FACULTY OF ENGINEERING

AIN SHAMS UNIVERSITY

CSE 223 : OPERATING SYSTEMS

PAGE REPLACEMENT ALGORITHMS

DECEMBER, 2018

AIN SHAMS UNIVERSITY – FACULTY OF ENGINEERING

1 Al-Sarayyat st, Abbassiya, Cairo 11517, Egypt

Page Replacement Algorithms

Submitted by:

Abdelrahman Ibrahim ELGhamry
Ghamry98@hotmail.com
16P3043@eng.asu.edu.eg

Hossam ELDin Khaled Mohmed
hossampen97@gmail.com
16P3025@eng.asu.edu.eg

Abdelrahman Amr Issawi
aid-issawi@hotmail.com
16P6001@eng.asu.edu.eg

Submitted to:

Prof. Dr. Gamal Abdelshafy

December, 2018

A report for Operating Systems Course codded CSE223 with the requirements of Ain Shams University

Table of Contents

1.0 BRIEF DESCRIPTION 3

2.0 IMPLEMENTATION 3

3.0 TEST CASES 6

3.1 Test Case (1) : 6

3.2 Test Case (2) : 7

3.3 Test Case (3) : 8

1.0 BRIEF DESCRIPTION

- This project is a console application providing the implementation of the FIFO, LRU, LFU, Second Chance, Enhance Second-Chance, and Optimal page replacement algorithms.
- The user enters a number of frames, that number of page frames can vary from 1 to 20.
- The system generates a random page-reference string where page numbers range from 0 to 99, the length of the reference string is specified by the user.
- The system applies the random page-reference string to each algorithm and record the number of page faults incurred by each algorithm.
- The system prints a 2D matrix contains each iteration of the program (Memory Frames).
- Finally, the system prints the number of faults of each algorithm.

2.0 IMPLEMENTATION

Methods:

1. **firstInFirstOut()**
   - Void method that implements FIFO algorithm on the randomly generated reference string.
   - Firstly, it loops on the reference string checking whether the specified page is found in the frames or not.
   - In case of page fault, it replaces the page which came first to the memory.
   - Finally, it prints the memory during the process and the number of faults using the method printOneColumn(int[][] mem, int faults, String s).

2. **leastRecentlyUsed()**
    - Void method that implements LRU algorithm on the randomly generated reference string.
    - It records the time when a page is allocated to a frame in memory, and updates it when a page which resides in memory is referenced again.
    - In case of page fault, it calls getMinIndex() that finds the page which has the longest time without being referenced in memory and returns its index, and then replaces the frame with a new page, increments the faults count and update the memory.
    - Finally, it prints the memory during the process and the number of faults using the method printOneColumn(int[][] mem, int faults, String s).

3. **leastFrequentlyUsed()**
    - Void method that implements LFU algorithm on the randomly generated reference string.
    - Firstly, there is a class defined for a page in LFU algorithm called, LFUPage that contains the page number, the number of page demands to that page, the time at which the page is allocated in the memory and the its place in the memory.
    - In case of a page fault, it calls getReplacementIndex(int[][] mem) method that is defined in LFUPage class, this method finds the page with the least frequency in the memory, if it finds more than a page with the same minimum frequency, it selects from this set the page which has the longest time without being referenced in the memory. After that, it sets the frequency of the replaced page to zero.
    - Finally, it prints the memory during the process and the number of faults using the method printOneColumn(int[][] mem, int faults, String s).

4. **secondChance()**
    - Void method that implements second chance algorithm on the randomly generated reference string.
    - Firstly, it loops on the reference string checking whether the specified page is found in the frames or not.
    - In case of page hit, the specified page valid bit is set to one, and the pointer value don't change.
    - In case of page fault, the pointer starts to loop among the frames searching for a frame with a zero valid bit. While looping, if found a frame with a valid bit equals to one, it is set to zero and the pointer continue looping till finding a frame with a zero bit.
    - Finally, it prints the memory during the process, the valid bits during the process, and the number of faults using the method printTwoColumns(int[][] memory, int[][] validBit, int faults, String s).

