AN INPUT ENHANCEMENT TECHNIQUE TO MAXIMIZE THE PERFORMANCE OF PAGE REPLACEMENT ALGORITHMS

Mahesh Kumar M R[1], Renuka Rajendra B[2]

[1]*Department of CSE, JSSATE, Bengaluru, India*
[2]*Department of CSE, JSSATE, Bengaluru, India*

Abstract

One of the main goals of the memory management is to allow multiprogramming. Several strategies are used to allocate a memory to the process needed. These strategies require the entire process should be in the memory before execution and some of these strategies suffer from fragmentation. Virtual memory does not require entire process to be in memory before execution. It loads only those pages of the process in to the memory that are needed for execution. This can be achieved using paging scheme having number of frames in the memory to accommodate for each pages. Whenever one of the page in the memory need to be replaced, one of the page replacement algorithms are used. The performance of these page replacement algorithms depends on the total number of page faults achieved. In this paper, a new algorithm called Comparison Counting FIFO (CC-FIFO) and Distribution Counting FIFO (DC-FIFO) has been proposed using input enhancement technique. Our results and calculations show that the proposed algorithm minimizes the page fault rate compared to FIFO, LRU and Optimal page replacement algorithms.

Keywords-*Page replacement; Page fault; FIFO; OPT; LRU; Comparison Counting; Distribution Counting;*

----------***----------

1. INTRODUCTION

Memory management allows multiprogramming so that operating systems keeps several processes in the memory. We need to allocate memory in the most efficient way possible. Memory can be allocated using multiple partition schemes, variable partition scheme (first fit, best fit, and worst fit) [4]. But some of these strategies suffer from fragmentation problem. We can avoid this problem using paging and segmentation memory management scheme. All these schemes require that entire process to be in the physical memory before we can execute the process.

Virtual memory does not require the entire process to be in the memory before we can execute because sometimes a user does not require the entire process in the memory. User may be interested in a few pages of the process rather than the entire process at any point of time. In such a case, bring only those pages from the disk into the memory that is demanded during the execution. If the pages of the process are not demanded, then these pages are never loaded in to the memory. This technique is called demand paging [4].

If the process tries to access a page in the memory that was not brought from the disk, then such page is called page fault. In order to identify the page fault, we will use valid / invalid bit scheme. If the bit is set as valid to one of the page, then that page is in the memory. Otherwise, the page is not valid and is currently on the disk not in the memory [4]. The page fault may occur when the desired page is not in the memory or the desired page is currently on the disk and is ready to bring it into the memory but the memory is full. At this point to allow the execution of the desired page, either bring the page from the disk into the memory or replace some of the pages that are already in the memory using one of the page replacement algorithms. Among all the page replacement algorithms, optimal page replacement is the most efficient algorithm which has less page fault rate.

In an attempt towards reducing the page fault compared to the existing algorithms, we have proposed a new algorithm CC-FIFO and DC-FIFO using input enhancement techniques. Our algorithm takes less number of page fault rate compared to FIFO, OPT and LRU replacement algorithms.

The reminder of the paper is organised as follows. Section II describes the different page replacement algorithms. Section III describes the related work done in the area of page replacement algorithm. Section IV input enhancement techniques for comparison counting and distribution counting. Section V describes the proposed algorithm. Section VI describes results and observations for few test cases. Section VII concludes with the summary.

2. PAGE REPLACEMENT ALGORITHMS

There are several page replacement algorithms that determine which page to replace to accommodate a desired page in to the memory.

2.1 First In First Out (FIFO)

This algorithm replaces the page that has been in the memory for the longest period of time. In order to do this the system must keeps track of the order in which pages enter memory [5]. This algorithm will result in more page

faults even the number of frames are increased called belady's anomaly.

2.2 Optimal or OPT

This algorithm replaces the page that will not be referenced again for the longest period of time in the future. This algorithm never suffers from belady's anoma*ly*.

2.3 Least Recently Used (LRU)

This algorithm replaces the page by considering the past behaviour of the pages in the memory. This will replace the page that has spent longest period of time in the memory being unused.

2.4 Second Chance Algorithm

This algorithm is a variation of FIFO. It checks the bit of the old page. If the bit is off, the algorithm immediately selects the page for replacement. Otherwise the algorithm turns off the bit and moves the page to the end of the FIFO queue [5].

