1. 開發環境:Windows11 anaconda python 3.8(需安裝程式裡面含有的 package 才能正常的執行)
2. 實作方法和流程、包含 3.特殊機制考量與設計:

有設計一個讀檔和寫檔的 class

在任務一和二有設計一個進度條，因為任務一的方式真的太慢了，需要知道還有多久，還有必須等待進度條跑完再去輸入下一筆 input 輸入檔名資料，不然會導致進度條出錯。

程式正在執行請耐心等候，尤其是任務一、二、四

(1)第一題對於讀檔好的資料(List)，就直接使用兩層 while 迴圈進行 bubble sort

(2)第二題先把讀檔好的資料(List)，利用在一個 process 內對於切好的 k 份資料，利用迭代的方式做完 k 次 bubble sort 之後，再做 k-1 次的 merge

(3)第三題先把讀檔好的資料(List)，我是利用 multiprocessing package 裡面 pool 的功能，用非同步的方式對於切好 k 分的資料進行 bubble sort 之後，能很好的運用電腦 CPU 的效能，假如 k 值大於或等於 CPU 的核心數量，bubble sort 的速度會很快，而 merge 是用 starmap 的 funciton，可能沒辦法好好利用 CPU 多核心的效能，因為在這邊我的寫法是兩個 sort 好的資料，merge 到一個 buffer 裡面，每個 sort 好的資料都必須等前面的 merge 結束才能繼續 merge，而且 merge 對於資料量

大的時候和 k 值大的時候速度相對會慢許多，不過也是 4 種方法裡在 k 值為 100 最快的

(4)第四題先把讀檔好的資料(List)，我自己引用了 threading 的 package 寫一個 Return value of Thread 的物件 Return value of Thread 的物件取得 thread 裡面執行完 return 的資料，一開始是先建立 k 個 Return value of Thread 的物件，裡面就包含了 thread 物件，好處是可以交由這 k 個 thread 物件進行 bubble sort，再交由這 k 個 thread 物件進行 merge sort，這個方法在 k 值很大的時候有不錯的效果，能產生很多 thread 幫忙 bubble sort，也因我的寫法 bubble sort 的交於 k 個 thread 執行沒有資料共用的問題，但 merge 卻需要等前面的 thread 用完才行，所以相對於 multi thread，我的寫法沒有能好好改善利用多的 thread 去幫忙 merge，因為需要上一個 thread merge 完的資料，為了 thread 之間的互斥性，所以我的方法是讓後面的 thread 等前面的 thread。

3. 分析結果和原因:

