**第一支程式說明**

**10520191 資工三甲 黃博泓**

**(一)使用的開發環境**

使用軟體:Dev C++、VSCode // win10、Linux。

使用c++語言，完成第一題、第二題、第四題的功能，第三題有bug錯誤。依據command執行對應的程式功能，利用vector資料結構儲存切完的資料片段，在做multi-thread時，調用bubblesort()與mergsort() function需注意要使用joinable()函式來確保thread的不會往下進行運算，最後做mergesort()時在做完一層merge時，也要記得join，merge合併到第一個資料區間(避免資料間的race condition)，最後輸出資料的排序結果與對應的cpu使用時間到檔案中。

**(二)分析**

以下是依據題型分別跑了一千筆，一萬筆，十萬筆，五十萬筆的資料，X軸為題型，Y軸為執行時(以秒為單位)。

第一題單純使用單process 跑BubbleSort()。

第二題使用multi-thread各自跑BubbleSort()與MergeSort()。

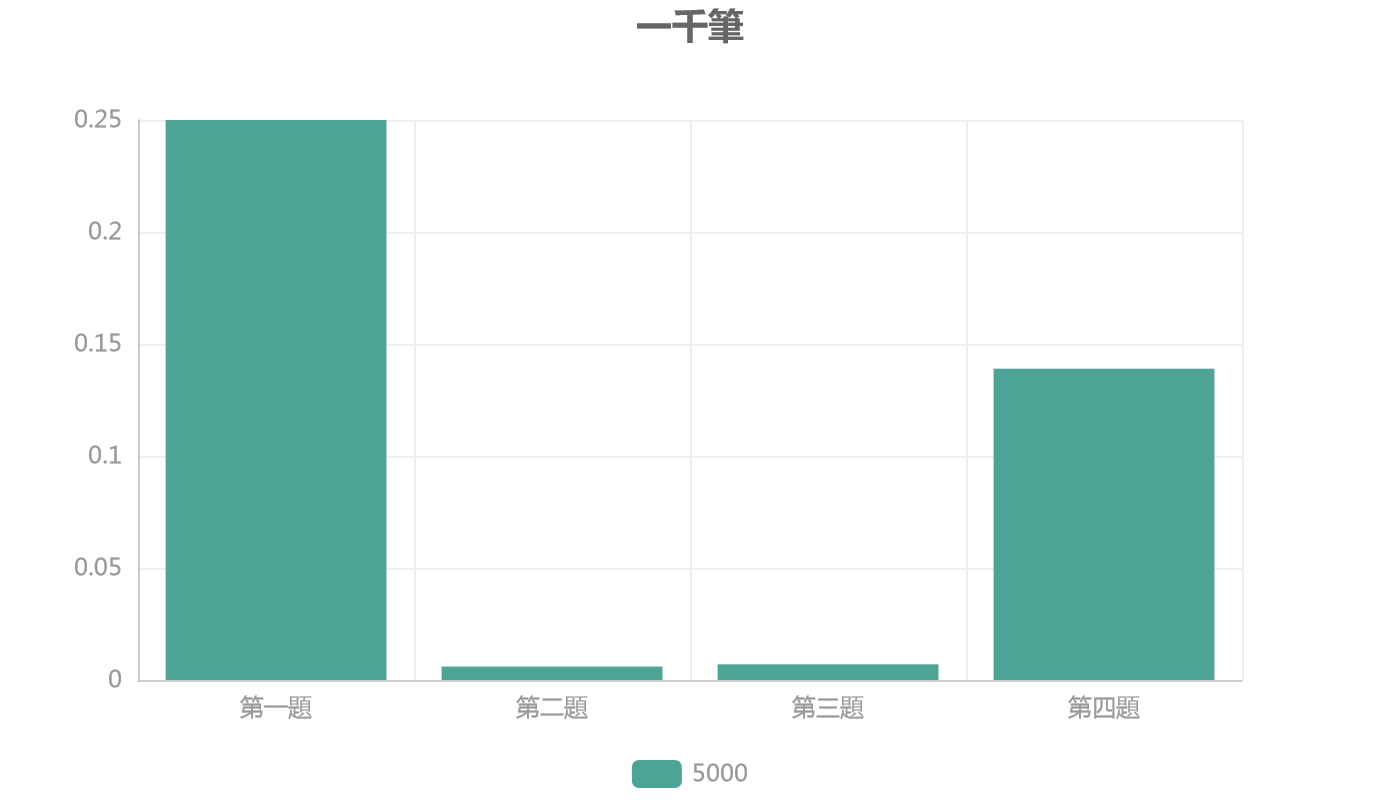
第三題使用multi-process各自跑BubbleSort()與MergeSort()。