2.5 Least Frequently Used (LFU)

This algorithm replaces the page that is least frequently used. To do this, we will count the number of references being made to each page. So finally replace the page that having smallest count.

2.6 Most Frequently Used (MFU)

This algorithm replaces the page that is most frequently used that results in largest count.

3. RELATED WORK DONE

In the recent years many researchers have came forward with their work and ideas in the area of page replacement algorithms.

- Wang hong, compared and analysed the page fault and page fault frequency of different algorithm [1].
- Pooja khulbe, shruti pant, suggested a advanced version of least recently used algorithm by reducing the overall page fault rate [8].
- Yogesh Niranjan, Shailendra Tiwari, proposed a new page replacement algorithm for proxy server [6].
- Anupam Battacharjee, Bideb Kumar Debnath, proposed a new randomized algorithm approximating random replament and least recently used scheme for page replacement in web caches [3].
- Ali Khasrozadeh, Sanaz Pashmforoush, proposed a page replacement algorithm based on LRu by considering certain parameter that can offer better improvement [7].
- Guangxia Xu, Lingling Ren, Yanbing Liu, proposed an efficient page replacement algorithm called FAPRA for NAND flas memory based storage devices [2].

4. INPUT ENHANCEMENT TECHNIQUES

The idea is to pre-process the problem's input and store the additional information obtained to accelerate solving the problem afterward [9]. The following two input enhancement methods are used for our proposed algorithm.

4.1 Comparison Counting

In this sorting, for each element in a list to be sorted, count the total number of element that are smaller than this element and store the results in a table. These numbers indicate the position of the elements in the sorted list [9].

- Consider the page reference string { 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1}. The number of occurrence of each element is {7-2_1, 0-6_2, 1-4_3, 2-4_4, 3-3_5, 4-1_6}. Subscript number is for future references.
- The number of occurrence obtained in the above step {2_1, 6_2, 4_3, 4_4, 3_5, 1_6} is an input to the comparison counting algorithm.
- Once we apply the algorithm, the number of occurrence gets sorted {1_6, 2_1, 3_5, 4_4, 4_3, 6_2} in increasing order.
- For each element in the sorted list, replace with corresponding number of occurrences of element from the first step as shown below. The new page reference string is {4, 7, 7, 3, 3, 3, 2, 2, 2, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0}. This is considered as a new page reference string in our proposed algorithm.

4.2 Distribution Counting

This algorithm sorts the n elements in the array. This algorithm is recommended when the input array elements consist of duplicate, random elements and also the elements in the array are known to come from a set. Consider the array A elements: {1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6}

- The elements in the array elements are known to come from the set S {1, 2, 3, 4, 5, 6, 7} where lower bound is 1 and upper bound is 7. Suppose if one of the elements in the set say 5 is not in the input array elements, then the input array elements are not derived from the set and hence this algorithm cannot be applied.
- After analyzing the input, we have to calculate the frequency of each element in the set from the input array and distribution values for the element in the set. For the above example the calculations are shown below.

Table 1

Array values Set S	1	2	3	4	5	6	7
Frequency	4	6	4	1	1	3	1
Distribution Values D	4	10	14	15	16	19	20

- Now process the input array elements right to left. Consider the last element in the array. In the above

example A[19] = 6. Subtract it from the lower bound that is 6 – 1 = 5. This value is used to access the element stored in D[5] = 19 and subtract it by 1 that is 19 – 1 = 18. So insert the element A[19] in the position of new array B[18]. This new position indicates the position in the sorted array B. This will be repeated for each element in the given array from right to left.

- Now the entire elements in the new array B are in sorted array. For the above array elements, the final sorted order after applying this algorithm is {1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 5, 6, 6, 6, 7}. This is considered as a new page reference string in our proposed algorithm.
- Since the array value is fixed, the time efficiency of this algorithm is linear.

5. PROPOSED ALGORITHM

The main goal of our proposed algorithm is to minimize the total number of page faults by obtaining a new page reference string from the above algorithm compared to FIFO, OPT and LRU page replacement algorithms.

