Programming Assignment Report

工海三

B05602022

盧庭偉

1. 演算法流程 (Algorithm Flow) <解釋程式的運作>

由於資料量不確定，因此不能確定資料的出現順序要使用什麼資料型態來記錄。因此我四個程式基本都是由許多 template function 組成，再程式開始時會先檢查資料的 size，若在 unsigned short 範圍內則用 unsigned short(用 unsigned 是因為做 index 的時候用不到負號很浪費)，超過則用 unsigned int。

而程式運作流程則與我使用的資料結構有關，因此在第二題會詳述。

2. 資料結構 (Data Structure) <解釋"特別的"資料結構>

原本我 insertionSort 是使用 std::list，其他是使用 std::vector 來儲存自己定義的 node 類別，每個 node 儲存字串的內容及出現順序。但經過反覆的測試後發現這些資料結構在資料的交換速度上遠不及我的預期。尤其是 list，理當能在 O(n)內完成插入，但實際速度卻非常緩慢，甚至不及 vector。因此最後我決定用最簡單的一個 string array: words 來儲存字串，一個 unsigned short 或 unsigned int array: indexs(視檔案大小)來儲存順序。但又發現同時

對兩個 array 排序有點慢，因此決定再開一個 unsigned short / int 的 array: sorted，作為一組資料的 hashmap(若 sorted 的值為 X 表示 words 中的 X 個字及 indexs 中第 X 個數)，我們只要對這個 hashmap 做 sorting，要比較時就用 words 查表， 如此一來就能節省很多資料交換上的時間。

