OS HW1 10820109 陳詩云

一、開發環境 ：

1. Windows11
2. IDE : dev c++
3. Cygwin compiler (模擬 linux 環境)

(檔案路徑不能有中文)

二、實作方法和流程&四種方法比較 ：

(a) **Case1 (Bubble Sort):**

從第一筆資料開始，逐一比較相鄰的兩筆資料，如果兩筆資料大小順序有誤則做交換，反之則不動，直到所有元素都比過為止。

(b) **Case2 (Bubble Sort+Merge Sort):**

將 input 切成使用者輸入的 k 份，先對每份數據做 bubble sort，再用 merge sort 的 merge 合併資料。

(c) **Case 3 (MultiProcess):**

將 input 切成使用者輸入的 k 份，並 fork 出 k 個 process 分別對資料做 bubble sort，最後將排序後的資料兩兩合併，直到剩一筆資料。

(d) Case 4 (MultiThreads):

將 input 切成使用者輸入的 k 份，並建立 k 個 thread 分別對資料做 bubble sort，排序後的資料再用 thread 兩兩合併，直到剩一筆資料。

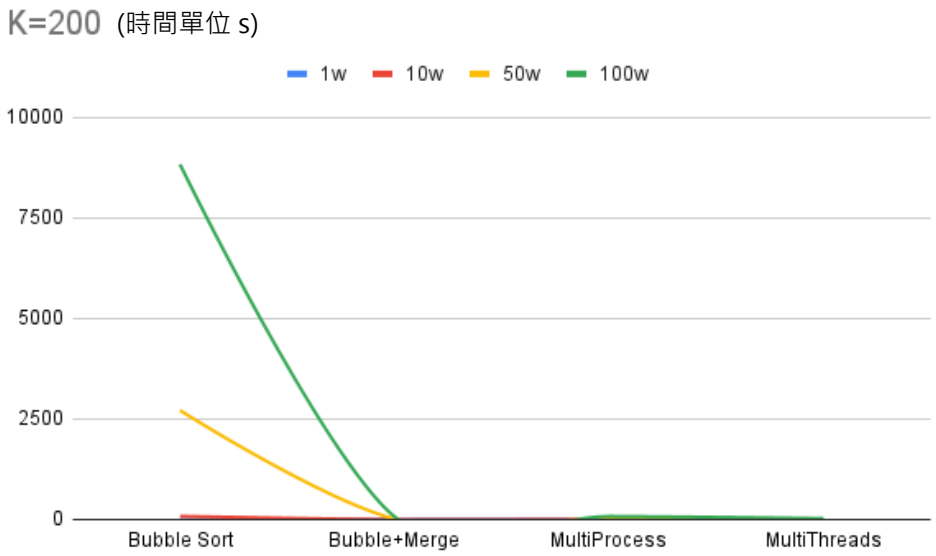
三、特殊機制考量與設計

因為要將資料分成 k 等份，所以用二維 vector 儲存資料。

四、分析結果和原因

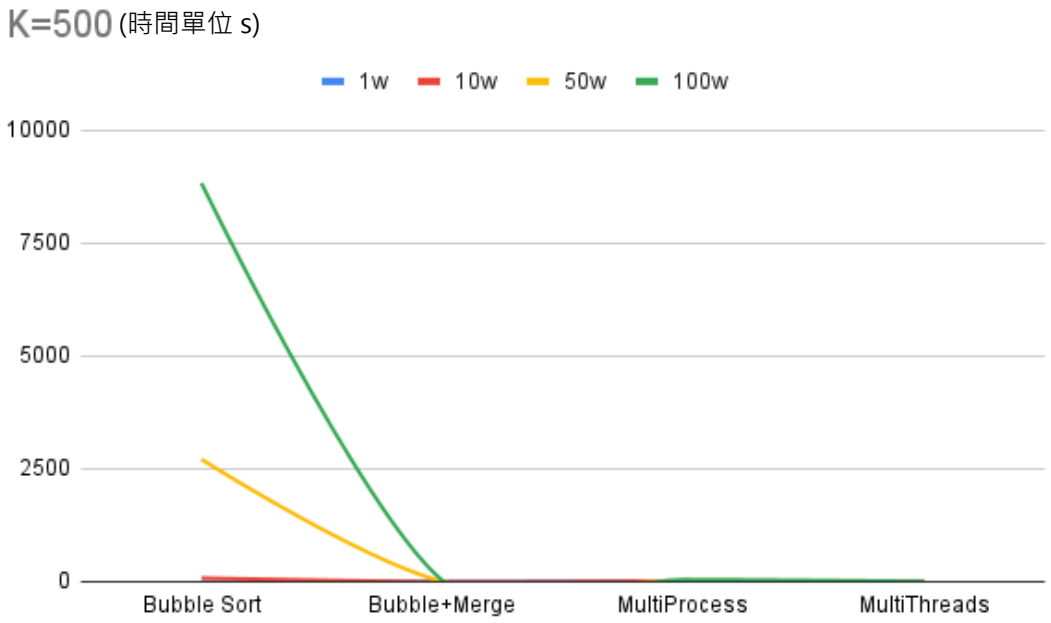
1. K=200

	1w	10w	50w	100w
Bubble Sort	0s	81s	2713s	8829s
Bubble+Merge	0s	0s	17s	72s
MultiProcess	6s	7s	23s	78s
MultiThreads	0s	0s	6s	27s



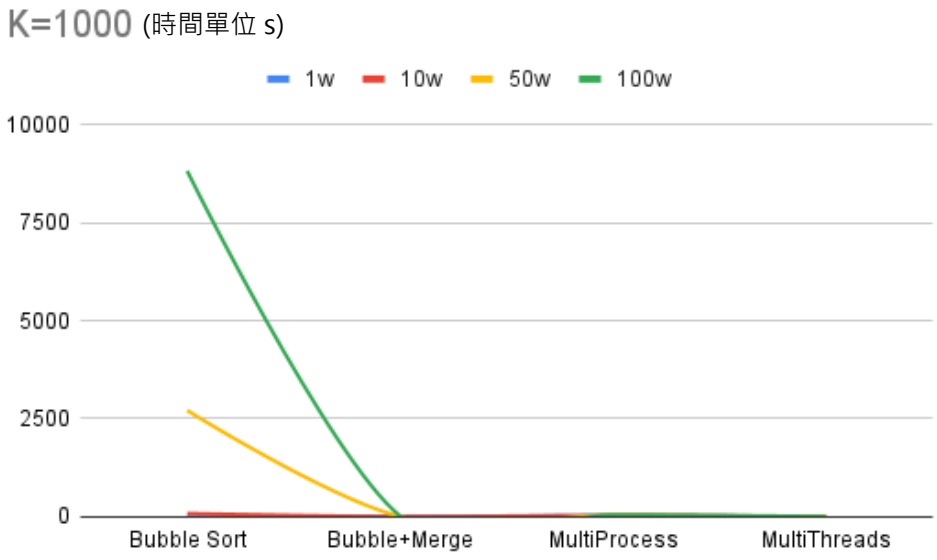
2. K=500

	1w	10w	50w	100w
Bubble Sort	0s	81s	2713s	8829s
Bubble+Merge	0s	0s	7s	30s