5.1 Comparison Counting FIFO (CC-FIFO) Algorithm

If the reference string contains random, duplicate elements and the elements are not derived from the set, then we will apply the below algorithm.

1) Read an array of page reference string and the number of frames in the memory from the user.
2) Analyse the input carefully.
3) Apply comparison counting algorithm to sort the list (shown in section IV).
4) The elements are now in the sorted order.
5) Apply FIFO page replacement algorithm to the new page reference string obtained in the above step.
6) The number of page faults obtained using our algorithm is less than to that of other page replacement algorithm.

5.2 Distribution Counting FIFO (DC-FIFO) Algorithm

If the page reference string contains random, duplicate elements and the elements in the array are known to come from the set, then we apply the below algorithm.

1) Read array of page reference string and the number of frames in the memory from the user.
2) Analyse the input carefully.
3) Apply distribution counting algorithm to sort the entire array (shown in section IV).
4) The elements are now in the sorted order.
5) Apply FIFO page replacement algorithm to the new page reference string obtained in the above step.
6) The number of page faults obtained using our algorithm is less than to that of other page replacement algorithm.

6. RESULTS AND OBSERVATIONS

Before we implement the steps performed in the proposed method, we have taken two different cases for both comparison and distribution counting FIFO algorithm. For each case we will implement the traditional algorithm and also the proposed algorithm and calculate the page faults for each algorithm. In the next section we will implement the proposed algorithm and compare the results obtained in these two cases with our new algorithm to show the improved performance.

Case 1: Consider the page reference string
{7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1} Assume three frames in the memory. Fig. 1 to Fig. 3 show the representation of FIFO, OPT and LRU page replacement algorithm.

