CS305 作業系統概論 Prog. #4 Page Replacement 說明報告.docx

作者 ：1043335 賴詩雨

版本	修改日期	說明	修改者
1.0.0.0	2017-06-12	初稿	賴詩雨
1.0.0.1	2017-06-13	修改	賴詩雨

目錄

第1章　程式完成部分

a. 4 種演算法 (Optimal、FIFO、LRU、LFU)

b. 加分項目 : Clock algorithm

第2章　設計理念

2.1 Optimal Algorithm

- **變數介紹**

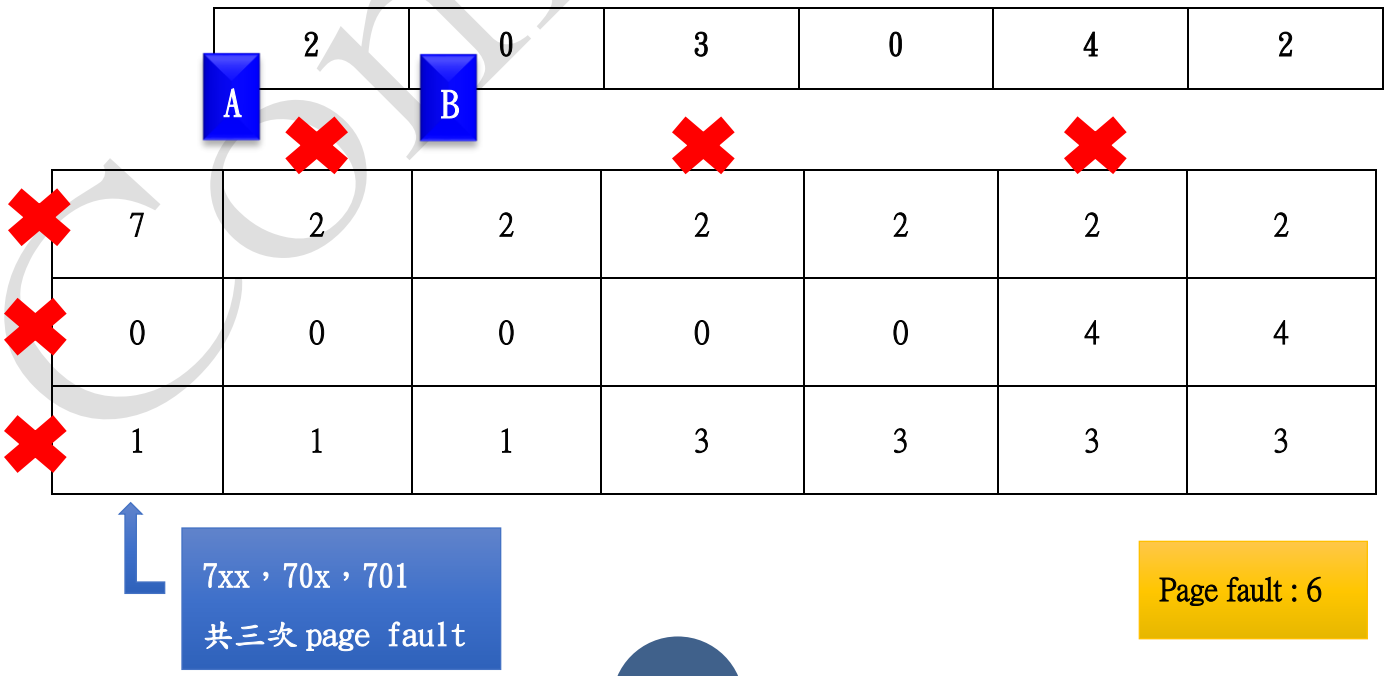
```
vector <int> frame;                  // Frame 的數量
vector <int> first(Fsize, -1);       // 計算 frame 中進來數字的陣列編號(編號小代表越早進來)

int fault = Fsize;                   // page fault 的數量
int point = 0, Fcount = 0;           // 目前處理到數字的位置, 目前 frame 要換的位置
```

