

1. 開發環境：Windows10 + VScode + TDM-GCC + C++2a

2. 資料結構：

`struct Data; // 用於儲存針對單一 frame 的資料(frame, queue, fault)`

`struct SUM_UP; // 用於儲存單一方法的最後一行的統計訊息`

對於每一個 page 中的對應之 queue 使用 `std::vector` 儲存

對於 LFU、MFU 的計數使用 `std::unordered_map` 實現類似 hash map 的功能

3. 運作流程：

讀檔 -> 根據 method 執行對應的 function(s) -> 寫檔 -> 結束

4. 實作方法：

※所有方法都是依序 iterate 單一 frame(page)

甲、FIFO：

(a) 當 queue 沒有目前的 page 時：

若 queue 已滿，把 queue 中最後一個作為 victim 移出，更新 page replace 次數

更新 page fault，並在 queue 最前面插入目前的 page

(b) 將 queue 轉為 `std::string` 以便寫檔

乙、LRU：

(a) 當 queue 沒有目前的 page 時：

若 queue 已滿，把 queue 中最後一個作為 victim 移出，更新 page replace 次數

更新 page fault，並在 queue 最前面插入目前的 page

(b) 否則就刪除 queue 中已經存在的、與目前相同的 page，並在 queue 最前面插入目前的 page

(c) 將 queue 轉為 `std::string` 以便寫檔

丙、LFU + FIFO：

(a) 更新對應目前 page 的 count

(b) 當 queue 沒有目前的 page 時：

若 queue 已滿，先找到 queue 中最少的 count，再反向找到對應的第一個 page 作為 victim，將其 count 歸零，移出 queue，更新 replace 次數

更新 page fault，並在 queue 最前面插入目前的 page

(c) 更新 page fault，並在 queue 最前面插入目前的 page

丁、MFU + FIFO：

- (a) 更新對應目前 page 的 count
- (b) 當 queue 沒有目前的 page 時：
若 queue 已滿，先找到 queue 中最多的 count，再反向找到對應的第一個 page 作為 victim，將其 count 歸零，移出 queue，更新 replace 次數
- (c) 更新 page fault，並在 queue 最前面插入目前的 page

戊、LFU + LRU：

- (a) 更新對應目前 page 的 count
- (b) 當 queue 沒有目前的 page 時：
若 queue 已滿，先找到 queue 中最少的 count，再反向找到對應的第一個 page 作為 victim，將其 count 歸零，移出 queue，更新 replace 次數
- (c) 否則就刪除 queue 中已經存在的、與目前相同的 page，並在 queue 最前面插入目前的 page
- (d) 更新 page fault，並在 queue 最前面插入目前的 page

己、All：依上面方法順序全部執行一遍

5. 分析不同方法之次數比較：

Method	Page fault (input1, input2)	Page replace (input1, input2)
FIFO	(9, 15)	(6, 12)
LRU	(10, 12)	(7, 9)
LFU + FIFO	(10, 13)	(7, 10)
MFU + FIFO	(9, 15)	(6, 12)
LRU + LFU	(10, 11)	(7, 8)

6. 結果與討論：

1. 實驗數據與發現：

- 若用 $(input1 + input2)/2$ 由小到大排序 fault 與 replace 都是：
 $LRU + LFU < LRU < LFU + FIFO < FIFO = MFU + FIFO$
- 可得出 FIFO 與 MFU + FIFO 最差，LRU + LFU 最佳
- 經常 access 的 page 是不應該做為 victim 的，所以 MFU + FIFO 表現最差
- FIFO 與 MFU + FIFO 具備相同的表現，但程式碼與時間複雜度較低，若要實現 MFU + FIFO 的效果不如直接使用 FIFO
- LRU + LFU 雖然表現最佳，但程式碼與時間複雜度也最高，消耗大