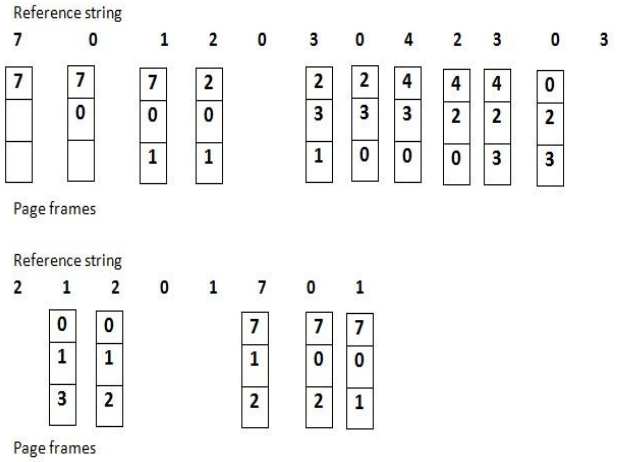

Fig 1: FIFO Page Replacement algorithm

Total number of page faults in FIFO = 15

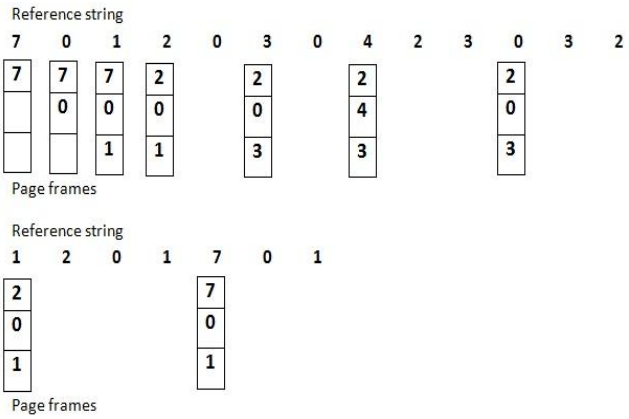

Fig 2: Optimal Page Replacement algorithm

Total number of page faults in Optimal = 9

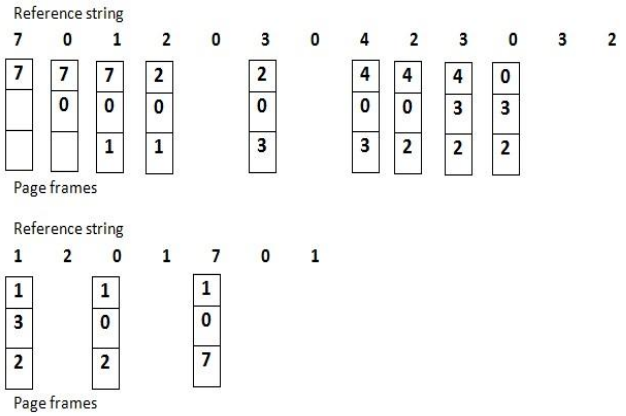

Fig 3: LRU Page Replacement algorithm

Total number of page faults in LRU = 12

Case 2: Consider the page reference string
{1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6} Assume three frames in the memory. Fig. 4 to Fig. 6 show the representation of FIFO, OPT and LRU page replacement algorithm.

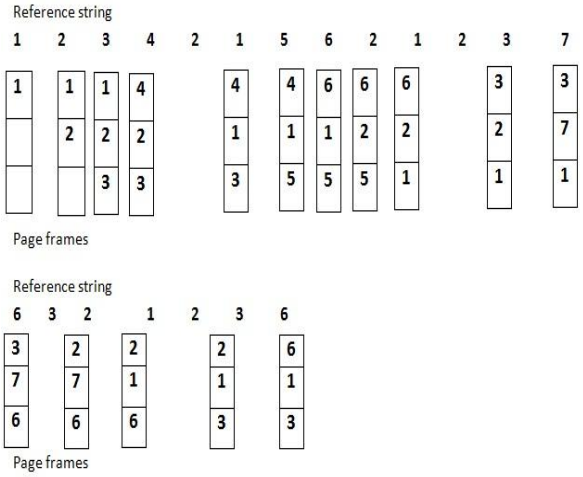

Fig 4: FIFO Page Replacement algorithm

Total number of page faults in FIFO = 16

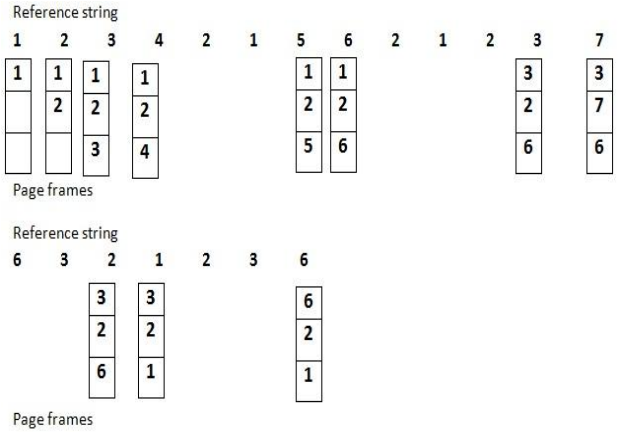

Fig 5: Optimal Page Replacement algorithm

Total number of page faults in Optimal = 11

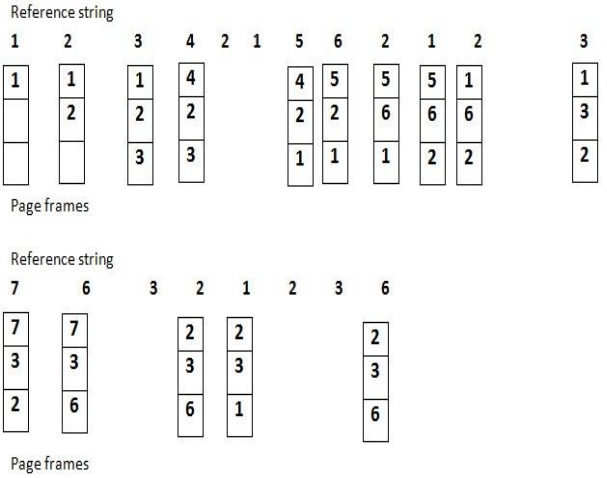

Fig 6: LRU Page Replacement algorithm

Total number of page faults in LRU = 15

6.1 Proposed Method

In this section, we will apply the proposed algorithm for the two cases to show how the pages can be accessed faster.

Case 1: Consider the page reference string
{7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1} Assume three frames in the memory.

