ANALYSIS OF THE PAGE REPLACEMENT ALGORITHMS GRAPHICALLY USING PERL

PROBLEM

REPORT ON PACKAGE

SUBMITTED BY

KIRUSHIKESH D B **Roll No : 18PT18**

JEYAKUMARAN M **Roll No : 18PT15**

SUBJECT : OPERATING SYSTEMS

DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES

PSG COLLEGE OF TECHNOLOGY

COIMBATORE - 641004

CONTENTS

ABSTRACT

A virtual memory system requires efficient page replacement algorithms to make a decision which pages to evict from memory in case of a page fault. Many algorithms have been proposed for page replacement. Each algorithm is used to decide on which free page frame a page is placed and tries to minimize the page fault rate while incurring minimum overhead. As newer memory access patterns were explored, research mainly focused on formulating newer approaches to page replacement which could adapt to changing workloads.
This project attempts to summarize major page replacement algorithms. We look at the traditional algorithms such as Optimal replacement, LRU, FIFO,Second Chance,LFU,MFU,Additional Bit,Random Page Replacement Algorithms.We will also compare their relative performance of Algorithms using probabilistically generated reference strings.

1.1 INTRODUCTION

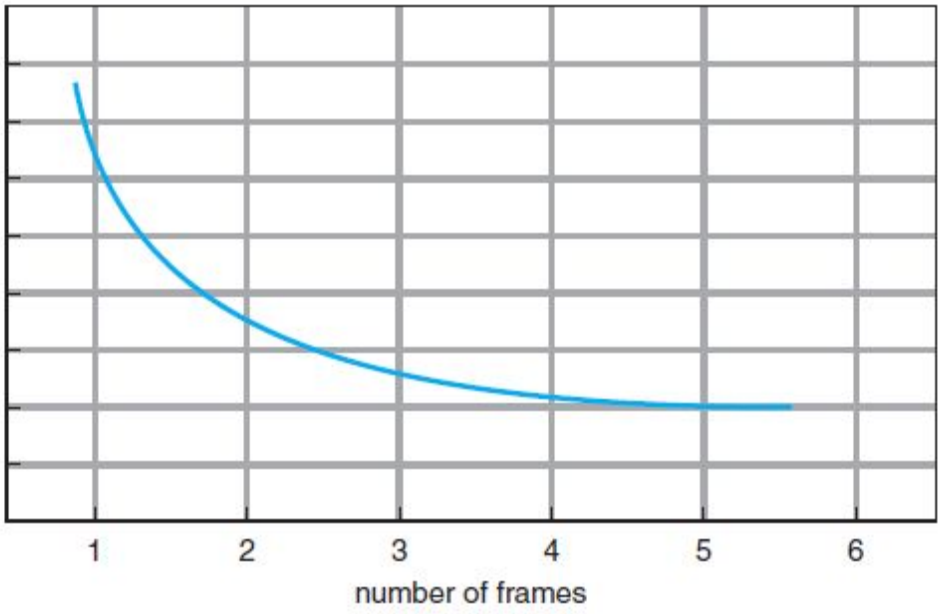
The full potential of multiprogramming systems can be realized by interleaving the execution of more programs. Hence we use a twolevel memory hierarchy consisting of a faster but costlier main memory and a slower but cheaper second memory. In virtual memory the combined size of program code, data and stack may exceed the amount of main memory available in the system. This is made possible by using secondary memory, in addition to main memory. Pages are brought into main memory only when the executing process demands them, this is known as demand paging.
A page fault typically occurs when a process references a page that is not marked present in main memory and needs to be brought from secondary memory. In such a case an existing page needs to be discarded. The selection of such a page is performed by page replacement algorithms which try to minimize the page fault rate at the least overhead. This paper outlines the major advanced page replacement algorithms and compares the performance between different algorithms.

1.2 DESCRIPTION

There are many different page-replacement algorithms. Every operating system probably has its own replacement scheme. How do we select a particular replacement algorithm? In general, we want the one with the lowest page-fault rate.

We evaluate an algorithm by running it on a particular string of memory references and computing the number of page faults. The string of memory references is called a reference string.
For example,consider the reference string of a process follows the sequence:

1,4,1,6,1,6,1,6,1,6,1

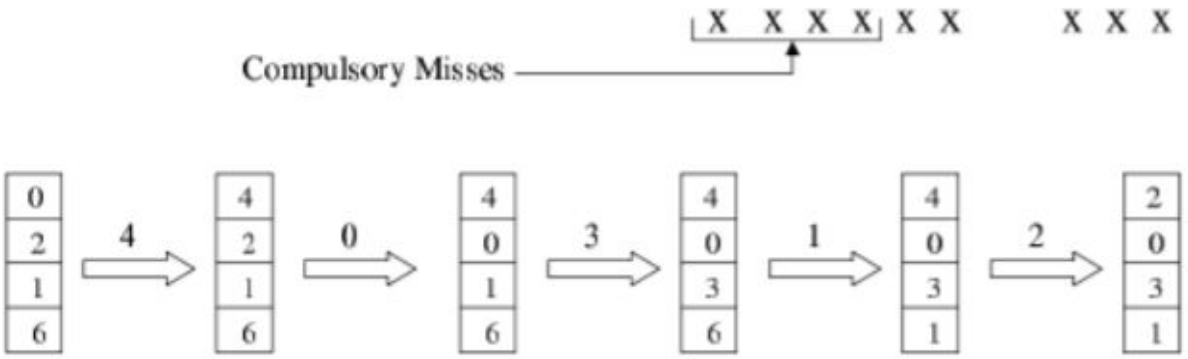


To determine the number of page faults for a particular reference string and page-replacement algorithm, we also need to know the number of page frames available. As the number of frames available increases, the number of page faults decreases.

1.2.1 First In First Out (FIFO)

The First in First out (FIFO) page replacement algorithm is the simplest approach of replacing the page.The idea is to replace the oldest page in main memory from all the pages i.e. that page is replaced which has been in main memory for the greatest period of time. A FIFO queue can be created to hold all the pages in main memory. The page that is at the front is replaced and when the page is fetched into memory it is inserted at the Rear end. For this reason, FIFO algorithm is seldom used.

•Consider the following reference string: 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1



•Fault Rate = 9 / 12 = 0.75

The advantage of FIFO page replacement algorithm is easy to implement and disadvantage is that it suffers from Belady's anomaly. Belady's anomaly is an unexpected result in FIFO page replacement algorithm. In some of the reference strings, increasing the size of the memory increases the page fault rate.

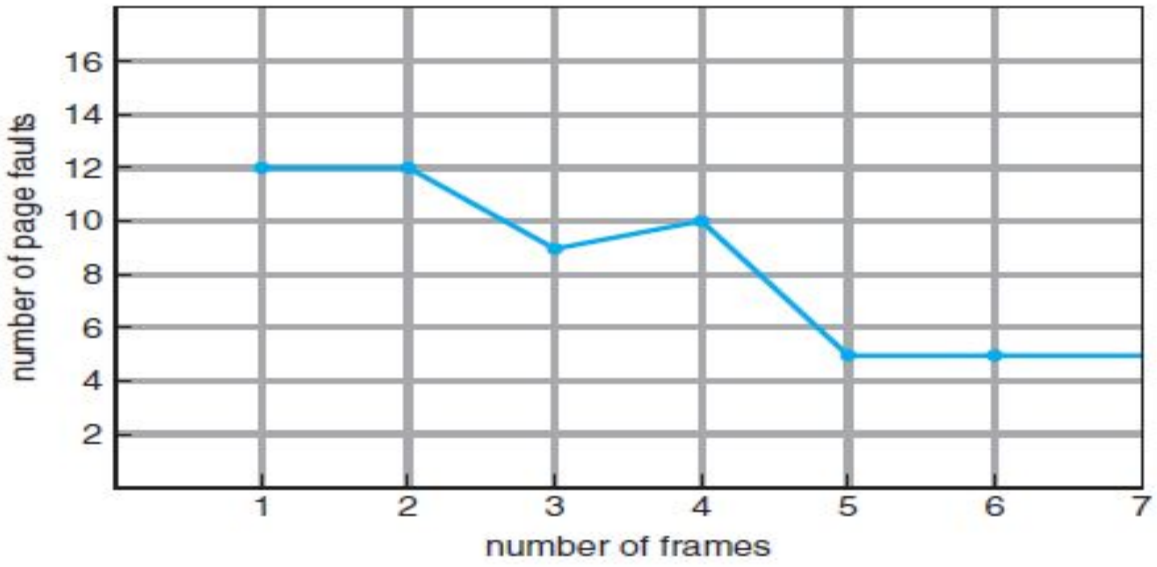
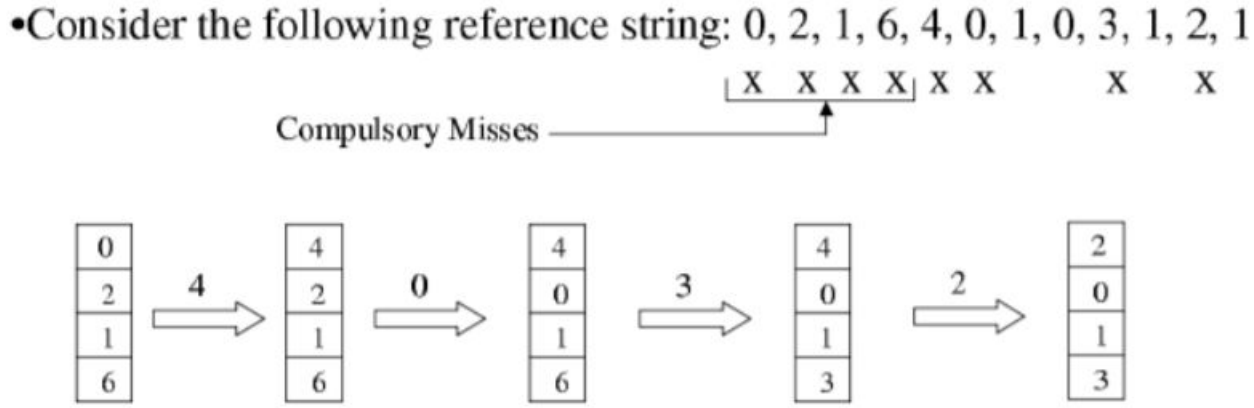


Figure 9.13 Page-fault curve for FIFO replacement on a reference string.