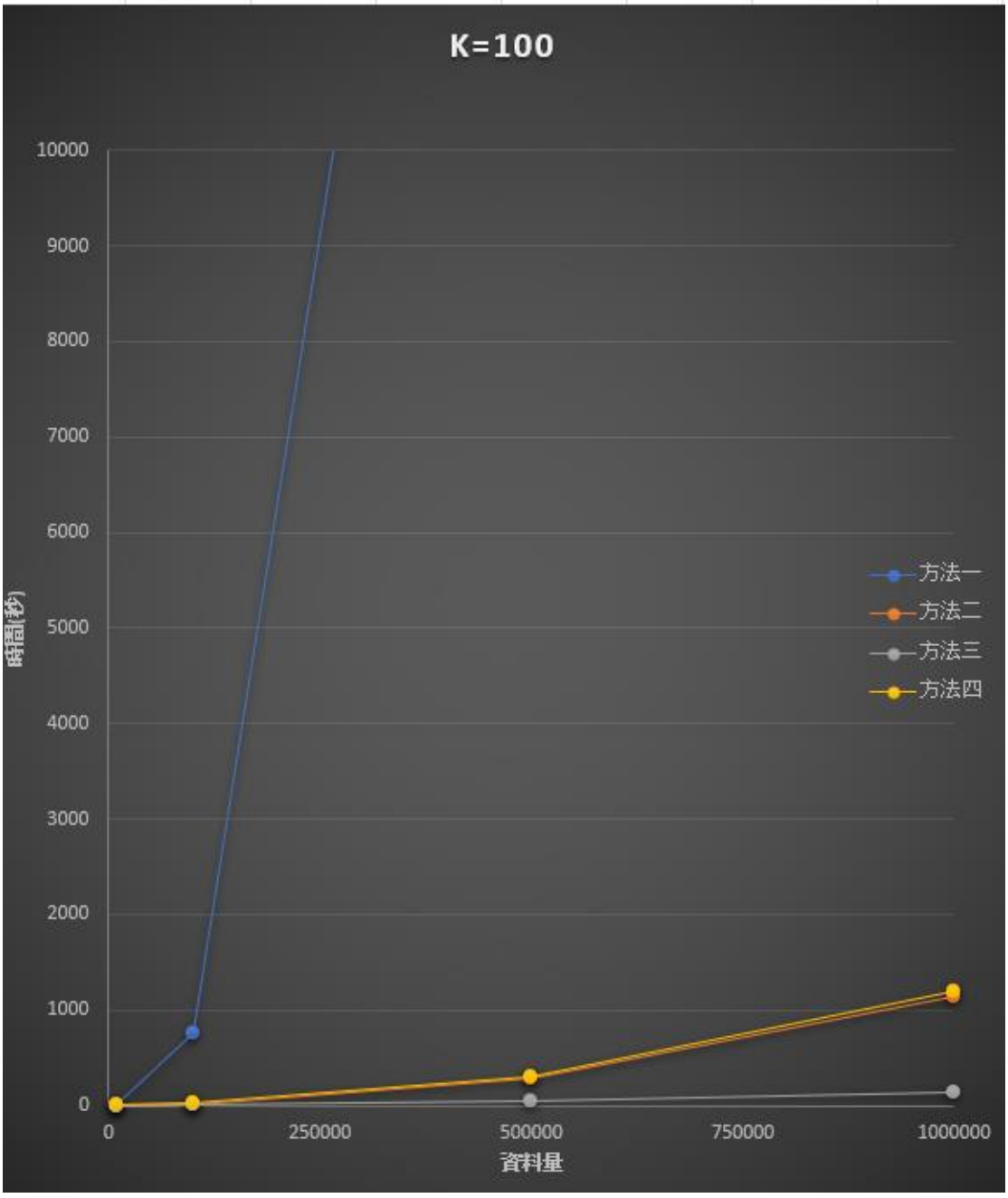
[不同 N 值 vs. 執行時間]比較四種方法在資料筆數 1 萬、10 萬、50 萬、100 萬所耗費的時間，(共 16 次實驗，如下表格)，執行速度的 差別，使用圖表分析。