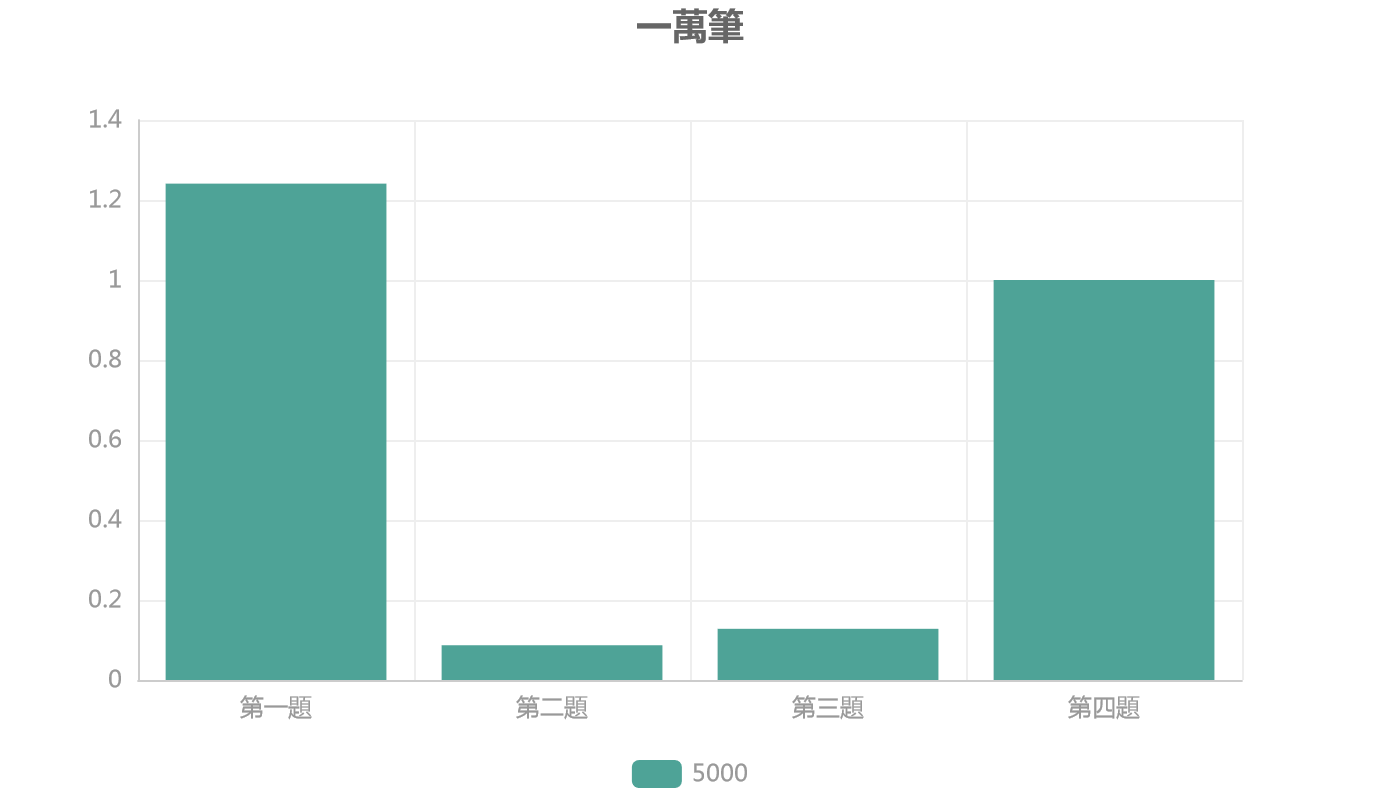
第四題使用單Process來跑BubbleSort()與MergeSort()。