Since the page reference string are in random order and contain duplicate elements. These elements are not known from the set. Then apply the CC-FIFO algorithm shown in section IV to obtain the new page reference string. The new page reference string after sorting is {4, 7, 7, 3, 3, 3, 2, 2, 2, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0}. Now apply the remaining steps in the algorithm to complete the task. Fig. 7 shows the representation of CC-FIFO algorithm for the new page reference string. Table II show the comparison of all the algorithms with our proposed algorithm for case 1.

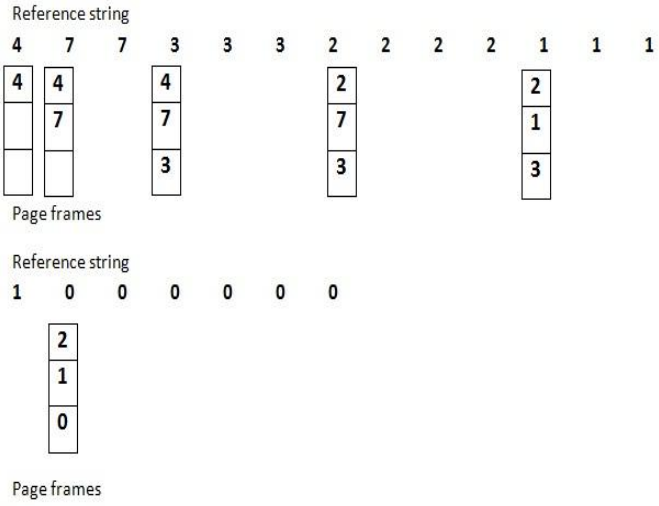

Fig 7: CC-FIFO algorithm

Total number of page faults in CC-FIFO = 6

Table 2

Algorithms	Total number of Page Faults
FIFO	15
Optimal	09
LRU	12
CC-FIFO	06

Case 2: Consider the page reference string
{1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6} Assume three frames in the memory.

Since the page reference string are in random order and contain duplicate elements. The elements are said to be come from the set {1, 2, 3, 4, 5, 6, 7}. Then apply the DC-FIFO algorithm shown in section IV to obtain the new page reference string. The new page reference string after applying algorithm is {1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 5, 6, 6, 6, 7}. Now apply the remaining steps in the algorithm to complete the task. Fig. 8 shows the representation of DC-FIFO algorithm for the new page reference string. Table III show the comparison of all the algorithms with our proposed algorithm for case 2.

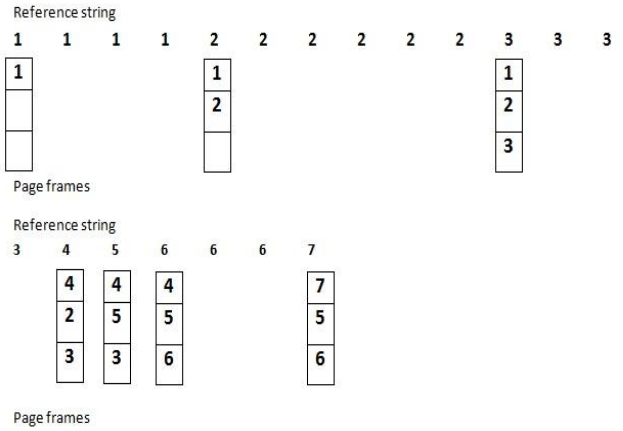

Fig 8: DC-FIFO algorithm

Total number of page faults in DC-FIFO = 7

Table 3

Algorithms	Total number of Page Faults
FIFO	16
Optimal	11
LRU	15
DC-FIFO	07

6.2 Comparison

Fig. 9 shows the comparison of total number of page faults in FIFO, Optimal and LRU with our algorithm called CC-FIFO replacement algorithm for case 1. Thus from this figure we showed that our new algorithm CC-FIFO results in reduced page faults compared to other replacement algorithms.

