作業系統-HW1

資工三乙 11027222 黃彥霖

開發環境: Windows11, Visual Studio Code

語言: python 3.10.1

任務簡介:總共分為四種任務

任務 1:運用 bubble sort 將一筆資料用單一 process 完成，並輸出時間和結果。

實作方法:用一個 bubble sort 一次處理完全部資料。

任務 2:先將一筆資料分成 k 筆，分別對他們做完 bubble sort，對 k 筆資料做完排序後，用 merge sort 將全部資料合併在一起，最後合併成一個完整的且已排序的資料。全部的 sort 都是在一個 process 下依序輪流做完再換下一個。

實作方法: 用 bubble sort 一次 k 份資料，會得到 k 分已經排序好的資料，再來用迴圈判斷 list 中的 list 的數量是否為 1，因為合併到最後會只剩下一個 list。每兩個 list 用 merge sort 處理最後會合併成一個 list，合併完後再加入原來的 list，會在選兩個 list 去做 merge sort(依照被加進來的順序)，直到 list 裡面剩下唯一一個 list 而已。

任務 3: 先將一筆資料分成 k 筆，先用 k 個 processes 做 bubble sort，再用 k-1 個 processes 做 merge sort。

實作方法:這次任務會用到 multi-processes 的寫法，先把資料分成 k 份，k 個資料就創建 k 個 processes，把它們分別交給不同的 processes 去執行，這些 processes 是可以同時間去執行的，最後將每個 process 的結果放在一個 list 中，收集所有 k 個 process 完成的 bubble sort 結果。再來用 k-1 個 merge sort 把全部 bubble sort 的結果合成一份完整的排序結果，我使用的方法是每兩個 list 都丟入 merge sort，兩兩一組，有幾組就創建幾個 processes 去做執行，有剩下一個的資料(基數筆資料)沒辦法分組的話，就等到下一輪再做合併，會去蒐集 merge sort 的結果並重新丟入 list 裡面，直到 list 只剩下一筆資料，就代表全部都合併完成。

創建 process 的方法使用 ProcessPoolExecutor，允許在一個進程池中並行，執行多個任務，也可以通過共享資料來方便地在這些進程之間傳遞訊息。

用 as_completed()來收取各 process 做完的資訊，用.result()來接收做完的結果，bubble sort 和 merge sort 使用相同的方法做 multi-processes，且我會先切好資料份數在丟入不同 process。

任務 4: 先將一筆資料分成 k 筆，先用 k 個 threads 做 bubble sort，再用 k-1 個 threads 做 merge sort。

實作方法: 這次任務會用到 multi-threads 的寫法，先把資料分成 k 份，k 個資料就創建 k 個 threads，再輪流呼叫 threads 讓他跑，可以用 Semaphore()去控制 threads 的數量，有小於 k 數量的 threads 就可以去創建新的 thread，都創建好之後可以用 put()去接收 thread 跑完的結果，用 join()去等所有的 threads 結束再把所有做好 bubble sort 的 list 回傳。蒐集到 k 個 threads 做完 bubble sort 的結果。再來用 k-1 個 merge sort 把全部 bubble sort 的結果合成一份完整的排序結果，方法是每兩個 list 都丟入 merge sort，兩兩一組，有幾組就創建幾個 threads 去做執行，接收 thread 的方法和上面一樣，有剩下一個的資料(基數筆資料)沒辦法分組的話，就等到下一輪再做合併，會去蒐集 merge sort 的結果並重新丟入 list 裡面，直到 list 只剩下一筆資料，就代表全部都合併完成。

用 Thread()來創建不同的 thread 且.start()來呼叫使 thread 開始執行，用.join()來收取 thread 是否做完的資訊，做完的資料會存在 queue 裡面，再去收取每個 thread 的結果。

其他加快運算的方法:

改善 bubble sort:
用 swap 來判斷是否這一輪資料有沒有交換，有的話設成 true，沒有交換會用預設的 false，且可以知道整份資料沒有需要交換的部分，可以提前確定資料排完，不用檢查全部的資料，在結束 bubble sort。

@jit 的使用
一個 Numba 資料庫中的修飾符，可以大幅提升 python 的效率，不使用 python Interpreter 的特性，而是把它轉換成機器碼，不用每次遇到成是在做翻譯，可以提高效率，我是放在 bubble sort 和 merge sort 前面，提升主要兩個需要效率的程式。

實驗數據:

可由以下一張圖表判斷相同數量不同 k 值，或相同 k 值不同數量的結果。方法一不考慮 k 值的影響。

K=切成 K 份資料，t 為花費時間，N 為資料量

K = {1, 3, 8, 17}	N = 1 萬	N = 10 萬	N = 50 萬	N = 100 萬
方法 1	K=1, t=0.360	K=1, t=22.335	K=1, t=608.537	K=1, t=2815.392
方法 2	K=1, t=0.318 K=3, t=0.692 K=8, t=0.750 K=17, t=2.438	K=1, t=9.632 K=3, t=3.605 K=8, t=1.784 K=17, t=1.546	K=1, t=242.333 K=3, t=78.225 K=8, t=29.605 K=17, t=14.960	K=1, t=945.461 K=3, t=310.196 K=8, t=123.242 K=17, t=58.132
方法 3	K=1, t=0.781 K=3, t=1.938 K=8, t=3.220 K=17, t=9.405	K=1, t=10.224 K=3, t=3.657 K=8, t=3.641 K=17, t=6.866	K=1, t=251.804 K=3, t=33.098 K=8, t=9.995 K=17, t=10.835	K=1, t=981.565 K=3, t=138.752 K=8, t=30.202 K=17, t=58.132
方法 4	K=1, t=0.301 K=3, t=0.655 K=8, t=0.702 K=17, t=0.9139	K=1, t=9.608 K=3, t=3.491 K=8, t=1.733 K=17, t=1.478	K=1, t=249.189 K=3, t=83.577 K=8, t=30.026 K=17, t=16.558	K=1, t=973.067 K=3, t=339.085 K=8, t=127.772 K=17, t=61.967