1.2.2 Least Recently Used (LRU)

Page Replacement Algorithm replaces the page in memory that has not been used for the longest period of time.



•Fault Rate = 8 / 12 = 0.67

The problem with this algorithm is the complexity in implementation.The implementation may involve hardware or software support. The implementation of this policy can be possible using matrix, counters, linked list etc.
The advantage of LRU page replacement algorithm is that it does not suffer from Belady's anomaly and the disadvantage is that it needs expensive hardware support or additional data structure to implement.

1.2.3 Second Chance Page Replacement Algorithm

In the Second Chance page replacement policy, the candidate pages for removal are considered in a round robin matter, and a page that has been accessed between consecutive considerations will not be replaced. The page replaced is the one that, when considered in a round robin matter, has not been accessed since its last consideration.

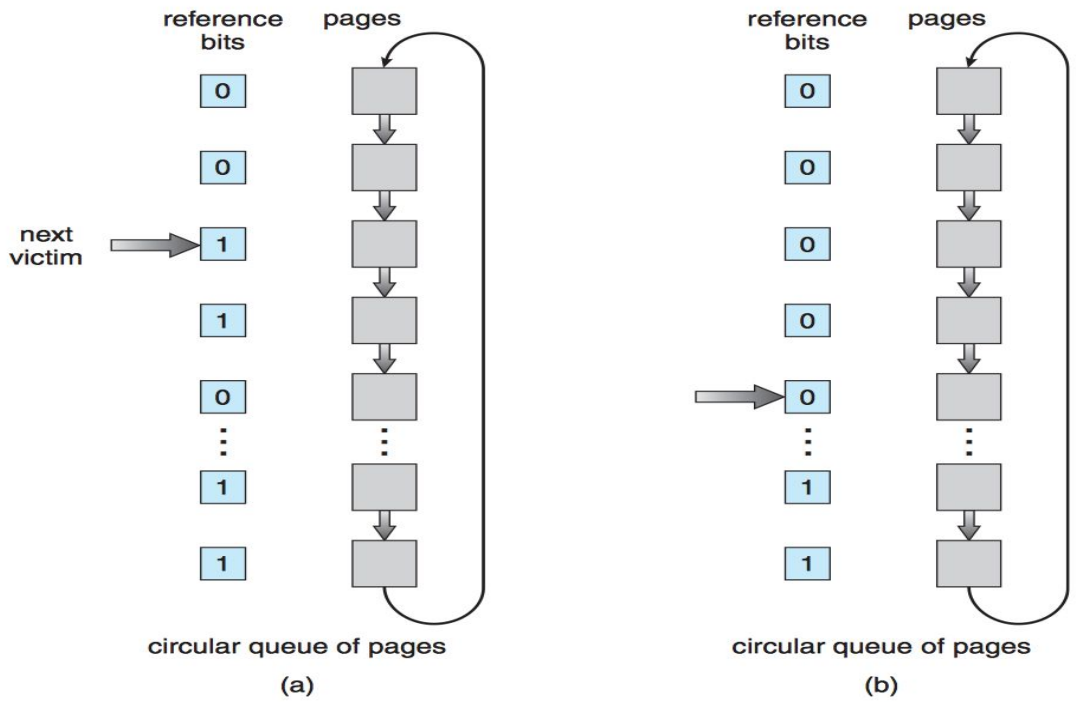
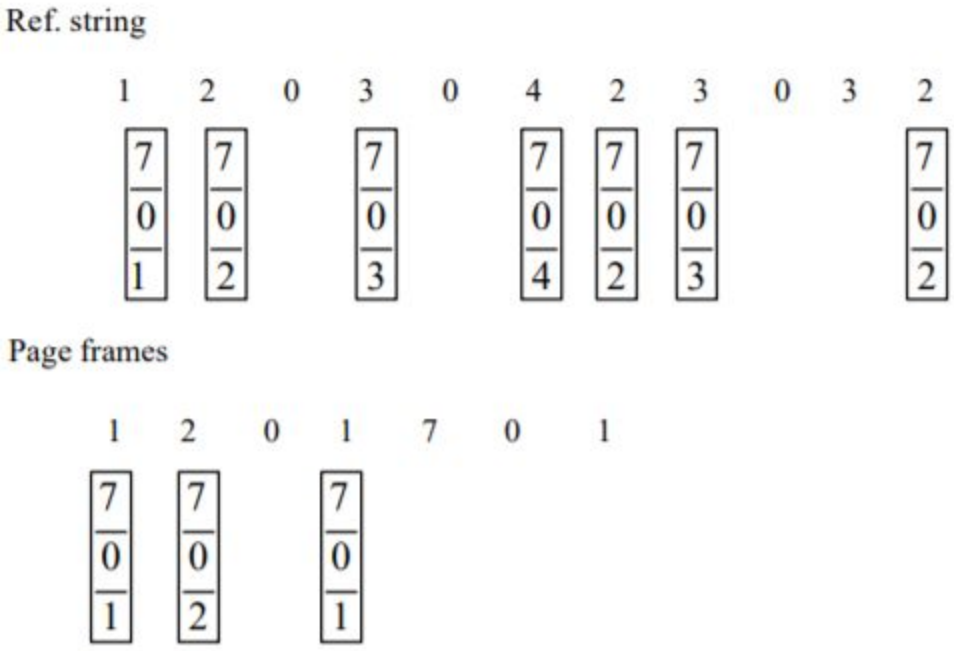