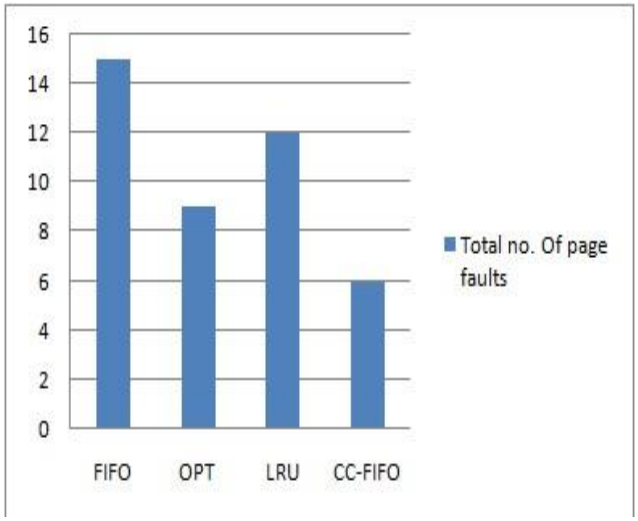

Fig 9: Comparison of page faults for case1

Fig. 10 shows the comparison of total number of page faults in FIFO, Optimal and LRU with our algorithm called DC-FIFO replacement algorithm for case 2. Thus from this figure we showed that our new algorithm DC-FIFO results in reduced page faults compared to other replacement algorithms.

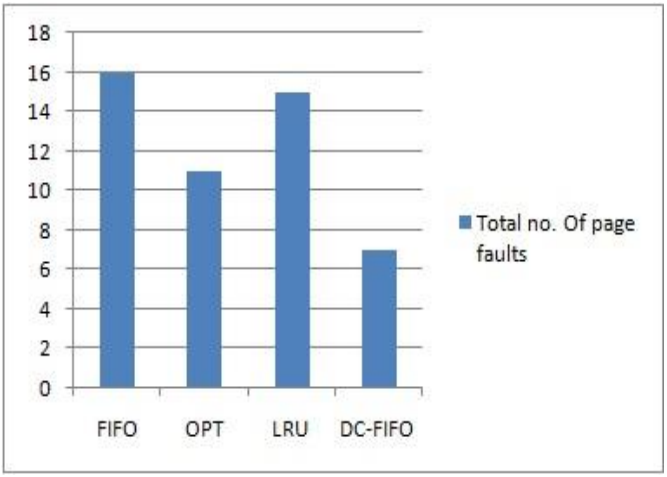

Fig 10: Comparison of page faults for case2

7. CONCLUSION

The performance page replacement algorithms like FIFO, OPT, LRU, Second Chance, LFU and MFU is measured in terms of number of page faults rate. We select one of the algorithm that result in low page fault rate. Among all these algorithms, optimal page replacement algorithm is efficient since it results in lowest page fault rate of all the algorithms. In an attempt towards this, we have proposed a new algorithm called Comparison Counting FIFO (CC-FIFO) and Distribution Counting FIFO (DC-FIFO) using input enhancement technique and implemented the same for two different reference strings. Our results and calculations show that proposed algorithm reduces the total number of page fault rate compared to the general approach of page replacement algorithms.

REFERENCES

[1] Wang Hong, "Study of Page Replacement Algorithm based on Experiment", International conference on mechanical Engineering and Automation, Advances in Biomedical Engineering, volume 10, 2012.

[2] Guangxia Xu, Lingling Ren and Yanbing Liu, "Flash Aware Page Replacement Algorithm", Mathematical Problems in Engineering, Hindawi Publishing Corporation, volume 2014.

[3] Anupam Bhattacharjee, Biolab Kumar Debnath, "A New Web Cache Replacement Algorithm", IEEE Conference on Communications, Computers and Signal Processing, August, 2005.

[4] Silberchatz, Galvin and Gagne, "Operating Systems Concepts", 8th edition, John Wiley and Sons, 2012.

[5] Dietel, Dietel and Choffnes, "Operating Systems", 3rd edition, Pearson education, 2009.

[6] Yogesh Niranjan and Shailendra Tiwari, "Design and Implementation of Page Replacement Algorithm for Web Proxy Caching", IJCTA, Volume 4, April 2013.

[7] Ali Khosrozadeh, Sanaz Pashmforoush, "Presenting a novel Page Replacement Algorithm based on LRU", Journal of Basic and Applied Scientific Research, 2012.

[8] Pooja Khulbe, Shruti Pant, "Hybrid (LRU) page Replacement Algorithm", IJCA, Volume 91, April 2014.

[9] Anany Levitin, "Introduction to the Design and Analysis of Algorithms", second edition, Pearson, 2008.