2. 畢雷迪反例：

在 page frames = 3, 4, 5 的限制之下，1234512512345 之表現：

<pre># out_test_fifo 1 1 1 F 2 2 21 F 3 3 321 F 4 4 432 F 5 5 521 F 6 1 521 F 7 2 214 F 8 3 352 F 9 4 435 F 10 5 435 F 11 1 521 F 12 2 214 F 13 3 352 F 14 4 435 F 15 5 435 F 16 Page Fault = 9 Page Replaces = 6 Page Frames = 3 17 -----LRU----- 18 1 1 F 19 2 21 F 20 3 321 F 21 4 432 F 22 1 143 F 23 2 214 F 24 3 321 F 25 4 432 F 26 1 143 F 27 2 214 F 28 3 321 F 29 4 432 F 30 1 143 F 31 2 214 F 32 3 321 F 33 4 432 F 34 1 143 F 35 2 214 F 36 3 321 F 37 4 432 F 38 1 143 F 39 2 214 F 40 3 321 F 41 4 432 F 42 1 143 F 43 2 214 F 44 Page Fault = 10 Page Replaces = 7 Page Frames = 3</pre>	<pre># out_test_fifo 1 1 1 F 2 2 21 F 3 3 321 F 4 4 4321 F 5 5 54321 F 6 1 54321 F 7 2 2154 F 8 3 3215 F 9 4 4321 F 10 5 5432 F 11 1 54321 F 12 2 2154 F 13 3 3215 F 14 4 4321 F 15 5 5432 F 16 Page Fault = 10 Page Replaces = 6 Page Frames = 4 17 -----LRU----- 18 1 1 F 19 2 21 F 20 3 321 F 21 4 4321 F 22 1 1432 F 23 2 2143 F 24 3 3214 F 25 4 4321 F 26 1 1524 F 27 2 2154 F 28 3 3215 F 29 4 4321 F 30 1 1524 F 31 2 2154 F 32 3 3215 F 33 4 4321 F 34 1 1524 F 35 2 2154 F 36 3 3215 F 37 4 4321 F 38 1 1524 F 39 2 2154 F 40 3 3215 F 41 4 4321 F 42 1 1524 F 43 2 2154 F 44 Page Fault = 8 Page Replaces = 4 Page Frames = 4</pre>	<pre># out_test_fifo 1 1 1 F 2 2 21 F 3 3 321 F 4 4 4321 F 5 5 54321 F 6 1 54321 F 7 2 54321 F 8 3 54321 F 9 4 54321 F 10 5 54321 F 11 1 54321 F 12 2 54321 F 13 3 54321 F 14 4 54321 F 15 5 54321 F 16 Page Fault = 5 Page Replaces = 0 Page Frames = 5 17 -----LRU----- 18 1 1 F 19 2 21 F 20 3 321 F 21 4 4321 F 22 1 1432 F 23 2 2143 F 24 3 3214 F 25 4 4321 F 26 1 1524 F 27 2 2154 F 28 3 3215 F 29 4 4321 F 30 1 1524 F 31 2 2154 F 32 3 3215 F 33 4 4321 F 34 1 1524 F 35 2 2154 F 36 3 3215 F 37 4 4321 F 38 1 1524 F 39 2 2154 F 40 3 3215 F 41 4 4321 F 42 1 1524 F 43 2 2154 F 44 Page Fault = 5 Page Replaces = 0 Page Frames = 5</pre>
<pre># out_test_fifo 45 -----Least Frequently Used Page Replacement----- 46 1 1 F 47 2 21 F 48 3 321 F 49 4 432 F 50 1 143 F 51 2 214 F 52 3 321 F 53 4 432 F 54 1 143 F 55 2 214 F 56 3 321 F 57 4 432 F 58 1 143 F 59 2 214 F 60 3 321 F 61 4 432 F 62 1 143 F 63 2 214 F 64 3 321 F 65 4 432 F 66 1 143 F 67 2 214 F 68 3 321 F 69 4 432 F 70 1 143 F 71 2 214 F 72 3 321 F 73 4 432 F 74 Page Fault = 10 Page Replaces = 7 Page Frames = 3</pre>	<pre># out_test_fifo 45 -----Least Frequently Used Page Replacement----- 46 1 1 F 47 2 21 F 48 3 321 F 49 4 4321 F 50 1 4321 F 51 2 4321 F 52 3 4321 F 53 4 4321 F 54 1 4321 F 55 2 4321 F 56 3 4321 F 57 4 4321 F 