5. enhancedSecondChance()

- Void method that implements Enhanced Second-Chance algorithm on the randomly generated reference string and modify bits.
- The page enter the memory frames with r-bit initially equals to 1.
- Firstly, it loops on the reference string checking whether the specified page is found in the frames or not.
- In case of page hit, we assume that the specified page r-bit remains the same, and the pointer value don't change.
- In case of page fault, the algorithm will:
    a. Loop through the frames looking for <0,0>. If one is found, use that page.
    b. Loop through the frames looking for <0,1>. And set the r-bit to zero for all bypassed frames.
    c. If step (b) failed, all r-bits will now be zero and repetition of steps (a) and (b) are guaranteed to find a frame for replacement.
- Finally, it prints the memory during the process, the valid bits during the process, and the number of faults using the method printThreeColumns(int[][] memory,int[][] refBit,int[][] modifyBit, int faults,String s).

6. optimal()

- Void method that implements Optimal algorithm on the randomly generated reference string.
- The algorithm works on replacing the page that will not be used for longest period of time.
- Firstly, it loops on the reference string checking whether the specified page is found in the frames or not.
- In case of a page fault, it loops over the memory frames and for each page found there, it loops over the remaining reference string to find the next index of the same page and add it to inFramesNextIndex[] array, the algorithm then searches inFramesNextIndex[] array for the max index and it will refer to the page that the algorithm will replace.
- Finally, it prints the memory during the process and the number of faults using the method printOneColumn(int[][] mem, int faults, String s).
- Optimal algorithm is perfect, but not possible in practice as operating system cannot know future requests.
- The use of Optimal algorithm is to set up a benchmark so that other replacement algorithms can be analyzed against it.

3.0 TEST CASES

3.1 Test Case (1) :

```
Enter Number of Frames:
3
Enter the length of Reference String:
20

-----------------------------------------------------
Reference String:
   7    0    1    2    0    3    0    4    2    3    0    3    2    1    2    0    1    7    0    1

FIFO Algorithm:
   7    7    7    2    2    2    2    4    4    4    0    0    0    0    0    0    0    7    7    7
  -1    0    0    0    0    3    3    3    2    2    2    2    2    1    1    1    1    1    0    0
  -1   -1    1    1    1    1    0    0    0    3    3    3    3    3    2    2    2    2    2    1
The number of Faults: 15

-----------------------------------------------------
Reference String:
   7    0    1    2    0    3    0    4    2    3    0    3    2    1    2    0    1    7    0    1

LRU Algorithm:
   7    7    7    2    2    2    2    4    4    4    0    0    0    1    1    1    1    1    1    1
  -1    0    0    0    0    0    0    0    0    3    3    3    3    3    3    0    0    0    0    0
  -1   -1    1    1    1    3    3    3    2    2    2    2    2    2    2    2    2    7    7    7
The number of Faults: 12

-----------------------------------------------------
Reference String:
   7    0    1    2    0    3    0    4    2    3    0    3    2    1    2    0    1    7    0    1

LFU Algorithm:
   7    7    7    2    2    2    2    4    4    3    3    3    3    1    1    1    1    7    7    1
  -1    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  -1   -1    1    1    1    3    3    3    2    2    2    2    2    2    2    2    2    2    2    2
The number of Faults: 11

-----------------------------------------------------
Reference String:
    7     0     1     2     0     3     0     4     2     3     0     3     2     1     2     0     1     7     0     1

Second Chance Algorithm:
   7 1   7 1   7 1   2 1   2 1   2 1   2 1   4 1   4 1   4 1   4 0   3 1   3 1   3 0   3 0   0 1   0 1   0 0   0 1   0 1
  -1 0   0 1   0 1   0 0   0 1   0 0   0 1   0 0   2 1   2 1   2 0   2 0   2 1   1 1   1 1   1 1   1 1   7 1   7 1   7 1
  -1 0  -1 0   1 1   1 0   1 0   3 1   3 1   3 0   3 0   3 1   0 1   0 1   0 1   0 0   2 1   2 1   2 1   2 0   2 0   1 1
The number of Faults: 14

-----------------------------------------------------
Reference String:
     7       0       1       2       0       3       0       4       2       3       0       3       2       1       2       0       1       7       0       1

Enhanced Second Chance Algorithm:
   7 1 0   7 1 0   7 1 0   2 1 0   2 1 0   2 1 0   2 1 0   4 1 1   4 1 1   4 1 1   4 0 1   4 0 1   2 1 1   2 1 1   2 1 1   2 1 1   2 1 1   7 1 1   7 1 1   7 1 1
  -1 0 0   0 1 0   0 1 0   0 0 0   0 0 0   3 1 1   3 1 1   3 0 1   3 0 1   3 0 1   0 1 0   0 1 0   0 1 0   1 1 1   1 1 1   1 1 1   1 1 1   1 0 1   1 0 1   1 0 1
  -1 0 0  -1 0 0   1 1 1   1 0 1   1 0 1   1 0 1   0 1 0   0 0 0   2 1 0   2 1 0   2 1 0   3 1 1   3 1 1   3 0 1   3 0 1   0 1 1   0 1 1   0 0 1   0 0 1   0 0 1
The number of Faults: 14

-----------------------------------------------------
Reference String:
   7    0    1    2    0    3    0    4    2    3    0    3    2    1    2    0    1    7    0    1

Optimal Algorithm:
   7    7    7    2    2    2    2    2    2    2    2    2    2    2    2    2    2    7    7    7
  -1    0    0    0    0    0    0    4    4    4    0    0    0    0    0    0    0    0    0    0
  -1   -1    1    1    1    3    3    3    3    3    3    3    3    1    1    1    1    1    1    1
The number of Faults: 9
BUILD SUCCESSFUL (total time: 1 minute 9 seconds)
```

