1. 開發環境

Windows11,Anaconda Python 3.8(需安裝程式裡面含有的package才能正常的執行)

1. 實作方法和流程

FIFO:檢查有無Page Fault有Page fault的時候檢查page frame是否滿了，如果滿了，把先進page frame的page移出，在加入新的page，如果沒滿就直接加入page，沒有Page fault的話就什麼事都不用更動。

LRU: 檢查有無Page Fault，有Page fault的時候檢查page frame是否滿了，如果滿了，把最早進page frame的page移出，在加入新的page，如果沒滿就直接加入page，沒有Page fault的話，刪除Page frame裡相同的page，再加入Page，等於重新給予時間標記。

LFU used LRU:每個page frame都會有一個Counter，假如有Page Fault會找出Counter值最小的Page frame，移除掉再加入新的，假如沒有Page Fault的話，會移除掉在Page frame裡相同的Page，然後再重新加入，更新Counter table裡，每一個Page的時間標記，會對新來的Page的Counter值+1，如果有被參考就對Counter+1。

MFU-FIFO:基本上和FIFO是差不多的，檢查有無Page Fault有Page fault的時候檢查page frame是否滿了，如果滿了，把先進page frame的page移出，在加入新的page，如果沒滿就直接加入page，沒有Page fault的話就什麼事都不用更動，最後都要更新Counter。

MFU-LRU: 每個page frame都會有一個Counter，假如有Page Fault會找出Counter值最大的Page frame，移除掉再加入新的，假如沒有Page Fault的話，會移除掉在Page frame裡相同的Page，然後再重新加入，更新Counter table裡，每一個Page的時間標記，會對新來的Page的Counter值+1，而這邊用的MFU-LRU的方法，是會因為產生Page Fault而歸零swap out Page的Counter值。

3. 不同方法的比較

Page Fault 次數 Page Replace 次數

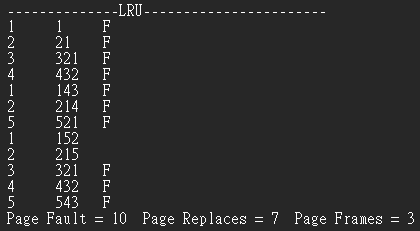
MFU-FIFO vs FIFO :



因為做法和想法上圍繞著FIFO，FIFO的順序和MFU的基本上就等於Counter的功能了，結果是一樣的。

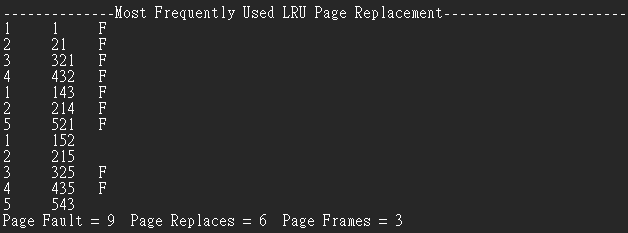
Page Fault 次數 Page Replace 次數，會因Page Frame的大小改變，如果Page frame夠大能減少次數過多的問題，不過還是要考慮Belady’s Anomaly。

LRU vs FIFO :



這兩個差別跟MFU-FIFO vs MFU-LRU的概念是差不多的，這兩者在刪除相同page上些許不同，FIFO是不動作，LRU是刪除給予新的Counter，同上。

MFU-FIFO vs MFU-LRU:

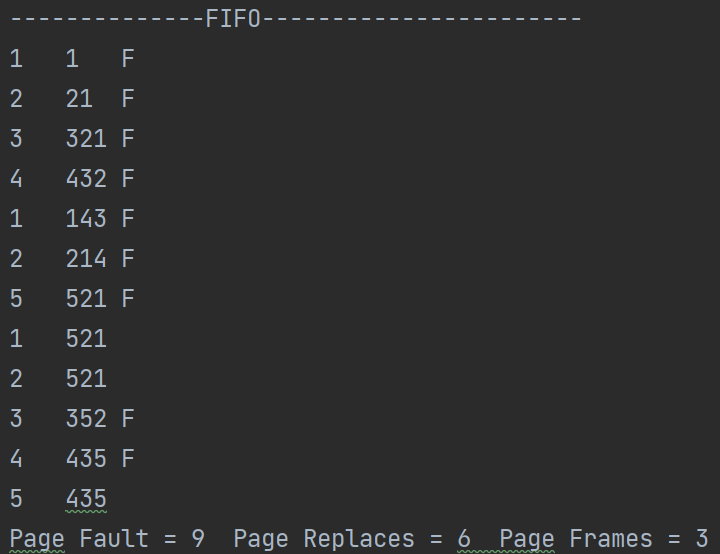
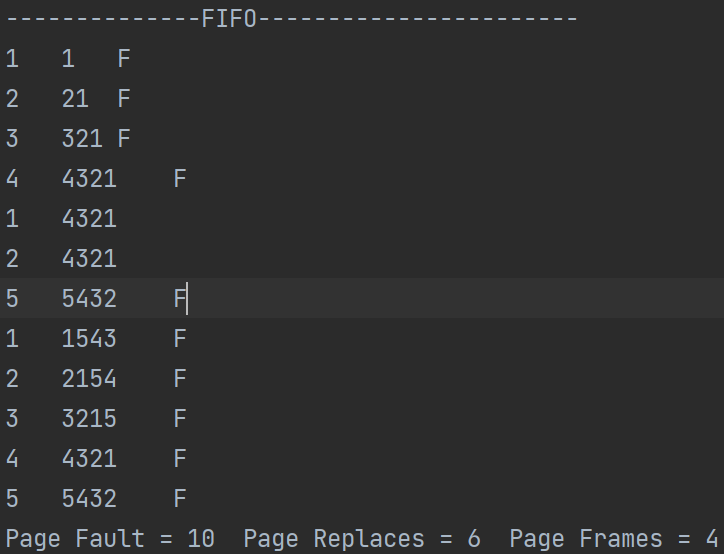


