作業系統

HW3 - Page Replacement 書面報告

資訊三甲 10927159 林玟君

一、開發環境：

Visual Studio Code (Python)

二、實作方法和流程：

根據使用者輸入的 input 檔，讀取方法種類、Page Frame 數量，以及各個 Page Reference 的次序，以一維串列的方式儲存 process 相關資訊，並根據不同的方法種類進行不同的 Page Replacement 方法。最後依據題目格式及檔名要求寫入對應檔案，輸出檔中應包含依據 Page Reference 抵達次序而改變的 Page Frame，並標示是否進行 Page Fault，最後輸出 Page Fault、Page Replaces、Page Frames。

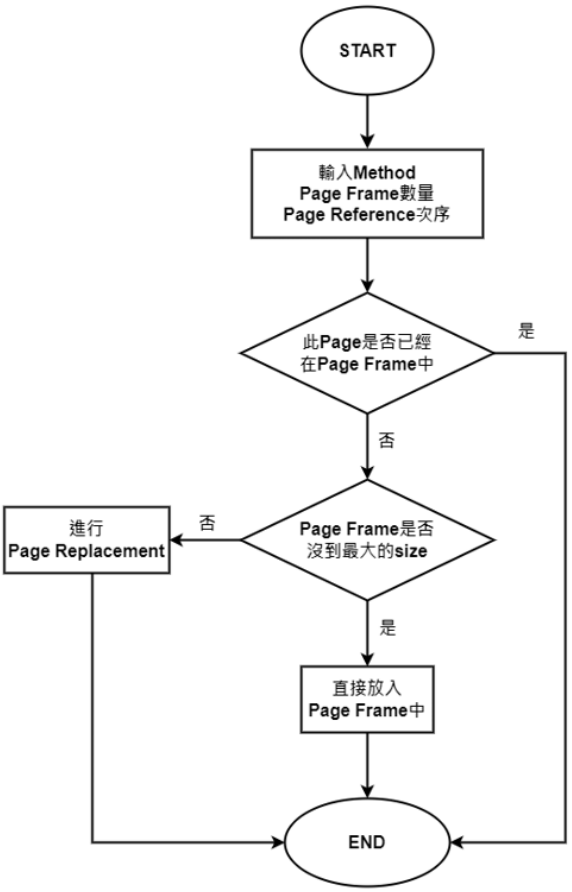

圖 1：Page Replacement 之簡易流程圖

方法一：First In First Out (FIFO)

FIFO 是根據 Page Reference 的次序來置換 Page Frame 裡的內容，當有新的 Page Reference，會先查看 Page Frame 裡面是否有此 Reference，若有的話則不需要再做置換，保持原本的 Page Frame；若無此 Reference 則將 Page Frame 中最先抵達的 Reference 移除，並將最新到達的 Reference 加入 Page Frame 中，若 Page Frame 還不滿最大的數量，則可以直接放入。

方法二：Least Recently Used (LRU)

LRU 是先將過去歷史紀錄中最久沒被 Reference 到的 Page 置換掉，當有新的 Page Reference，會先查看 Page Frame 裡面是否有此 Reference，若有的話則不需要再做置換，將最新抵達的 Page Reference 移至 Page Frame 的最前面；若無此 Reference 則檢查哪個 Page 最久沒被 Reference，並將其替換，若 Page Frame 還不滿最大的數量，則可以直接放入。

方法三：Least Frequently Used (LFU) + FIFO

LFU 是將過去歷史紀錄中最少被 Reference 到的 Page 置換掉，但此方式有可能造成新進的 Page 不斷被置換掉的錯誤情形發生。當有新的 Page Reference，會先查看 Page Frame 裡面是否有此 Reference，若有的話則不需要再做置換，保持原本的 Page Frame；若無此 Reference 則檢查哪個 Page 最少被 Reference，並將其替換，若 Page Frame 還不滿最大的數量，則可以直接放入。

方法四：Most Frequently Used (MFU) + FIFO

MFU 跟 LFU 非常相似，是將最常被 Reference 到的 Page 置換掉，其餘則和方法三相同。

方法五：LFU + LRU

LFU 結合 LRU 的方式，會先檢查 Page Frame 中哪個 Page 被 Reference 的次數最少，若次數相同，則會根據 LRU 的規則來決定誰會被置換，也就是會先置換掉過去歷史紀錄中最久沒被 Reference 到的 Page。

方法六：ALL

方法六為方法一到五的集大成。

三、不同方法比較：

Page Reference：Input1 → 1 2 3 4 1 2 5 1 2 3 4 5
(Page Frame 個數為 3)
Input2 → 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
(Page Frame 個數為 3)