3.2 Test Case (2) :

```
Enter Number of Frames:
3
Enter the length of Reference String:
15

--------------------------------------------------
Reference String:
  97   56   61   78   16   31   78   56    9    6   78   39   68   21   84

FIFO Algorithm:
  97   97   97   78   78   78   78   56   56   56   78   78   78   21   21
  -1   56   56   56   16   16   16   16    9    9    9   39   39   39   84
  -1   -1   61   61   61   31   31   31   31    6    6    6   68   68   68
The number of Faults: 14

--------------------------------------------------
Reference String:
  97   56   61   78   16   31   78   56    9    6   78   39   68   21   84

LRU Algorithm:
  97   97   97   78   78   78   78   78   78    6    6    6   68   68   68
  -1   56   56   56   16   16   16   56   56   56   78   78   78   21   21
  -1   -1   61   61   61   31   31   31    9    9    9   39   39   39   84
The number of Faults: 14

--------------------------------------------------
Reference String:
  97   56   61   78   16   31   78   56    9    6   78   39   68   21   84

LFU Algorithm:
  97   97   97   78   78   78   78   78   78   78   78   78   78   78   78
  -1   56   56   56   16   16   16   56   56    6    6    6   68   68   84
  -1   -1   61   61   61   31   31   31    9    9    9   39   39   21   21
The number of Faults: 13

--------------------------------------------------
Reference String:
   97     56     61     78     16     31     78     56      9      6     78     39     68     21     84

Second Chance Algorithm:
  97 1   97 1   97 1   78 1   78 1   78 1   78 1   56 1   56 1   56 1   78 1   78 1   78 1   21 1   21 1
  -1 0   56 1   56 1   56 0   16 1   16 1   16 1   16 0    9 1    9 1    9 0   39 1   39 1   39 0   84 1
  -1 0   -1 0   61 1   61 0   61 0   31 1   31 1   31 0   31 0    6 1    6 0    6 0   68 1   68 0   68 0
The number of Faults: 14

--------------------------------------------------
Reference String:
    97       56       61       78       16       31       78       56        9        6       78       39       68       21       84

Enhanced Second Chance Algorithm:
  97 1 0   97 1 0   97 1 0   78 1 1   78 1 1   78 0 1   78 0 1   78 0 1    9 1 1    9 1 1    9 0 1    9 0 1    9 0 1    9 0 1   84 1 1
  -1 0 0   56 1 1   56 1 1   56 0 1   56 0 1   31 1 1   31 1 1   31 1 1   31 1 1   31 0 1   78 1 0   78 1 0   78 1 0   78 1 0   78 1 0
  -1 0 0   -1 0 0   61 1 0   61 0 0   16 1 0   16 1 0   16 1 0   56 1 0   56 1 0    6 1 1    6 1 1   39 1 1   68 1 0   21 1 0   21 1 0
The number of Faults: 14

--------------------------------------------------
Reference String:
  97   56   61   78   16   31   78   56    9    6   78   39   68   21   84

Optimal Algorithm:
  97   97   97   78   78   78   78   78   78   78   78   39   68   21   84
  -1   56   56   56   56   56   56   56    9    6    6    6    6    6    6
  -1   -1   61   61   16   31   31   31   31   31   31   31   31   31   31
The number of Faults: 12
BUILD SUCCESSFUL (total time: 2 seconds)
```