- **實作方法**

一開始等 frame 全部先放滿，再來判斷接下來要使用的數字是否已存在 frame 中。

1. 若 frame 中無，則利用 for 迴圈去計算未來看的到的數字中有幾個存在 (count)
   A. Frame 中的數字都存在 (count == Fsize)，則最遙遠的那個數字極為要替換掉的
   B. Frame 中有數字不存在，則替換掉不存在當中最早進入的 (FIFO)
2. 若 frame 中已存在此數字，則直接印出即可

- **簡單示意圖**



- **流程簡介（ 以上述例子為例 ）**

以上述例子為例

1. 藍框 A 要判斷 2 是否再 frame 中，發現沒有，因此進入 for 迴圈，計算 count
2. 得到 count = 1（ 未來只能看到 0 一個數字）
3. 7 和 1 兩數當中，7 較早進入，因此替換掉 7
4. 可得 2 0 1
5. 藍框 B 要判斷 0 是否再 frame 中，發現有，因此直接印出
6. 其餘的依照上述方法完成

2.2 FIFO Algorithm

- **變數介紹**

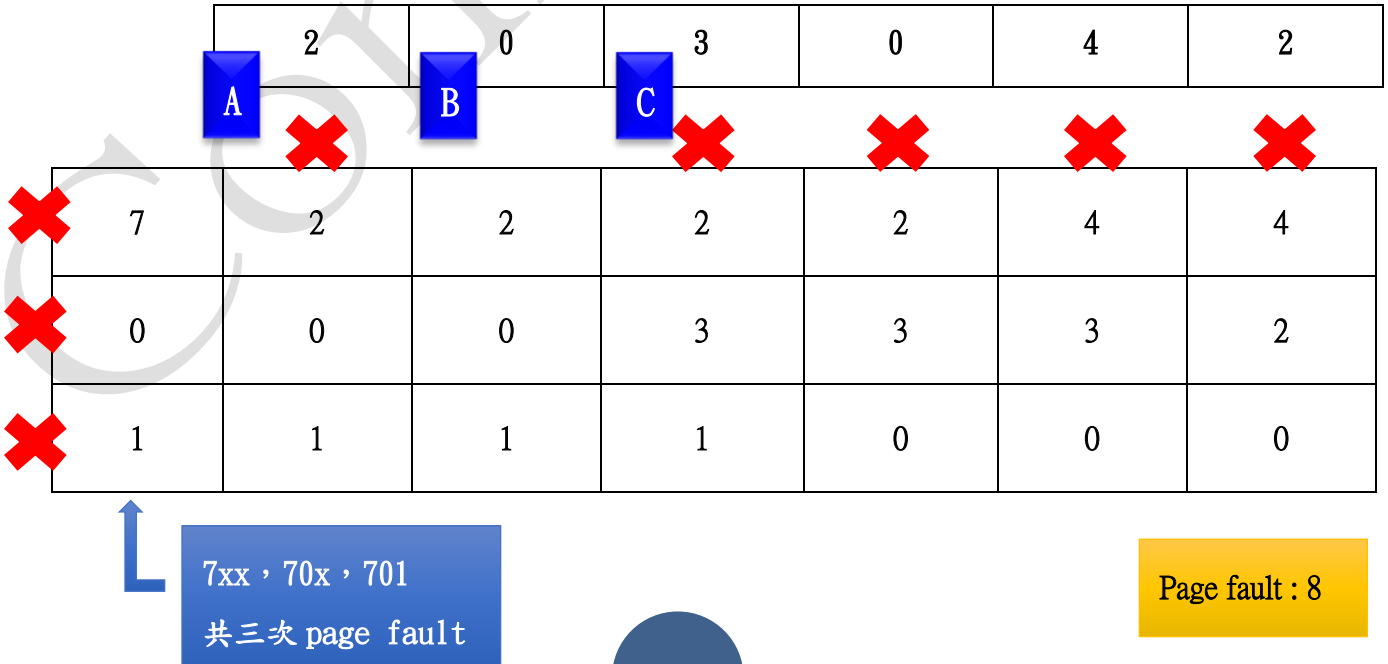
```
vector <int> frame;          // Frame 的數量
int fault = Fsize;           // page fault 的數量
int point = 0, Fcount = 0;   // 目前處理到數字的位置, 目前 frame 要換的位置
int fifo = 0;                // 目前要替換掉的 frame 位置
```