Figure 9.17 Second-chance (clock) page-replacement algorithm.

It can be implemented by adding a "second chance" bit to each memory frame-every time the frame is considered, this bit is set to 1, which gives the page a second chance, as when we consider the candidate page for replacement, we replace the first one with this bit set to 0 (while zeroing out bits of the other pages we see in the process).

1.2.4 Least Frequently Used (LFU)

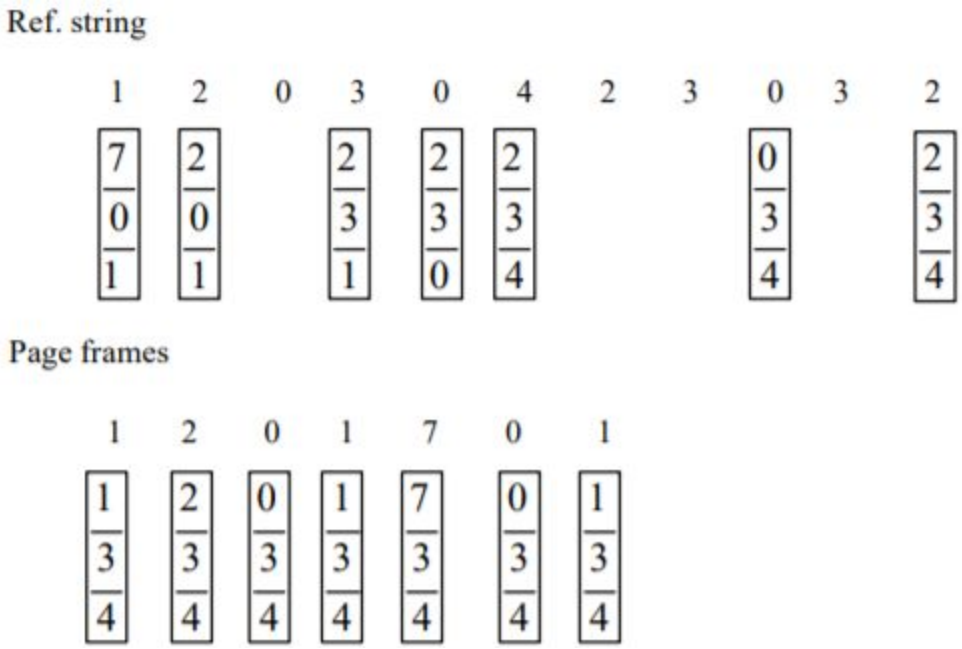
This is a type of cache algorithm used to manage memory within a computer. The standard characteristics of this method involve the system keeping track of the number of times a block is referenced in memory. When cache is full and requires more room the system will purge the item with the lowest reference frequency.



While the LFU method may seem like an intuitive approach to memory management it is not without faults. Consider the new items that just entered the cache are subject to being removed very soon again, because they start with a low counter, even though they might be used very frequently after that. Due to major issues like these, an explicit LFU system is fairly uncommon; instead, there are hybrids that utilize LFU concepts.

1.2.5 Most Frequently Used(MFU)

The most frequently used page-replacement algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used. So the page with the highest frequency is replaced whenever needed.



Neither MFU nor LFU replacement is common. The implementation of these algorithms is expensive, and they do not approximate OPT replacement well.

1.2.6 Additional Reference Bit Page Replacement Algorithm

We can gain additional ordering information by recording the reference bits at regular intervals. We can keep an 8-bit for each page in a table in memory. The Operating System shifts the reference bit for each page into the high-order bit of its 8-bit,shifting the other buts right by 1 bit and discarding the low-order bit.

These 8 bits contain the history of page use for the last eight time periods. If we interpret these 8-bit as unsigned integers, the page with the lowest number is the LRU page, and it can be replaced. For example, A page with a history register value of 11000100 has been used more recently than one with a value of 01110111.