3.3 Test Case (3) :

```
Enter Number of Frames:
4
Enter the length of Reference String:
17

--------------------------------------------------
Reference String:
   0    1    3    6    2    4    5    2    5    0    3    1    2    5    4    1    0

FIFO Algorithm:
   0    0    0    0    2    2    2    2    2    2    3    3    3    3    4    4    4
  -1    1    1    1    1    4    4    4    4    4    4    1    1    1    1    1    0
  -1   -1    3    3    3    3    5    5    5    5    5    5    2    2    2    2    2
  -1   -1   -1    6    6    6    6    6    0    0    0    0    5    5    5    5
The number of Faults: 14

--------------------------------------------------
Reference String:
   0    1    3    6    2    4    5    2    5    0    3    1    2    5    4    1    0

LRU Algorithm:
   0    0    0    0    2    2    2    2    2    2    2    1    1    1    1    1    1
  -1    1    1    1    1    4    4    4    4    4    3    3    3    3    4    4    4
  -1   -1    3    3    3    3    5    5    5    5    5    5    2    2    2    2    0
  -1   -1   -1    6    6    6    6    6    0    0    0    0    5    5    5    5
The number of Faults: 14

--------------------------------------------------
Reference String:
   0    1    3    6    2    4    5    2    5    0    3    1    2    5    4    1    0

LFU Algorithm:
   0    0    0    0    2    2    2    2    2    2    2    2    2    2    2    2    2
  -1    1    1    1    1    4    4    4    4    4    3    3    3    3    4    4    0
  -1   -1    3    3    3    3    5    5    5    5    5    5    5    5    5    5    5
  -1   -1   -1    6    6    6    6    6    6    0    0    1    1    1    1    1    1
The number of Faults: 12

--------------------------------------------------
Reference String:
    0      1      3      6      2      4      5      2      5      0      3      1      2      5      4      1      0

Second Chance Algorithm:
   0 1    0 1    0 1    0 1    2 1    2 1    2 1    2 1    2 1    2 1    3 1    3 1    3 1    3 1    4 1    4 1    4 1
  -1 0    1 1    1 1    1 1    1 0    4 1    4 1    4 1    4 1    4 1    4 0    1 1    1 1    1 1    1 0    1 1    1 0
  -1 0   -1 0    3 1    3 1    3 0    3 0    5 1    5 1    5 1    5 1    5 0    5 0    2 1    2 1    2 0    2 0    0 1
  -1 0   -1 0   -1 0    6 1    6 0    6 0    6 0    6 0    6 0    0 1    0 0    0 0    0 0    5 1    5 0    5 0    5 0
The number of Faults: 14

--------------------------------------------------
Reference String:
     0        1        3        6        2        4        5        2        5        0        3        1        2        5        4        1        0

Enhanced Second Chance Algorithm:
   0 1 0    0 1 0    0 1 0    0 1 0    2 1 1    2 1 1    2 1 1    2 1 1    2 1 1    2 0 1    2 0 1    2 0 1    2 0 1    5 1 1    5 1 1    5 1 1    5 1 1
  -1 0 0    1 1 1    1 1 1    1 1 1    1 0 1    1 0 1    1 0 1    1 0 1    1 0 1    0 1 0    0 1 0    0 1 0    0 1 0    0 1 0    4 1 0    4 1 0    4 1 0
  -1 0 0   -1 0 0    3 1 0    3 1 0    3 0 0    4 1 0    4 1 0    4 1 0    4 1 0    4 1 0    3 1 1    3 1 1    3 1 1    3 1 1    3 0 1    3 0 1    3 0 1
  -1 0 0   -1 0 0   -1 0 0    6 1 0    6 0 0    6 0 0    5 1 0    5 1 0    5 1 0    5 1 0    5 0 0    1 1 0    1 1 0    1 1 0    1 0 0    1 0 0    0 1 1
The number of Faults: 13

--------------------------------------------------
Reference String:
   0    1    3    6    2    4    5    2    5    0    3    1    2    5    4    1    0

Optimal Algorithm:
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
  -1    1    1    1    1    4    5    5    5    5    5    5    5    5    4    4    4
  -1   -1    3    3    3    3    3    3    3    3    1    1    1    1    1    1    1
  -1   -1   -1    6    2    2    2    2    2    2    2    2    2    2    2    2    2
The number of Faults: 9
BUILD SUCCESSFUL (total time: 2 seconds)
```