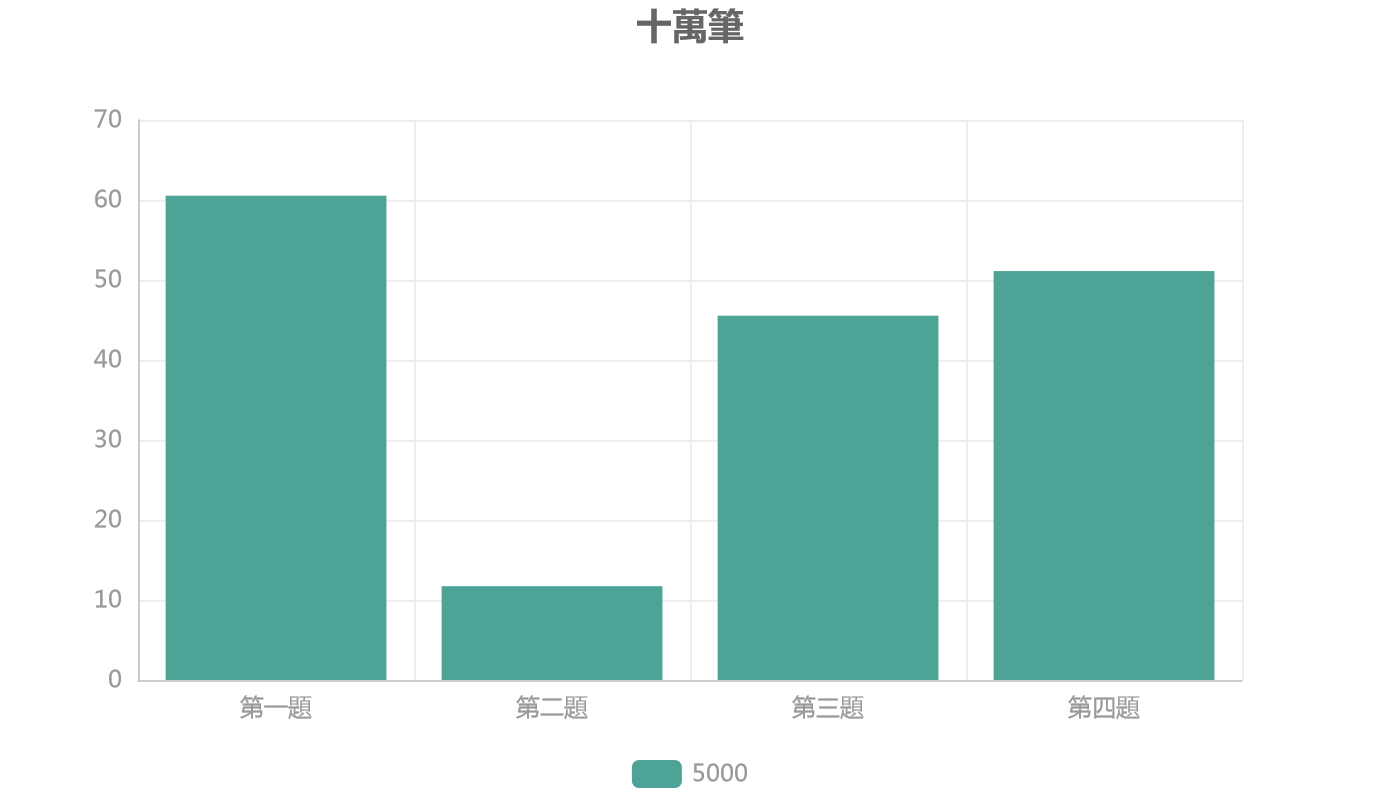
第二題、第三題使用concurrent平行化技術來實現。

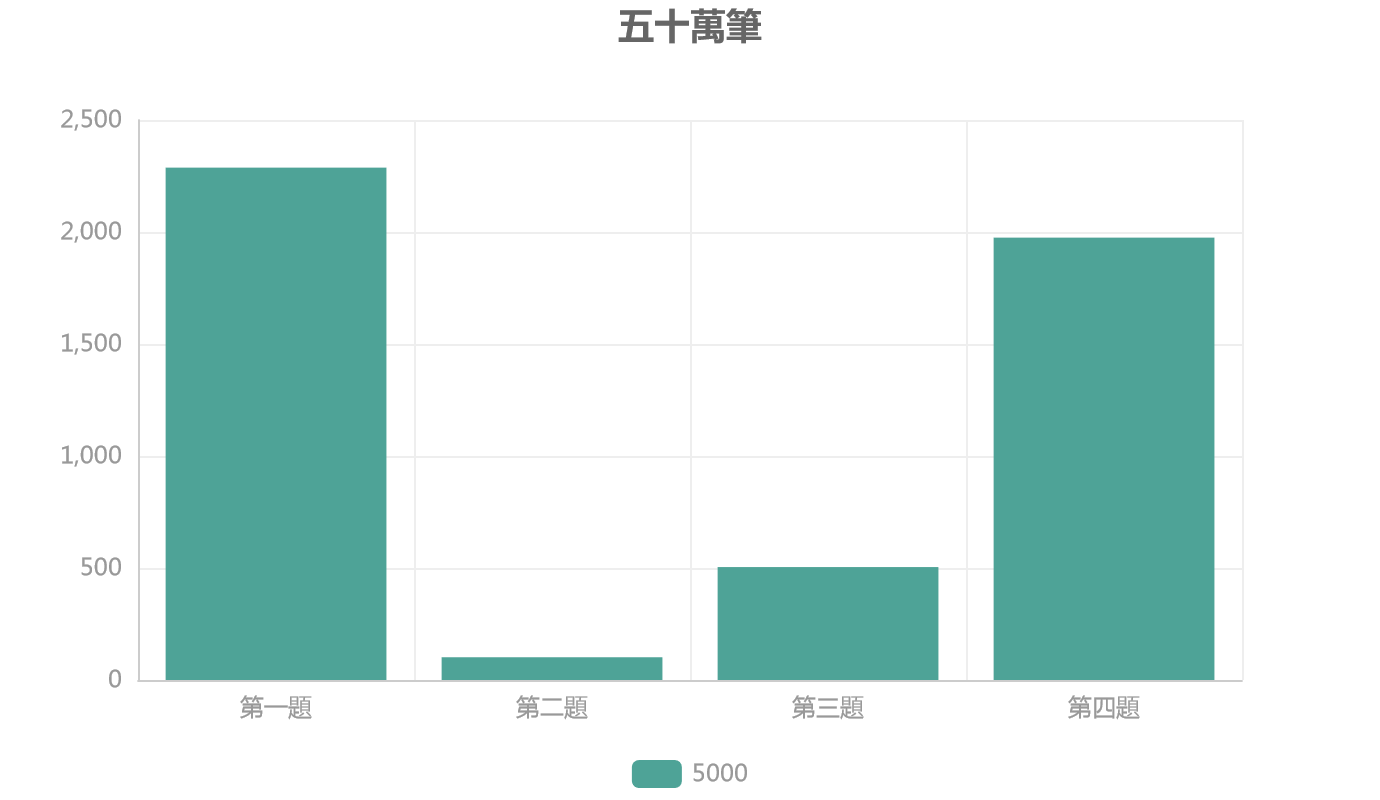
**(cut\_number = 100)。**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 第一題 | 第二題 | 第三題 | 第四題 |
| 1000筆 | 0.250 | 0.006 | 0.007 | 0.139 |
| 10000筆 | 1.241 | 0.087 | 0.128 | 0.634 |
| 100000筆 | 60.541 | 11.726 | 45.542 | 51.12 |
| 500000筆 | 2287.44 | 101.69 | 504.3 | 1975.1 |

****

****

****

****

從圖表中可以得知，multi-thread所執行的速度是最快的，而multi-process次之，先bubble後Merge，最後則是都用Bubble最慢。而第二題與第三題的差別在於，製造multi-thread的成本無multi-process高，因multi-thread彼此之間共用同份process的data、code section，所以create的成本較低，但multi-process彼此之間為不同的memory space，每次create需要額外的時間配置新的process空間，成本較高，速度較慢。