	FIFO	LRU	LFU+FIFO	MFU+FIFO	LFU+LRU
Input1	9	10	10	9	10
Input2	15	12	13	15	11

表 1：Page Fault 次數比較表

	FIFO	LRU	LFU+FIFO	MFU+FIFO	LFU+LRU
Input1	6	7	7	6	7
Input2	12	9	10	12	8

表 2：Page Replacement 次數比較表

	FIFO	LRU	LFU+FIFO	MFU+FIFO	LFU+LRU
Input1	3	3	3	3	3
Input2	3	3	3	3	3

表 3：Page Frame 次數比較表

可根據上表 1 以及表 2，觀察並推測出以下結論。

無論是 Input1 或是 Input2 的測資，套用到任何一種方法，Page Fault 的次數一定會大於 Page Replacement 的次數，只要有想要 Reference 的 Page 不在當前的 Page Frame 中，便會進行 Page Fault，而 Page Replacement 則是在此基礎之上，且 Page Frame 已滿，才會進行。而由此結論結合表 3，可以觀察出 Page Fault 次數 = Page Replacement + Page Frame。

而從 Input2 的各項次數比較表可以發現方法六（LFU+LRU）的 Page Fault 和 Page Replacement 次數是最少的，而這種比較方式是先查看誰是最少被 Reference 的 Page，若次數一樣，則檢查哪個 Page 是最久沒被 Reference 到的，而此比較方式適合 Input2 的 Page Reference 資料排序。

雖然輸入的測資沒有非常多，但我覺得這五種方式各有各的優缺點，適用於不同的 Page Reference 特性的狀況。舉例來說，若常常會有新的 Page 需要被 Reference，則可能不適用 LFU，因新進的 Page 被 Reference 肯定會比較少，但是此方法卻會不斷的將 Reference 最少的 Page 置換掉，導致剛進來的 Page 又被置換出去。

四、結果與討論：

增加 Page Frame，應該會讓 Page Fault 及 Page Replacement 的次數減少。然而，在某些情況下，反而造成更多的 Page Fault 及 Page Replacement，我們稱這是畢雷笛反例，而畢雷笛反例僅是少數特例。

而 Input1 的測資即剛好符合畢雷笛反例，將此測資套用到方法一的 FIFO 即可得到以下之次數比較表：

Page Reference→ 1 2 3 4 1 2 5 1 2 3 4 5

Page Frame	1	2	3	4	5
Page Fault	12	12	9	10	5
Page Replacement	11	10	6	6	0

表 4：模擬畢雷笛反例之次數比較表

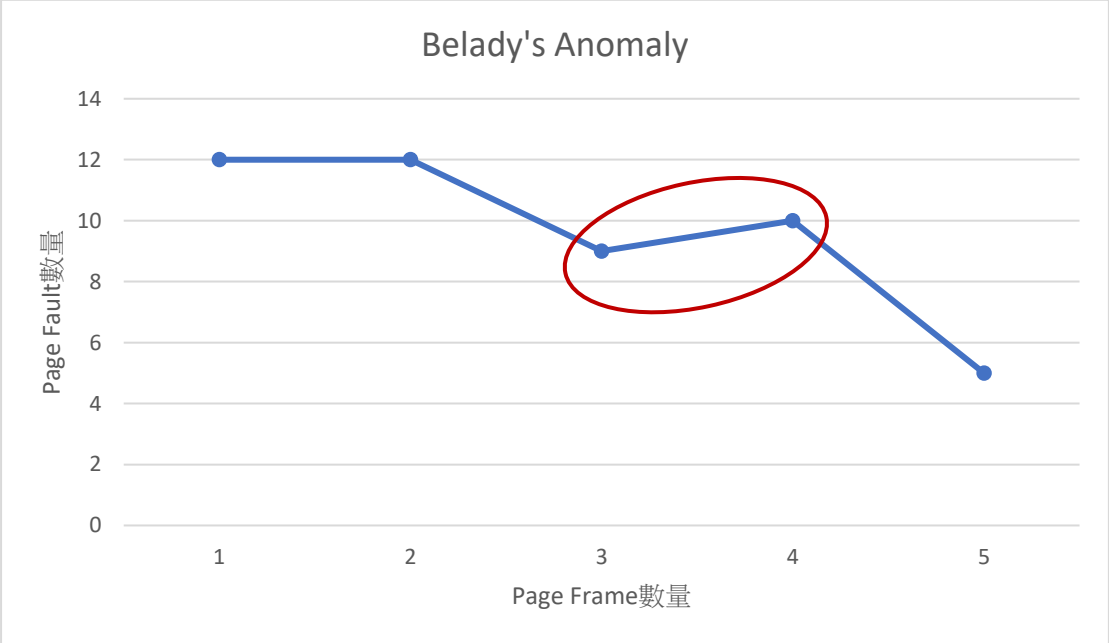

圖 2：畢雷笛反例之結果折線圖

由表 4 可以發現當 Page Frame 為 4 時，Page Fault 的次數卻比當 Page Frame 為 3 時，還要來的多，此例即為一經典的畢雷笛反例。

五、心得：

我覺得這次的作業還滿有趣的，雖然沒有前兩次複雜，執行時間也沒有那麼久，但我在理解第五題的題目的時候還是有小理解錯誤，所以導致第五題花了一點時間才打出來，但打完的時候還是很有成就感。在上網找畢雷笛反例的時候花了點時間，還去問了 ChatGPT，後來才發現 Input1 的測試數據其實就是一個畢雷笛反例！這個反例也讓我覺得很有趣，發現這個反例的人真的很厲害，因為這種例子應該不常見才對。最後希望我的 OS 期末考也能順順利利的度過！