1.2.7 Optimized Algorithm

It gives the optimized solution by replacing the page which is not going to be used in the nearest future. ie) It replaces the page which is not present in the upcomming pages or going to be used only at last.

1.2.8 Random Algorithm

When a Page Fault occurs it replaces a random page.

1.3 SYSTEM CALLS USED

1.3.1 RAND()

Function to generate random numbers.

1.4 TOOLS AND TECHNOLOGY

In this project, we made use of the Linux operating systems, used C Program to implement the algorithms and used the tool Chart::Gnuplot which is a library in Perl which uses Gnuplot to plot the results which we have analysed. We have used Perl scripts for integrating the algorithms and plotting the results through Gnuplot. We use Imagemagic to render the images of the Gnuplot.

1.5 WORKFLOW

The workflow of this project is divided into seven steps as shown in the figure given below :

1. We implemented all the algorithms in C.
2. Integrated all the algorithms.
3. Plotted their outputs
4. Calculate their average
5. Analysed through the plots we obtained.

1.6 RESULTS

We have noted down the page faults produced by the algorithms for an input 1,2,3,4,1,2,5,1,2,3,4,5. The results are shown in Figure 1 and Figure 2 respectively.

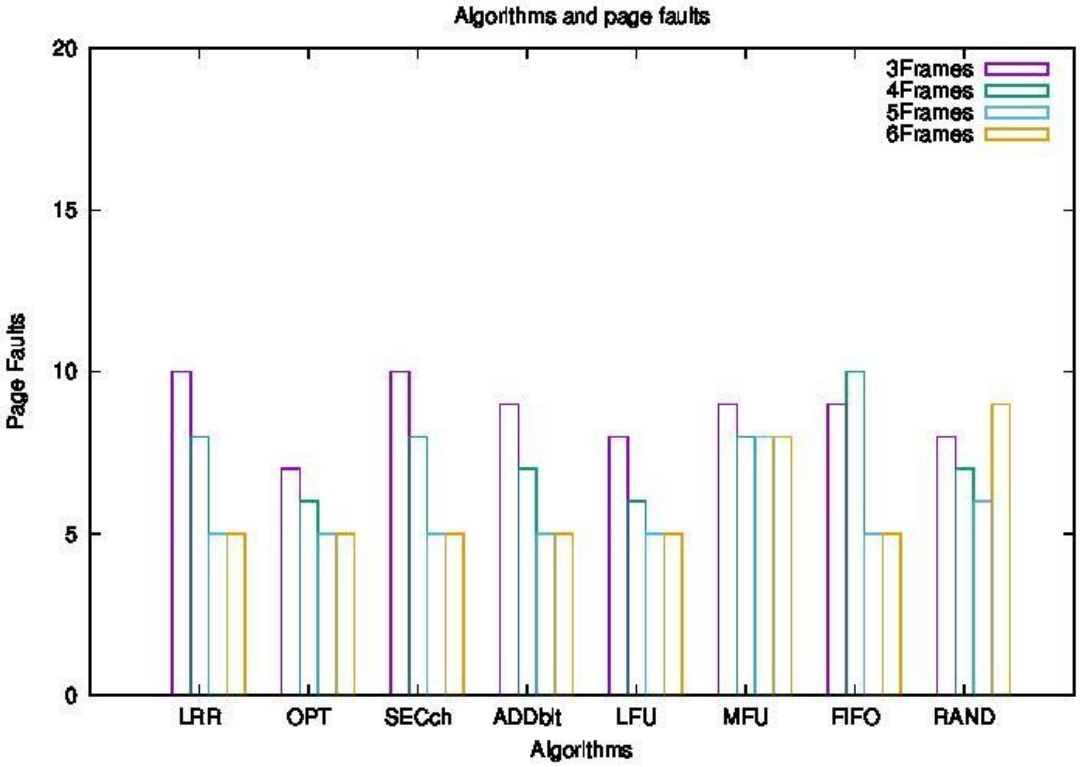


Figure 1

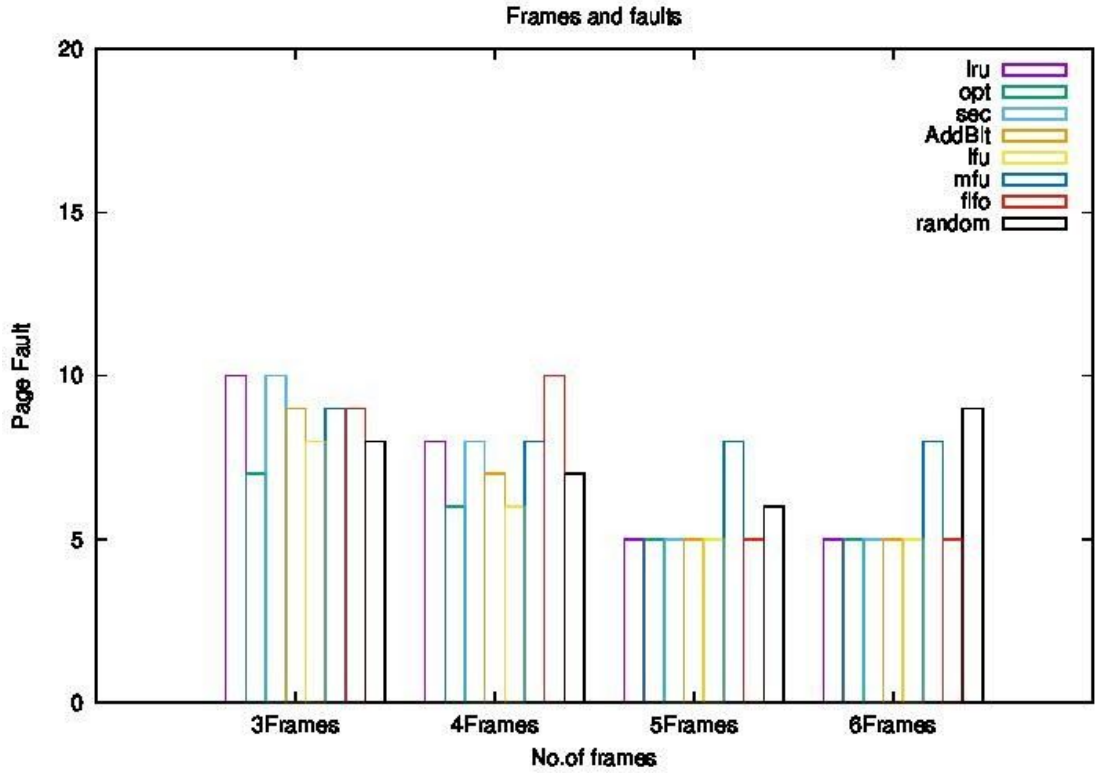
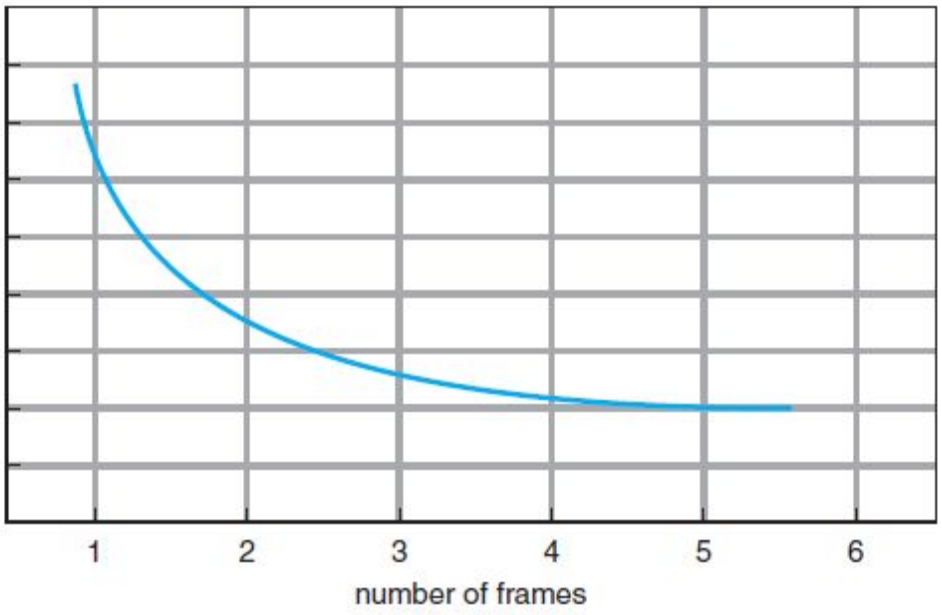


Fig 2

Normally in all the algorithms when the number of pages allocated increases then the page faults will be continuously decreasing like shown in Figure 3, but for some reference strings the algorithms which are implemented using queues like FIFO will show an increase in the number of page faults like shown in Figure 4, this phenomenon of the algorithm is known as Belady's Anomaly and finally beyond a certain point all the algorithms tend to converge to the optimal solution which is the solution of the optimized algorithm. All these characteristics are observed in the Figure 1 and Figure 2.



