OS-HW3-Page Replacement

資訊三甲 10927145 李俐瑩

一、開發環境: Visual Studio Code (Python)

二、實作方法、資料結構與流程:

- **Input:**

  將檔案第一行讀入後，分別存要使用的方法及 page frame 個數，將第二行讀入後存在一個 list 中，且每個數字都被放在一個單獨的子列表中。

- **FIFO:**

  將資料從第一個跑到最後一個，首先檢查現在這個 page 是否在 page frame(使用 list)內，若有就直接紀錄現在 page frame 的內容，若沒有的話就增加 page fault 的次數，並判斷 page frame 是否已滿，若還沒就將這筆資料放在 page frame 的第一個，若已滿代表需要做置換的動作，因此增加置換次數，再將 page frame 最尾端的資料 pop 掉後再 page in 至 page frame 的最前端。

- **LRU:**

  此方法與 FIFO 大同小異，差別在於第一步檢查 page 是否在 page frame 內，若發現已經在內，則要將這個 page 移到 page frame 的最前端，代表重給 time stamp 的動作。

- **LFU+FIFO:**

  這個方法我是用 FIFO 的方法再加上 counter 的設計來完成的，我多用了一個 list 來當作 counter，在加入 frame page 後就加入一個[0]的子 list，且每次被引用都在那個 frame 的位置+1，且若被 page out 要將該位置的 counter 刪除，每次在選擇誰是犧牲者時就找到 counter 內的最小值，若有多個最小值就選擇最後一個最小值，代表採用 FIFO 策略。

- **MFU+FIFO:**

  這方法與上一個大同小異，差別在於犧牲者選擇 counter 的最大值。

- **LFU+LRU:**

  這個方法相較於第三題採用了 LRU 的方法，因此若未發生 page fault 引用時，要將這個 page 移到 page frame 最前方。

- **ALL:**

  這題我直接將前五個方法的 method 個執行一次，但要注意的是，因為我的 page frame 及代表 page fault 的’ F’ 記號都是存於同個 reference variabe，因此傳入時須先 deep copy。

三、不同方法之間的比較:

Reference string:

Input1: 123412512345

Input2: 70120304230321201701

Page frame 為 3

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

Page fault 次數比較

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

Page replacement 次數比較

從上表可以發現 Page replacement 次數總是比 Page fault 次數少，因為若發生 replacement 就代表一定發生了 page fault，而反之則不一定，因為若還有空的 frame 就不需要 replacement。

另外可以得知 FIFO 的次數通常較高，這是因為若是在 3 個 page 內沒有再度被引用就會被置換掉，而 LRU 會將再度被引用的 page 重新得到 time stamp，使得最近被引用過的不會被換掉。

四、結果與討論:

Input1:　123412512345，FIFO

Page frame 個數	1	2	3	4	5
Page fault	12	12	9	10	5

這個例子使用了 FIFO，且為畢雷笛反例，當 page frame 變多時 page frame 不一定會下降，如表中所示，當 page frame 個數從 3 變為 4 時，page fault 次數反而增加了。