- **實作方法**

一開始等 frame 全部先放滿，再來判斷接下來要使用的數字是否已存在 frame 中。

1. 若 frame 中無，將先進來的替換掉（ fifo 指到的位置 ）
2. 若 frame 中已存在此數字，則直接印出即可

- **簡單示意圖**



- **流程簡介（ 以上述例子為例 ）**

1. 藍框 A 要判斷 2 是否再 frame 中，發現沒有，因此將最早進入的數字替換掉（ 7 ）

NOTE: int fifo 一開始在 7 的位置，被替換掉後會往下一格，到底再回第一格位置

2. Fifo 從 7 的位置移到 0 的位置
3. 藍框 B 要判斷 0 是否再 frame 中，發現有，因此直接印出，且 fifo 不變
4. 藍框 C 要判斷 3 是否再 frame 中，發現沒有，因此將 3 替換到 fifo 在的位置（ 0 ）
5. 其餘的依照上述方法完成

2.3 LRU Algorithm

- **變數介紹**

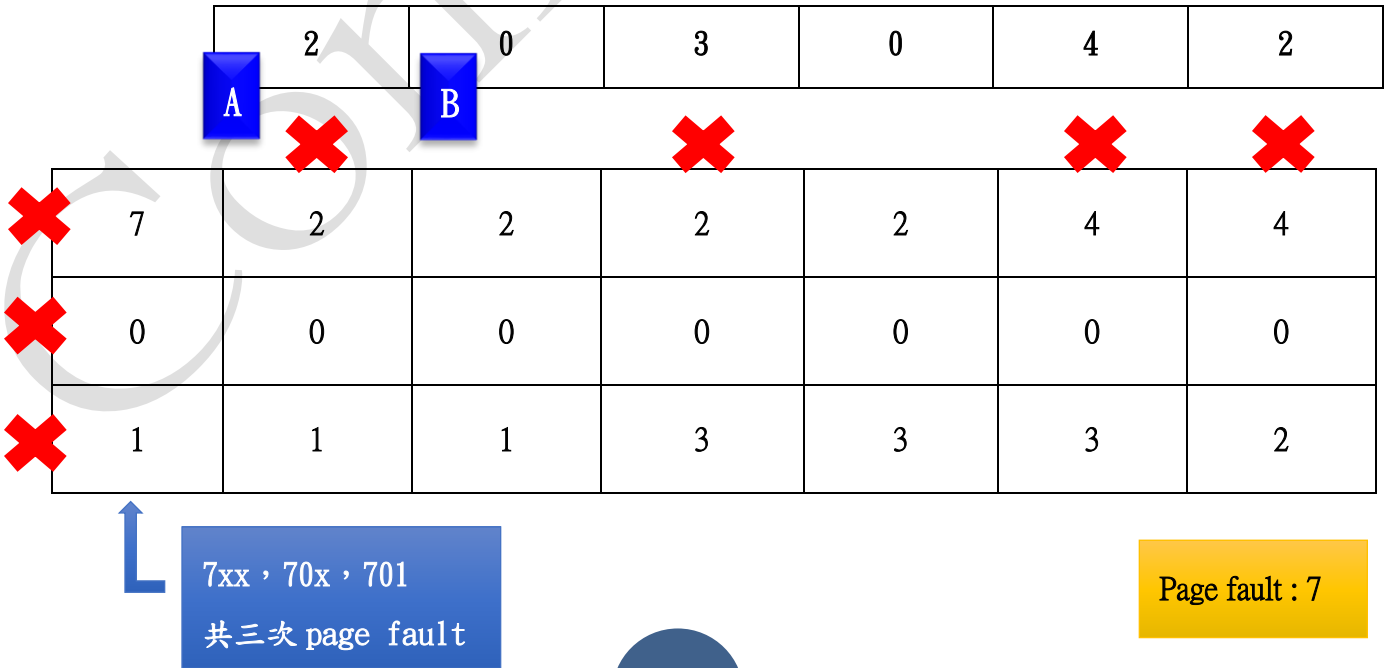
`vector <int> frame;`	`// Frame 的數量`
`int fault = Fsize;`	`// page fault 的數量`
`int point = 0, Fcount = 0;`	`// 目前處理到數字的位置, 目前 frame 要換的位置`
`vector <bool> check(Fsize, 0);`	`// 往回找，先找到的標示為 1，最後變 1 便替換掉`