3 Figure 3

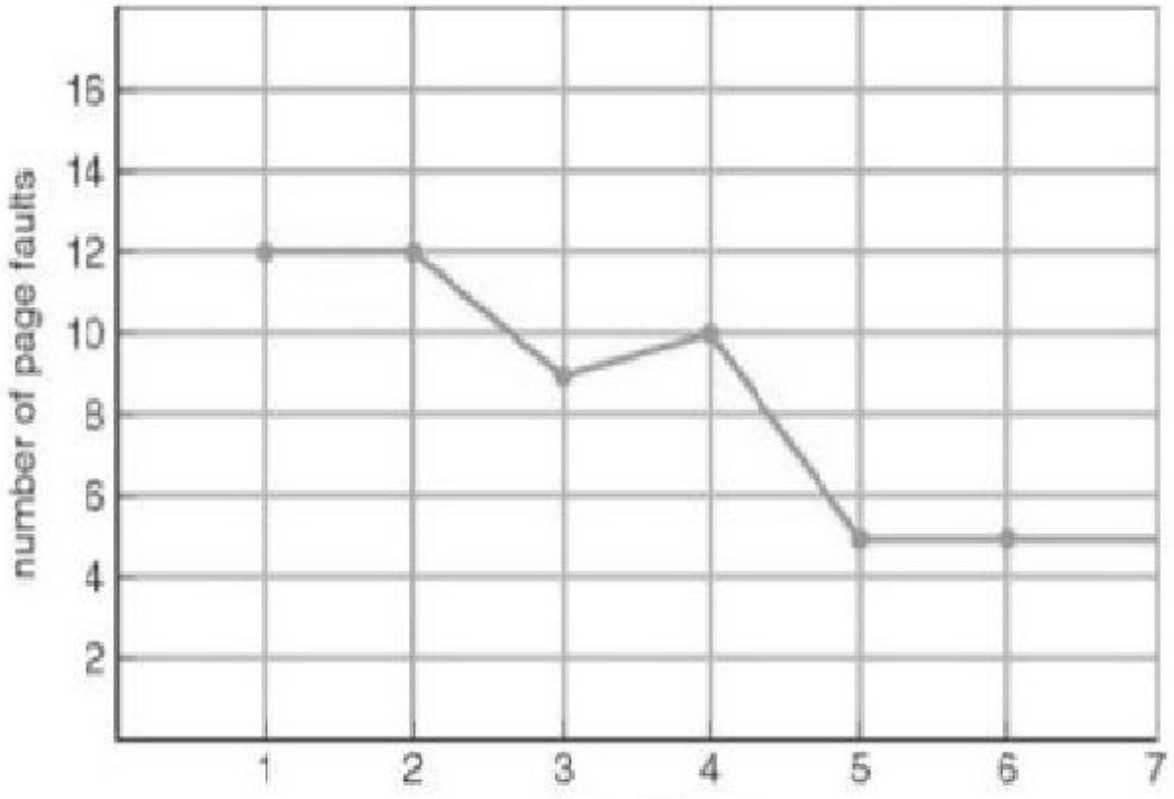


Figure4

This characteristic is not shown for all the reference strings. Let's take another example. We have noted down the page faults produced by the algorithms for an input 1,2,3,4,5,6,1,2,3,4,5,6 The results are shown in Figure 5 and Figure 6 respectively.