3. 問題與討論 (Discussion) <討論實作中遇到的問題及疑問>

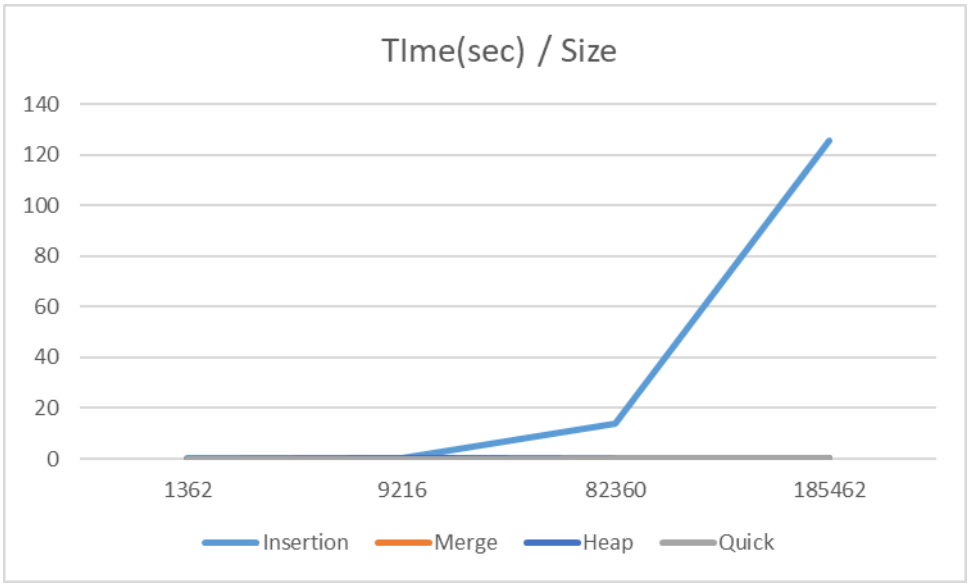
遇到最大的問題就如同上面所講的， std 的資料結構效率不如預期。

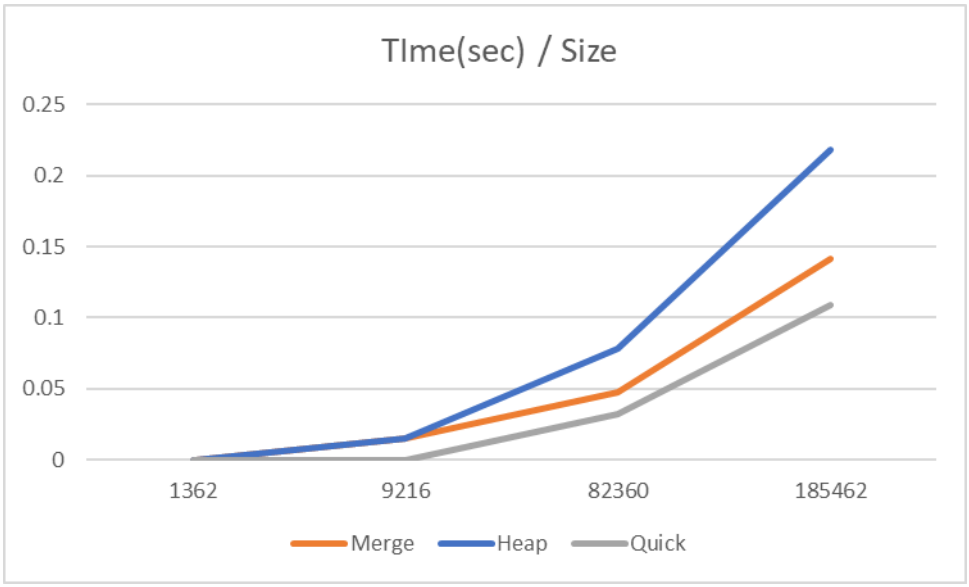
4. 效能紀錄與分析

	Insertion Sort		**Merge Sort**		**Heap Sort**		**Quick Sort**	
	Time (sec)	**Memery (MB)**	**Time (sec)**	**Memery (MB)**	**Time (sec)**	**Memery (MB)**	**Time (sec)**	**Memery (MB)**
Case1	**0.015**	**14.328**	**0**	**14.268**	**0**	**14.136**	**0**	**14.136**
Case2	**0.265**	**15.048**	**0.015**	**15.708**	**0.015**	**15.048**	**0**	**15.048**
Case3	**14.032**	**22.444**	**0.047**	**31.448**	**0.078**	**22.444**	**0.032**	**22.444**
Case4	**125.844**	**31.632**	**0.141**	**52.888**	**0.218**	**31.632**	**0.109**	**31.632**

5. CPU time vs. problem size

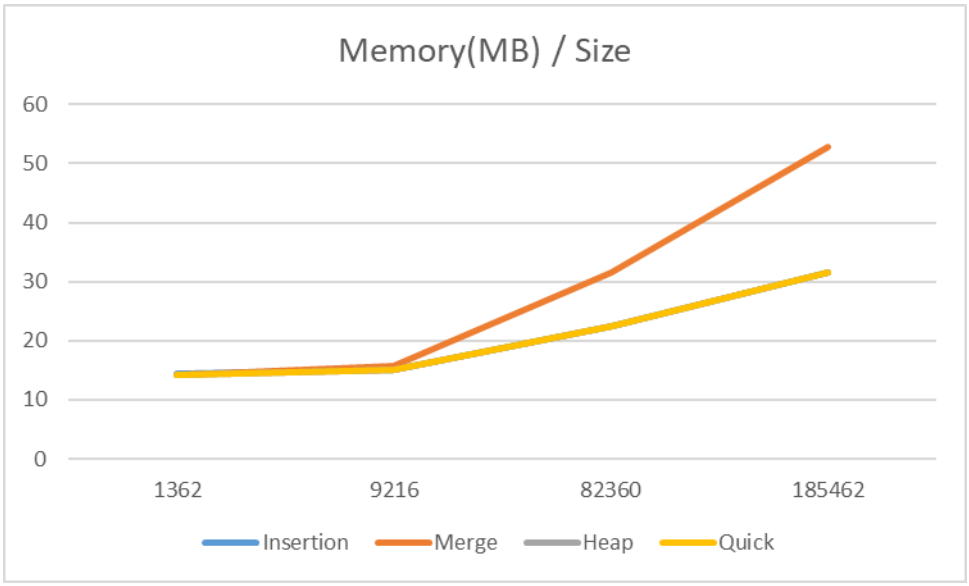
由於第一張圖 insertionSort 曲線上升太快，因此畫了第二張圖展示其他 sorting