- **實作方法**

一開始等 frame 全部先放滿，再來判斷接下來要使用的數字是否已存在 frame 中。

1. 若 frame 中無，則利用 for 迴圈去計算過去的數字誰最晚進就先標成 1（ check ），最早進的數字會最晚被標成 1，因此就是要被替換的
2. 若 frame 中已存在此數字，則直接印出即可

- **簡單示意圖**



- **流程簡介 （ 以上述例子為例 ）**

1. 藍框 A 要判斷 2 是否再 frame 中，發現沒有，因此進入 for 迴圈往前找，先將 1 的 check 標成 1，再將 0 的 check 標成 1，最後才將 7 的 check 標成 1
2. 因此 7 為要替換掉的
3. 可得 2 0 1
4. 藍框 B 要判斷 0 是否再 frame 中，發現有，因此直接印出
5. 其餘的依照上述方法完成

2.4 LFU Algorithm

- **變數介紹**

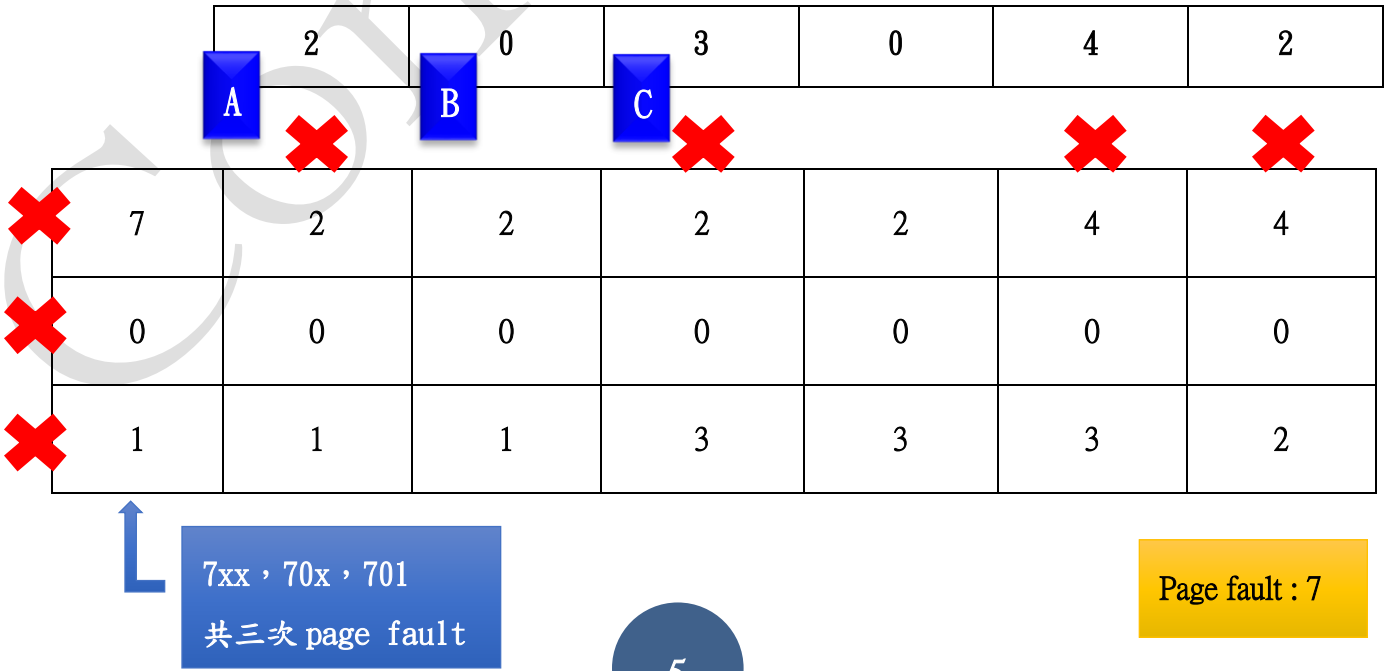
```
vector <int> frame;          // Frame 的數量
vector <int> first(Fsize, -1); // 數字進來時的陣列編號，越小代表越早
vector <int> count(Fsize, 1);  // frame 中數字目前被使用的次數
int fault = Fsize;           // page fault 的數量
int point = 0, Fcount = 0, Min; // 處理到數字,frame 要換的位置,count 中最小數
```