58 1 4321 F 59 2 4321 F 60 3 4321 F 61 4 4321 F 62 1 4321 F 63 2 4321 F 64 3 4321 F 65 4 4321 F 66 1 4321 F 67 2 4321 F 68 3 4321 F 69 4 4321 F 70 1 4321 F 71 2 4321 F 72 3 4321 F 73 4 4321 F 74 Page Fault = 8 Page Replaces = 4 Page Frames = 4</pre>	<pre># out_test_fifo 45 -----Least Frequently Used Page Replacement----- 46 1 1 F 47 2 21 F 48 3 321 F 49 4 4321 F 50 1 4321 F 51 2 4321 F 52 3 4321 F 53 4 4321 F 54 1 4321 F 55 2 4321 F 56 3 4321 F 57 4 4321 F 58 1 4321 F 59 2 4321 F 60 3 4321 F 61 4 4321 F 62 1 4321 F 63 2 4321 F 64 3 4321 F 65 4 4321 F 66 1 4321 F 67 2 4321 F 68 3 4321 F 69 4 4321 F 70 1 4321 F 71 2 4321 F 72 3 4321 F 73 4 4321 F 74 Page Fault = 5 Page Replaces = 0 Page Frames = 5</pre>
<pre># out_test_fifo 65 -----Most Frequently Used Page Replacement----- 66 1 1 F 67 2 21 F 68 3 321 F 69 4 432 F 70 1 143 F 71 2 214 F 72 3 321 F 73 4 432 F 74 1 521 F 75 2 521 F 76 3 352 F 77 4 435 F 78 5 435 F 79 Page Fault = 9 Page Replaces = 6 Page Frames = 3 80 -----Least Frequently Used LRU Page Replacement----- 81 1 1 F 82 2 21 F 83 3 321 F 84 4 432 F 85 1 143 F 86 2 214 F 87 3 321 F 88 4 432 F 89 1 152 F 90 2 215 F 91 3 321 F 92 4 432 F 93 5 521 F 94 Page Fault = 10 Page Replaces = 7 Page Frames = 3</pre>	<pre># out_test_fifo 65 -----Most Frequently Used Page Replacement----- 66 1 1 F 67 2 21 F 68 3 321 F 69 4 4321 F 70 1 4321 F 71 2 4321 F 72 3 4321 F 73 4 4321 F 74 1 54321 F 75 2 54321 F 76 3 3215 F 77 4 4321 F 78 5 5432 F 79 Page Fault = 10 Page Replaces = 6 Page Frames = 4 80 -----Least Frequently Used LRU Page Replacement----- 81 1 1 F 82 2 21 F 83 3 321 F 84 4 4321 F 85 1 1432 F 86 2 2143 F 87 3 3214 F 88 4 4321 F 89 1 1524 F 90 2 2154 F 91 3 3215 F 92 4 4321 F 93 5 54321 F 94 Page Fault = 8 Page Replaces = 4 Page Frames = 4</pre>	<pre># out_test_fifo 65 -----Most Frequently Used Page Replacement----- 66 1 1 F 67 2 21 F 68 3 321 F 69 4 4321 F 70 1 4321 F 71 2 4321 F 72 3 4321 F 73 4 4321 F 74 1 54321 F 75 2 54321 F 76 3 3215 F 77 4 4321 F 78 5 54321 F 79 Page Fault = 5 Page Replaces = 0 Page Frames = 5 80 -----Least Frequently Used LRU Page Replacement----- 81 1 1 F 82 2 21 F 83 3 321 F 84 4 4321 F 85 1 1432 F 86 2 2143 F 87 3 3214 F 88 4 4321 F 89 1 1524 F 90 2 2154 F 91 3 3215 F 92 4 4321 F 93 5 54321 F 94 Page Fault = 5 Page Replaces = 0 Page Frames = 5</pre>

- 觀察到 FIFO 與 MFU + FIFO 在 page frames = 3, 4 時 page fault 次數由 9 增加至 10，且 page replace 次數沒有減少，可得知這兩個表現最差的演算法會有機率導致 Belady's Anomaly 發生
- 由在 page frames = 5 時發現此兩種演算法並沒有繼續增加 page fault 次數，page replace 次數也有減少，可得知 Belady's Anomaly 為特例情況