4. 任務二三四 k 值分析

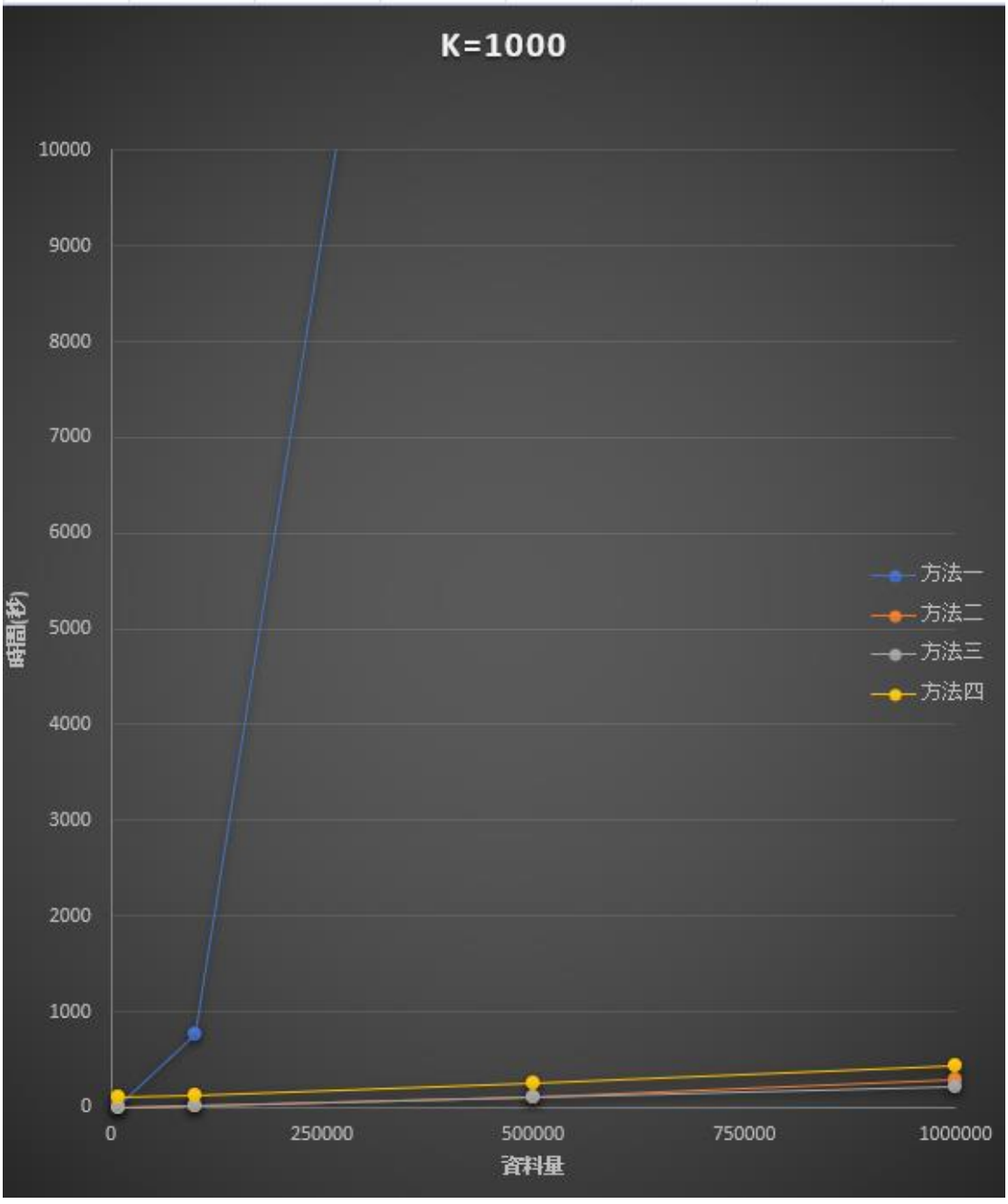
K 的大小必須要找到一個平衡點，不能大也不能小，k 值太大在我的方法裡會使 merge 的時間拉長，但 bubble sort 的時間變短，太大或太小會有一好沒兩好的情況，在二三四的任務中都有一個很特別的問題，就是 k 太大的時候，速度會變得很慢很慢，原因是 k 越大，merge 的次數變多，k 值為 100 的時候，速度卡在 bubble sort 上面，k 值為 10000 的時候，速度卡在

merge 上面，在 k 值為 1000 的時候可以看到 bubble sort 和 merge 取得良好的平衡。

k=100	10000	100000	500000	1000000
方法一	11.3819	755.1132	23006.11	97967.88
方法二	0.250056	12.34729	290.266	1148.984
方法三	1.607361	4.36498	39.57032	133.1856
方法四	11.39356	23.57431	306.822	1189.481



k=1000	10000	100000	500000	1000000
方法一	11.3819	755.1132	23006.11	97967.88
方法二	1.362306	16.21816	112.4026	291.9785
方法三	3.674826	21.82544	114.8853	225.3246
方法四	112.1364	128.27	247.3873	440.795



k=10000	10000	100000	500000	1000000
方法一	11.3819	755.1132	23006.11	97967.88
方法二	7.045629	137.6635	829.7373	1812.961
方法三	14.79069	179.9276	984.9443	2010.647
方法四	1095.808	1261.416	2057.271	3132.043