- **實作方法**

一開始等 frame 全部先放滿，再來判斷接下來要使用的數字是否已存在 frame 中。

1. 若 frame 中無，則利用 for 迴圈去找出 count 最少的，再去比較誰較早進來，較早且使用較少的被替換掉
2. 若 frame 中已存在此數字，將 frame 中此數字的 count 加一，再直接印出即可

- **簡單示意圖**



- **流程簡介 （ 以上述例子為例 ）**

1. 藍框 A 要判斷 2 是否再 frame 中，發現沒有，去找出 count 最少的，發現三個數字的 count 都一樣，因此要再找最早進來的，發現是 7，因此 7 為要替換掉的，可得 2 0 1
2. 藍框 B 要判斷 0 是否再 frame 中，發現有，因此將 0 的 count 加一再直接印出
3. 藍框 C 要判斷 3 是否再 frame 中，發現沒有，比對 count，發現 2 1 的次數一樣，因此從兩數當中找最早的，發現是 1，替換掉 1 後為 2 0 3
4. 其餘的依照上述方法完成

2.5 Clock Algorithm

- **變數介紹**

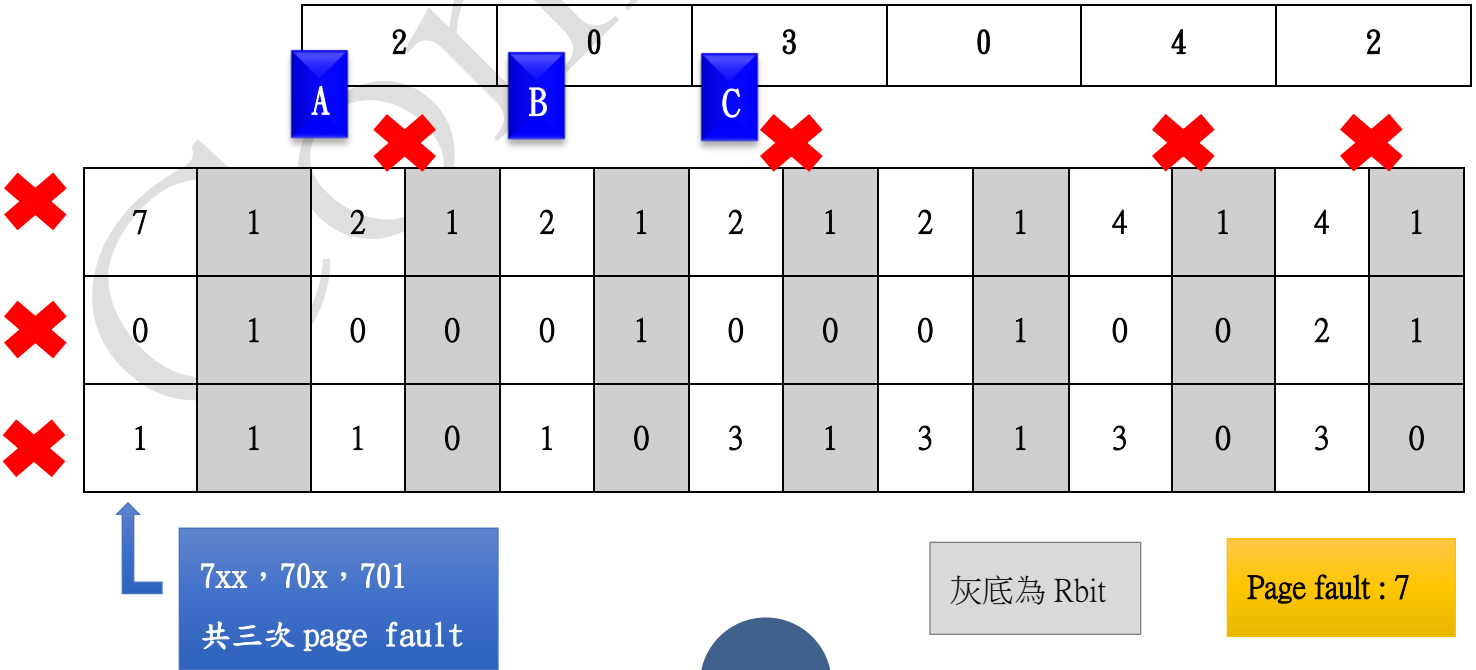
```
vector <int> frame;          // Frame 的數量
vector <int> Rbit(Fsize, 1); // 記錄第二次機會是否還在
int fault = Fsize;           // page fault 的數量
int point = 0, Fcount = 0;   // 目前處理到數字的位置, 目前 frame 要換的位置
int fifo = 0;                // 目前要替換掉的 frame 位置
```

- **實作方法**

一開始等 frame 全部先放滿，再來判斷接下來要使用的數字是否已存在 frame 中。