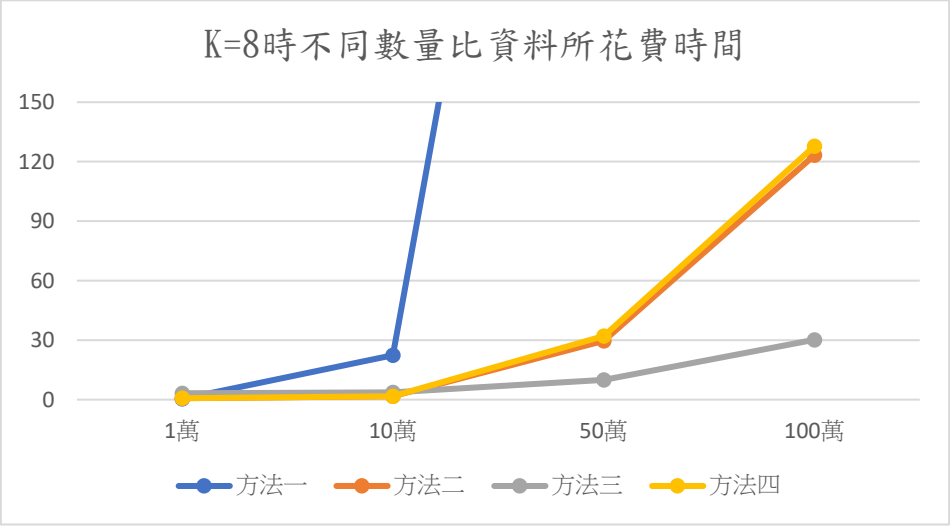

表 1-1:實驗記錄表格（t 的單位:s）

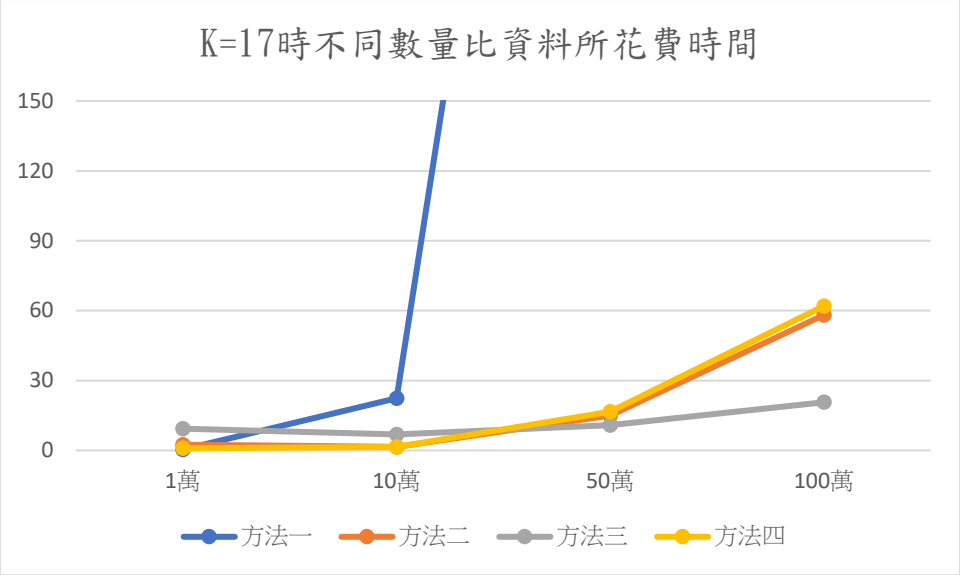

表 1-2:實驗記錄表格（t 的單位:s）

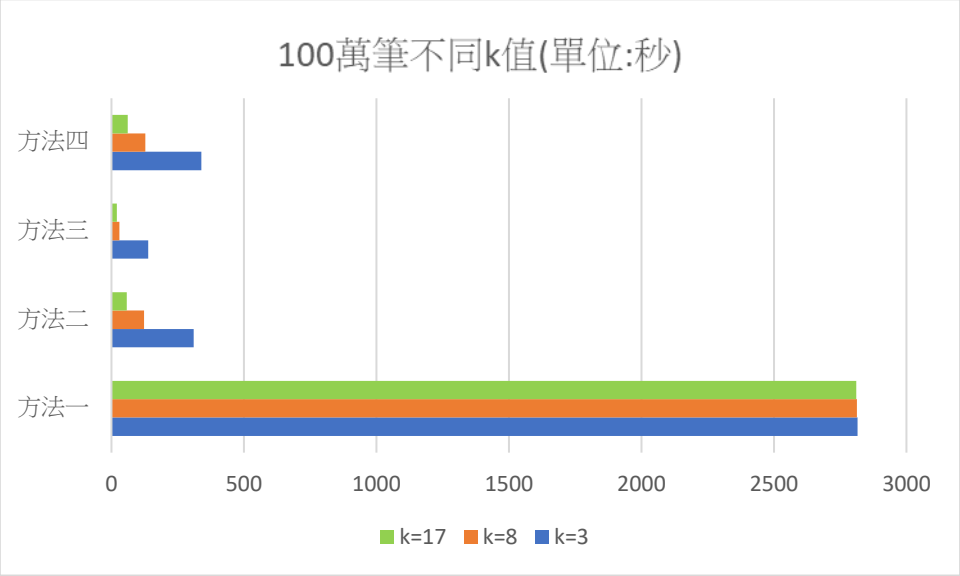

表 2:實驗記錄表格（t 的單位:s）

實驗結果分析:

任務一:
此任務不受 k 值影響故不討論。

不同 N 值與執行時間:bubble sort 為 O(2^n)之演算法，故花費時間會隨著資料數量提升而指數增加。

任務二:
不同 K 值與執行時間:隨著 k 數量的上升，排序完成的時間也會大幅提高，因為 merge sort 是 O(n*logn)之演算法，相比 bubble sort 有更快的效率，切越多份代表 bubble sort 要做的排序資料量會越小，整體速度也會隨之提升。如果資

料本身數量不大，且切成很多份的話，花費時間也會隨之上升，因為兩個 sort 速度不同，切太多份變成 merge sort 要做很久，反而降低效率。

不同 N 值與執行時間:此任務還是受到資料數量的影響，資料量越大所以花費的時間會比較多。

任務三:
不同 K 值與執行時間:由表 1-1 和表 1-2 的比較可以發現，在資料量非常巨大的條件時，k 值越大可以加快總共所花費的時間，且不同的 processes 是同步進行的，multi-process 可以大幅度降低處理資料所花費的時間。但可以發現兩張圖在資料量一萬時，k 值越大反而降低了執行效率，代表資料量小時，去創建大量的 process 不會有顯著的效能提升，合理的解釋方式是因為每次要衍生和刪除 process 要付出比較多的資源，會需要新的地址，且程式會被複製。造成 process 的資源會非常肥大，導致效能下降。

不同 N 值與執行時間:N 值提升，但不會像其他任務一樣花費時間指數成長，面對大筆數目的資料也能在合理的等待時間做完。

任務四:
不同 K 值與執行時間:由 1-1 和 1-2 可發現，實驗結果發現和任務二沒有相差多少，k 值上升會減少花費的時間。雖然說 thread 和 process 相比下來不用付出創造和刪除的成本，但從時間結果可以得知，是依序做完每一個 thread，但是不同的 thread 因為每次都是創造出新的一個 thread。至於沒有做到平行是 python 有 GIL 的限制，是 python Interpreter 的機制，只允許同一時間執行一個執行緒。

不同 N 值與執行時間:N 值提升，花費時間也會明顯提升，因為 GIL 的關係，就算用到多執行緒，這些執行緒都被序列化執行，沒有用到 multi-thread。