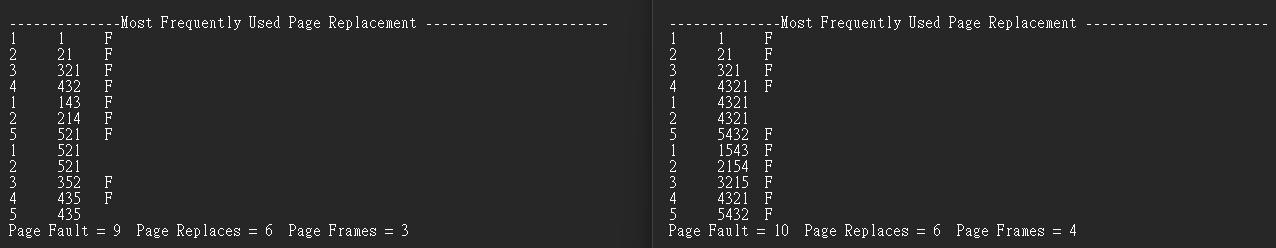
LRU版本會改變page frame裡的順序，而FIFO版本的不會，就照輸入的順序，差別在於說沒有Page Fault時，優先度能夠提升，在有限的大小裡面，常被使用到的不太會被swap out出去，減少了Page Fault和Page Replace的發生，而FIFO維持不變，只能依照輸入順序，並沒有在Page Frame把比較少用的swap out，提升其他常用的優先度，所以Page frame和Page replace較高。

4. 結果與討論

畢雷笛反例(Belady’s Anomaly)

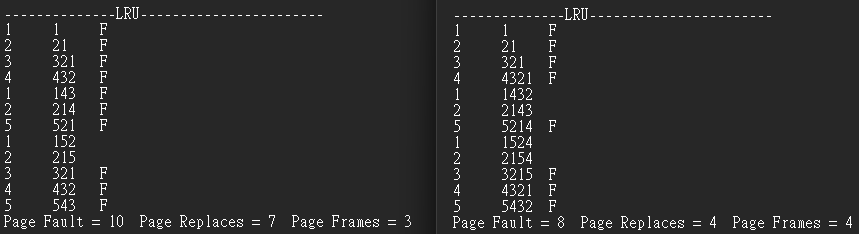
在比較Page frame : 3 vs Page frame : 4

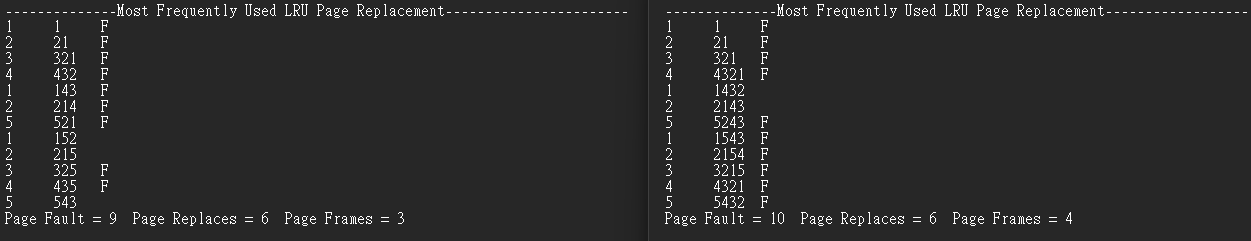
 



會因為輸入順序的關係，還有Page Frame當時裡面Page的狀態，來決定Page Fault 或是 Page replace的數值，依這個例子FIFO、MFU-FIFO來看，Page Frame不一定比較大就能減少Page Fault的錯誤發生。

LRU比較Page frame : 3 vs Page frame : 4





LRU在增加Page Frames大小起了作用，減少了2個Page Fault和3個Page Replaces。

但MFU-LRU卻不減反增，可以減少前面的Page fault但造成後面的Page Fault變多。