1. 若 frame 中無，則利用 while 迴圈去檢查第二次機會是否還在，若還有機會就把機會扣掉，若沒有機會了就是要替換掉的，利用 fifo 代表目前要被檢查的 frame
2. 若 frame 中已存在此數字，將 frame 中此數字的 Rbit 設成一，再直接印出即可

- **簡單示意圖**



- **流程簡介 （ 以上述例子為例 ）**

1. 藍框 A 要判斷 2 是否再 frame 中，發現沒有，fifo 依序檢查 7 0 1 發現他們都有一次機會，因次扣掉他們的機會再次回到 7
2. 發現他沒有機會了，所以將 7 替換掉，而替換進來的 2 有一次機會，此時 fifo 到 0 那格
3. 藍框 B 要判斷 0 是否再 frame 中，發現有，因此將 0 的 Rbit 設成 1，代表他現在有了一次機會
4. 藍框 C 要判斷 3 是否再 frame 中，發現沒有，因此從 fifo 所在那格（ 0 ）開始檢查機會，0 有一次機會，所以扣掉，再往下發現 1 沒機會了，因此將他替換掉，變成 2 0 3，且 3 的機會有一次
5. 其餘的依照上述方法完成

第3章　程式如何編譯

1. 有附上 Makefile 檔提供操作

   OR

2. 在命令列輸入　　：g++ 1043335_04.cpp -o 1043335_04
3. 編譯成功後，輸入 ：./1043335_04 replace.cfg

NOTE ： replace.cfg 可改成其他檔名，可在命令列改也可以從 Makefile 中更改

第4章　如何操作

1. 將 Makefile 中 replace.cfg 改成其他檔名，可換不同的測資
2. make main　=> 編譯
3. make run　=> 執行
4. make start => 編譯 & 執行
5. make clean => 清除編譯、執行後產生的檔案
6. make all　=> 清除檔案後編譯執行
7. 或是可以直接在命令列輸入 **第 3 章** 提供的指令