資工四 曾俊宏

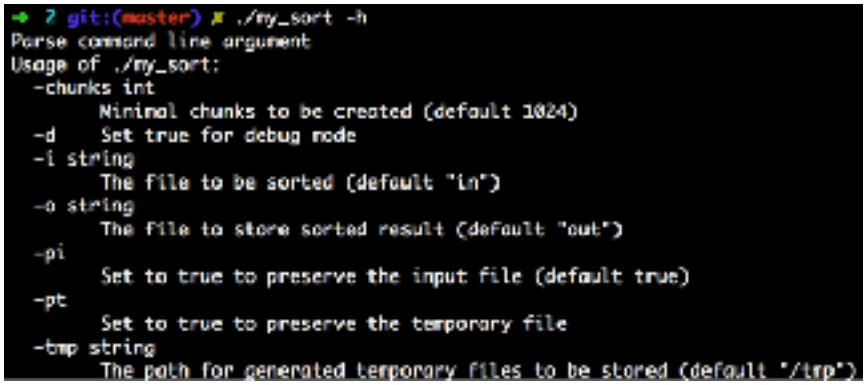
External sort

語言

撰寫程式用的語言是 Golang。這也是我第一次使用 Golang 撰寫有效能要求的程式。

Golang 真的在編譯上和執行效能上都不差，而且寫起來比 C 語言輕鬆許多。在撰寫時 error 幾乎都在存檔時就已經被 mark 起來了，修改容易！語法也比 C 稍微清晰些，指標用起來比較輕鬆。

執行參數的設計也非常容易，有個 flag 套件可以幫我們輕鬆的 parse 參數進來。



整個作業遇到最大的問題其實出在IO。對於 Golang 的 IO 機制不熟，剛開始沒有使用 bufio，導致單單是開檔案寫檔案就花掉80%的時間！

演算法

我是使用external merge sort來進行撰寫的，方法幾乎如同上課時所提。

對於input檔案，我會看使用者說要分成幾個chunk，之後盡量的切成使用者所指定的份數（有考慮份數比檔案大小大的情況）。切割的 chunk 在進行寫檔之前，會先排序。

Chunk 都切好後，會使用 winner tree 來 merge 。對於 winner tree 的 node，我存的資料有二：從哪個leaf來的，還有他的 value。所以對於pop的更新操作，我是從bottom-up的更新。

```
type TreeNode struct {
    value  string
    origin int
}
```

中間檔案的部分，我沒有多做特殊處理，所以是會使用 原始檔案 、 中間檔案 、 結果檔案 三個空間。

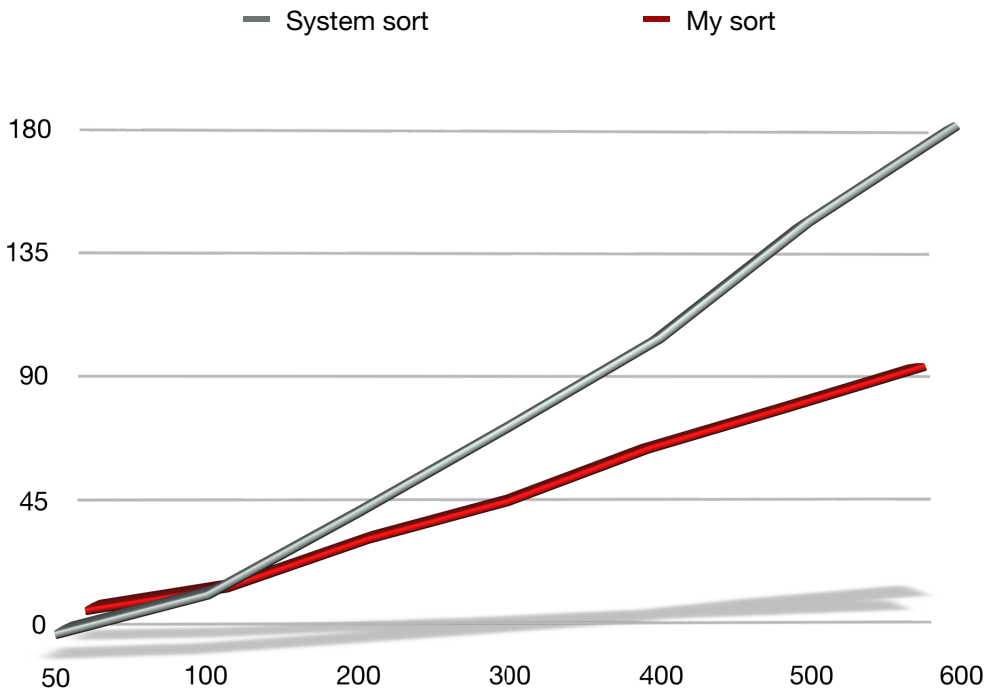
效能

我有寫一支亂數產生器，並利用他產出 50, 100, 200, 300, 400, 500, 600 MB不等的數字資料。資料格式為一行一個數字。

在系上的 linux.cs.ccu.edu.tw工作站上，我跑了一個自己寫的 benchmark 程式，他分別執行並且比較了系統內建的 sort 和我的 sort 的速度(chunks = 1000)。正確性的部分，我在兩個 sort 跑完後，使用了 cmp 進行比對。

以下圖表是執行結果。當測試資料大過70MB左右時，我的 sort 就會比較快了！

測資大小 (MB)	Sort 使用時間 (秒)	My sort 使用時間 （秒）
50	8.04	10.50
100	20.23	17.95
200	46.07	34.19
300	72.87	46.85
400	100.49	64.46
500	137.35	78.33
600	168.07	92.51



Ngram

拿來跑的資料來源是 PPT 的熱門看板，我們利用陳丕祐同學寫的 ptt 爬蟲，大家分頭去不同的看板爬了數十到數百MB的文章下來進行 ngram 。

資工四 曾俊宏

Ngram 的程式基本上就是對於 定長度 n 的字串輸出，如果遇到常見標點符號則跳過。

對於ngram輸出的檔案進行排序後，線性的跑過去統計重複出現的詞的個數後，再對於詞的出現個數再度排序，即可以得到一個由多到少的詞頻表。

有個問題尚未解決，就是如果 ngram 的 n 從 4~9的話，長詞優先的部分可能得套用上堂課程所講述的 hash table ，但是實作尚未完成，因此目前只有統計結果，沒法做到長詞優先。