	0.20440 (s)	8.01943 (s)	202.44851 (s)	838.93847 (s)
method3	0.92469 (s)	2.51945(s)	48.59095 (s)	219.65307(s)
method4	0.13930 (s)	8.03749(s)	210.42722 (s)	844.23219(s)



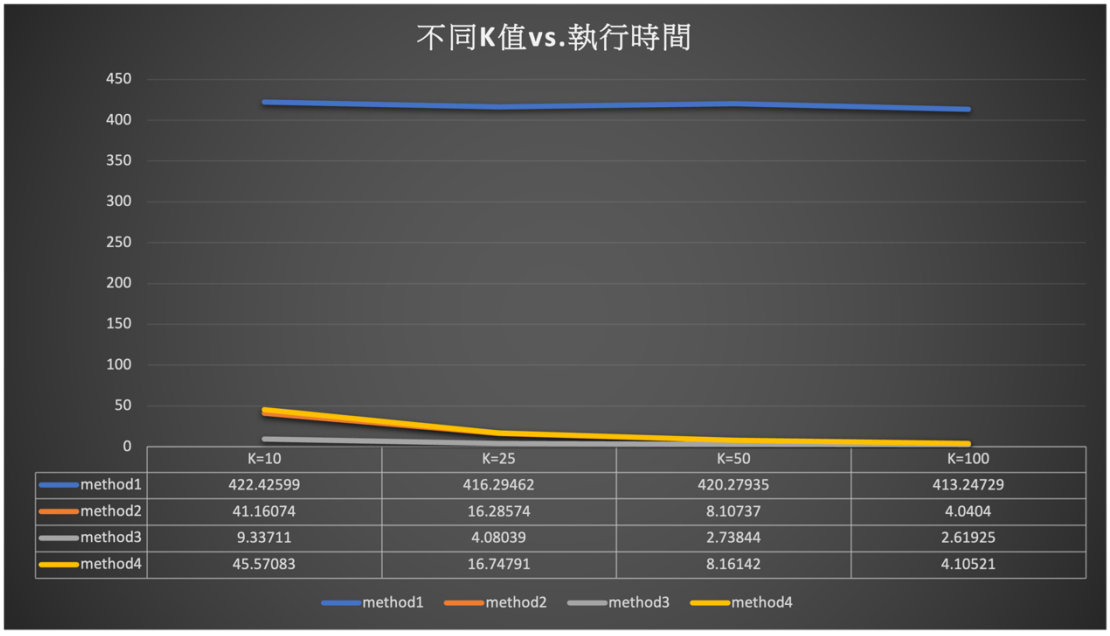


*** 怕老師助教們找不到 method2 特別圈起來！我的 method2、4 數據有點像**

因此如果不拉進一點會看不太到！！

不同 K 值 vs.執行時間 (都是 10w 筆)

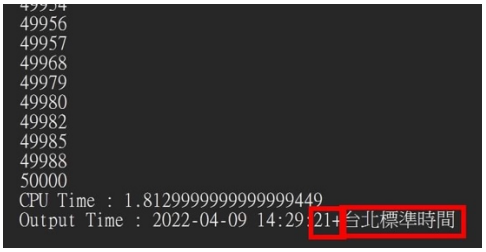
	K=10	K=25	K=50	K=100
method1	422.42599(s)	416.29462(s)	420.27935(s)	413.24729(s)
method2	41.16074(s)	16.28574(s)	8.10737(s)	4.04040(s)
method3	9.33711(s)	4.08039(s)	2.73844(s)	2.61925(s)
method4	45.57083(s)	16.74791(s)	8.16142(s)	4.10521(s)



method1 普遍花的時間會比較多（一百萬的數據甚至花了我快一天跑程式）因為是單純的 bubblesort 沒有切份也沒有 process、thread 做同步，而 method2 和 method4 一個是同一個 process 非同步作 bubblesort 跟 mergesort；一個是 k 個 threads 同步作 bubblesort 跟 mergesort，速度都相對快很多，最後 method3 是 k 個 processes 同步作 bubblesort 跟 mergesort 速度又更快了，我認為是和資源吃不吃得到同一份的關係（共用資源）。

5.撰寫程式時遇到的 bug(請截圖)及相關的解決方法:

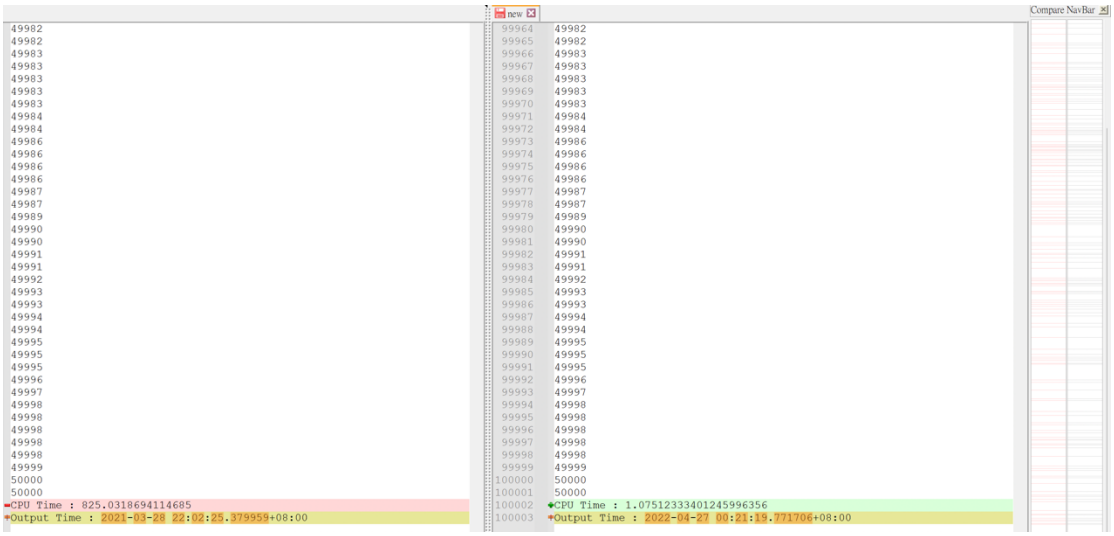
一開始不太清楚老師的說明，後來問了助教才確認各種方法的是否要切與排序的分類，開始寫後立馬遇到了問題，方法一是直接 bubblesort 比較沒有什麼問題，只有最後計算 CPU 的部分和紀錄當前時間與時區偏移量有點困難找不到方法能顯示時區偏移量（有找到但需要另外複雜的事），



後來便在 python 找到能更好紀錄當前時間與時區偏移量的方法（也有聽說用 c++寫 thread 會比較複雜），因此臨時改成用 python 寫。

（平常都習慣寫 c++其實還完全沒碰過 python 因此邊寫作業邊花了大量時間學習並研究 python，覺得特別棒！因為這份作業學到了 process 和 thread 的應用也學習了一種新的語言，因為都是程式語言基本上學習的方法架構程式用法其實都差不多，因此對於學習更多程式語言也更有信心了！！）

在要將一個 list 切成 k 份時因為不一定會剛好切成整數，因此出現了數量不符且缺漏非常多數目的結果，如以下與老師 output 比對結果



後來加上了如果有多餘的（k+1 個）條件，把 k+1 那個 list 裡面的所有數字增加進第 k 個 list，並將最後一個刪掉，便解決了這個問題。