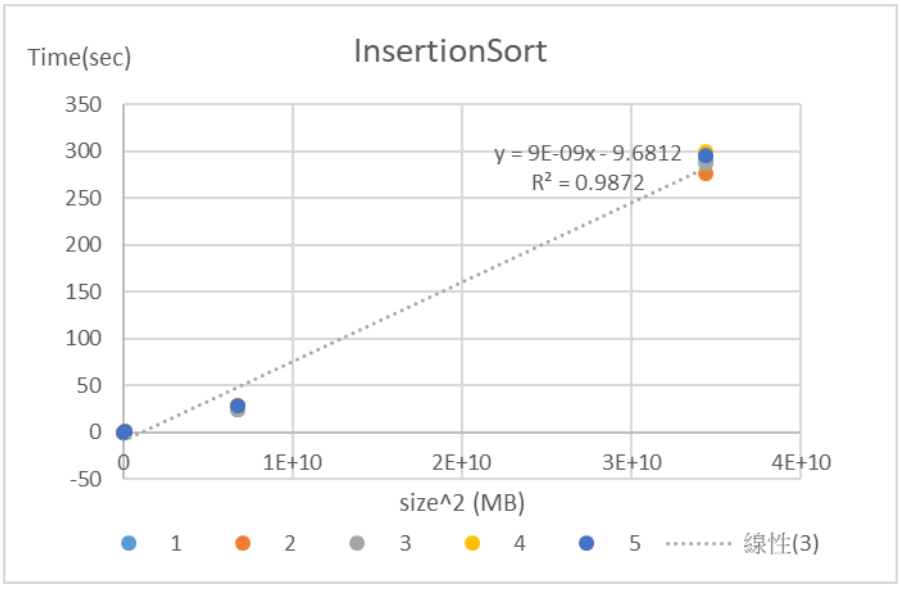
6. Memory vs. problem size

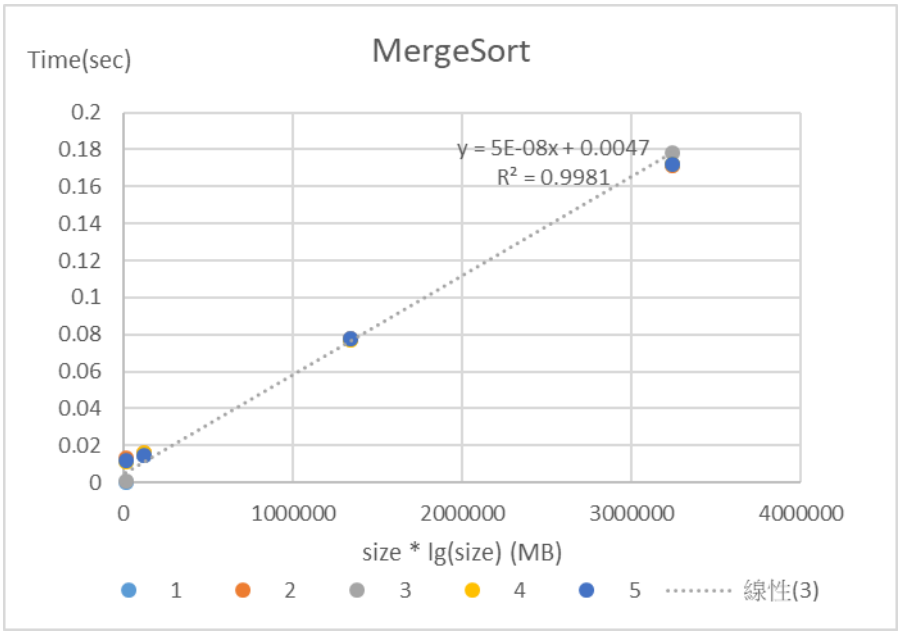
如第二題所說，由於我資料結構的選擇，因此我所使用的記憶體大小除了 mergeSort 之外都一樣(因為 merge 進行時會需要用到額外記憶體)