MultiProcess	15s	16s	24s	47s
MultiThreads	0s	0s	2s	10s



3. K=1000

	1w	10w	50w	100w
Bubble Sort	0s	81s	2713s	8829s
Bubble+Merge	0s	0s	3s	15s
MultiProcess	31s	30s	34s	46s
MultiThreads	0s	0s	1s	4s



4. **探討造成執行速度差別的原因**

(1) 方法二 K 值越大，執行時間越短，因為資料數越小 bubble sort 速度越快。

(2) 方法三和方法二四相比的執行時間較長，因為大部分時間都浪費在 shared memory 的同步資料。

- 解決方法: 用一維陣列儲存資料，並紀錄每筆資料的起始位置，這樣可以直接在陣列上排序，不須把資料傳來傳去。

(3) MultiThread 執行速度最快，因為所有 thread 平行執行且共用記憶體，不須同步資料。

五、撰寫程式時遇到的 bug 及相關的解決方法

1. 用 multithread 合併資料時，若沒有等待所有 thread 執行結束，會造成合併錯誤。

- 解決方法: 等待所有 thread 結束再進入下一次合併。

```
// =======merge ======
while( vec2.size() != 1 ){

    for( int k = 0; k < vec2.size(); k+=2 ){
        threads.push_back(thread(merge, k)) ;
    }

    for( int i = 0; i < threads.size(); i+=1 ) {
        threads.at(i).join();
    }
    threads.clear() ;

    mu.lock() ;
    for( int k = 0; k+1 < vec2.size(); k+=1 ){
        vec2.erase(vec2.begin()+k+1) ; // delete k+1
    }

    mu.unlock() ;

}
```