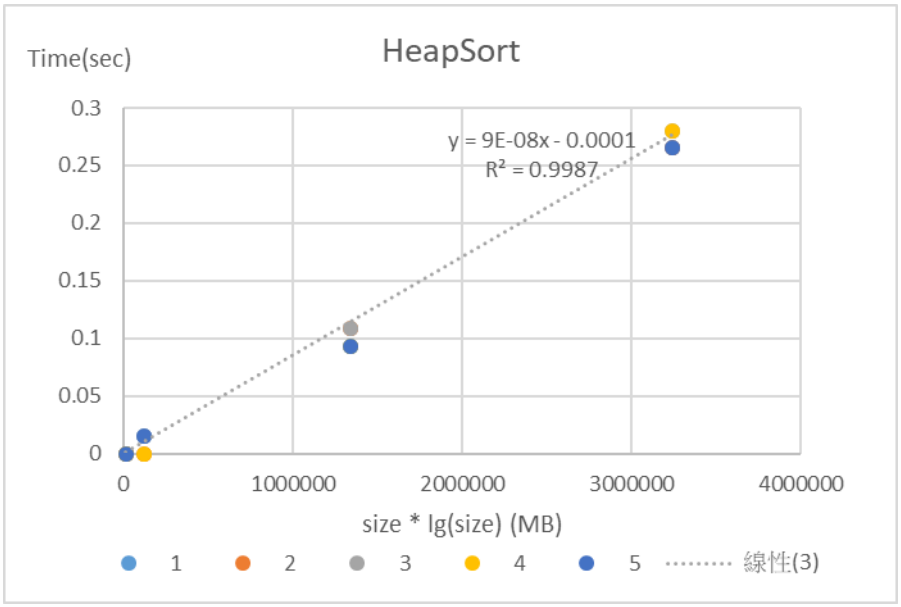


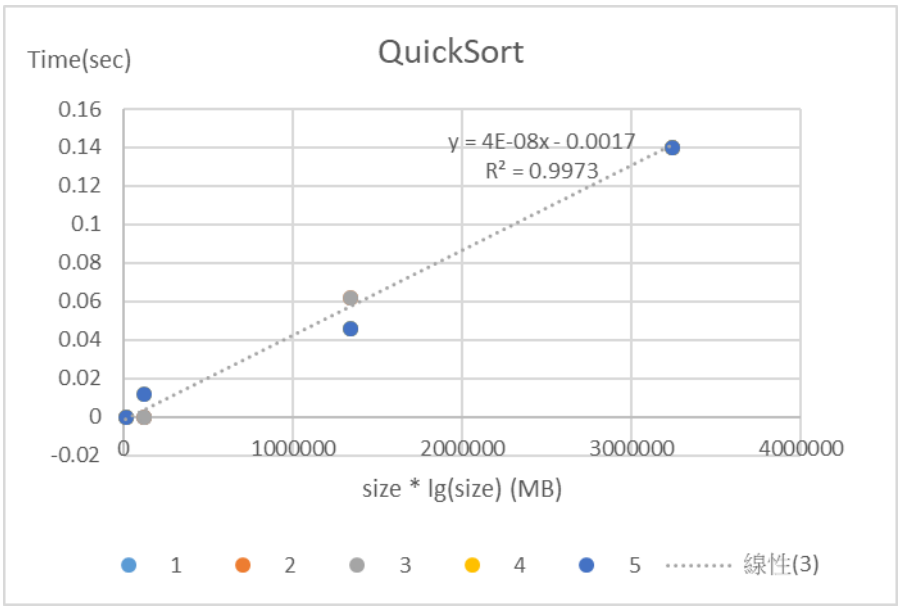
7. Bonus

為了找線性關係，因此 insertionSort 我們用 input size 的平方對時間作圖，其餘用 input size * lg(input size)對時間作圖。




MergeSort
Time(sec)
y = 5E-08x + 0.0047
R² = 0.9981
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
1000000
2000000
3000000
4000000
size * lg(size) (MB)
1
2
3
4
5
線性(3)


HeapSort
Time(sec)
y = 9E-08x - 0.0001
R² = 0.9987
0.3
0.25
0.2
0.15
0.1
0.05
0
0
1000000
2000000
3000000
4000000
size * lg(size) (MB)
1
2
3
4
5
線性(3)


QuickSort
Time(sec)
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
y = 4E-08x - 0.0017
R² = 0.9973
0
1000000
2000000
3000000
4000000
size * lg(size) (MB)
1
2
3
4
5
線性(3)