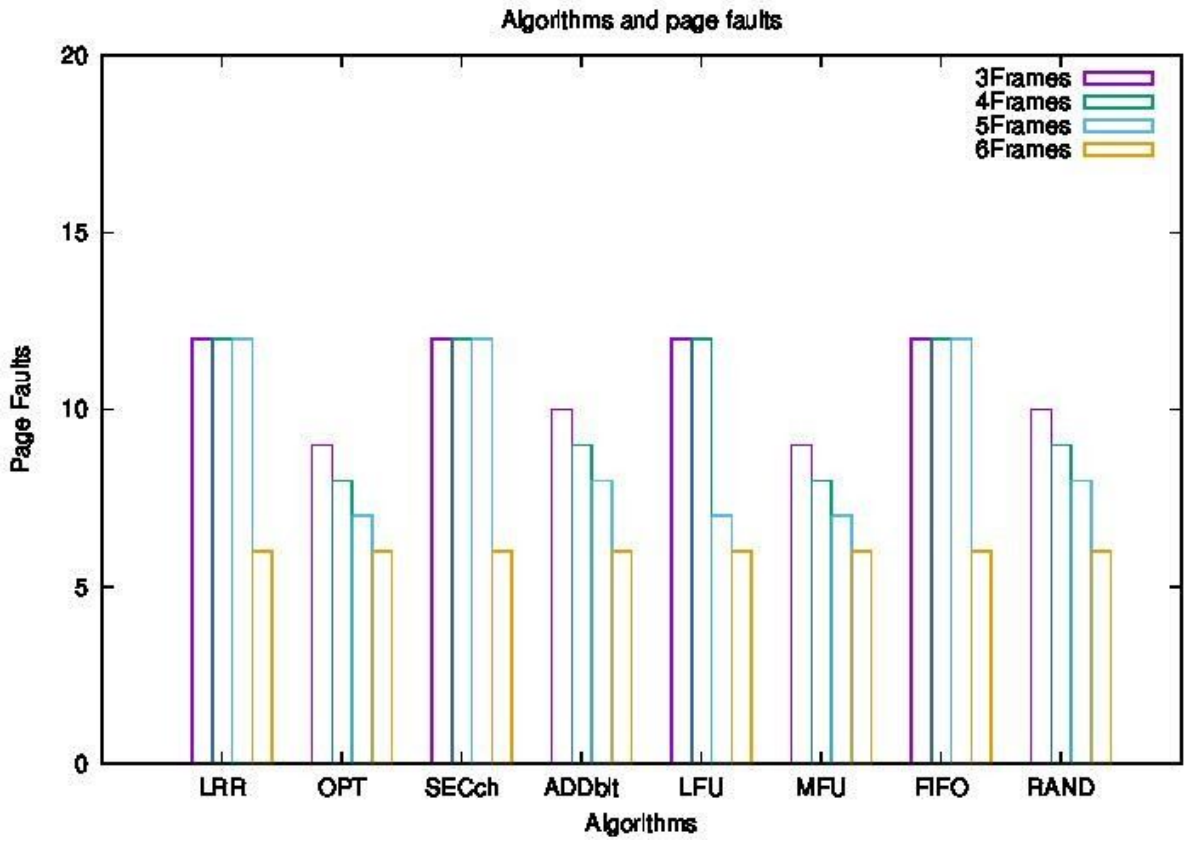

Algorithms and page faults
3Frames
4Frames
5Frames
6Frames
Page Faults
20
15
10
5
0
LRR
OPT
SECch
ADDbit
LFU
MFU
FIFO
RAND
Algorithms

Figure 5

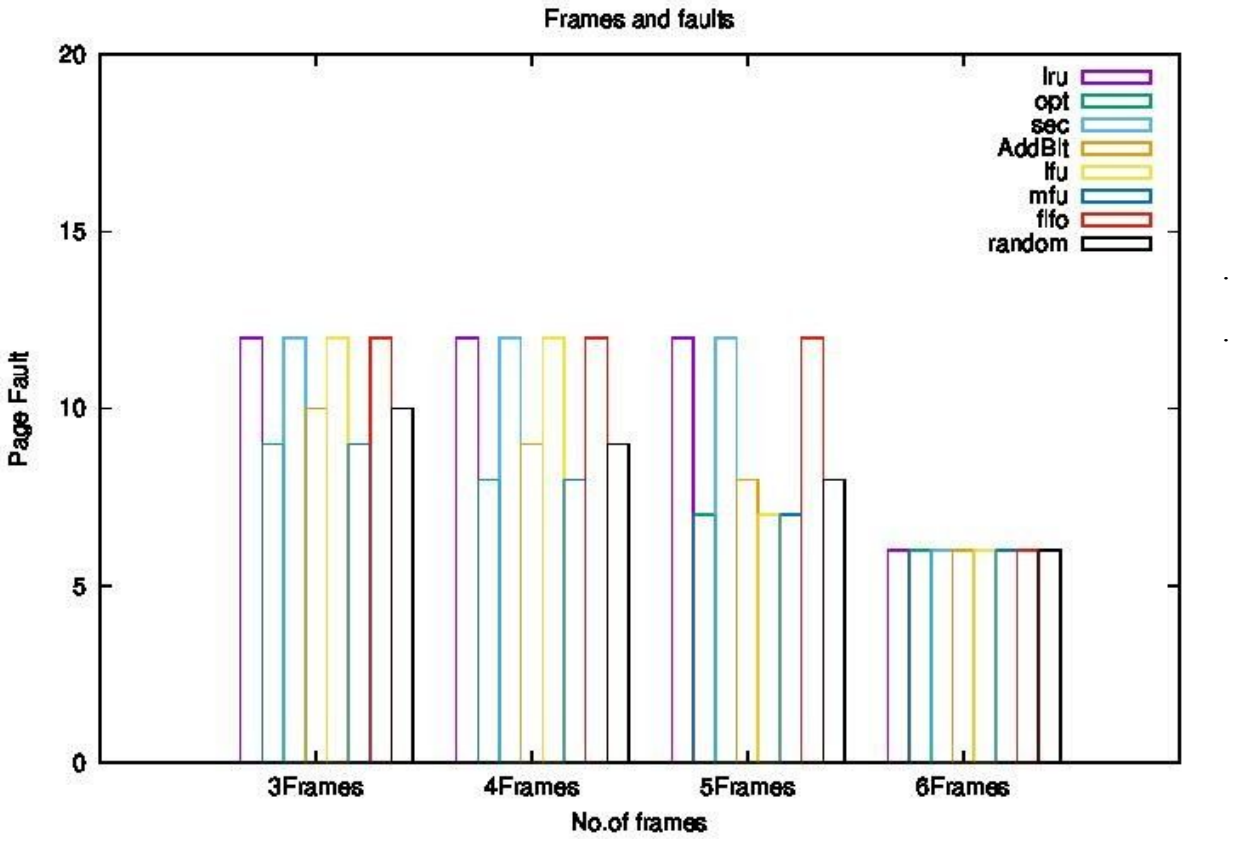


Figure6

To choose the best algorithm we need to take the average of the hit ratios of the above results.

An algorithm is chosen for our Operating System based on its

1. Page Fault
2. Time Complexity
3. Memory used

Since the process of replacing a page is a difficult job and also consumes more CPU time. One can invest on the time complexity and the memory requirement of the algorithm which is used to decide the page which is to be replaced.

The average of the results are given in the Figure 7 and Figure 8

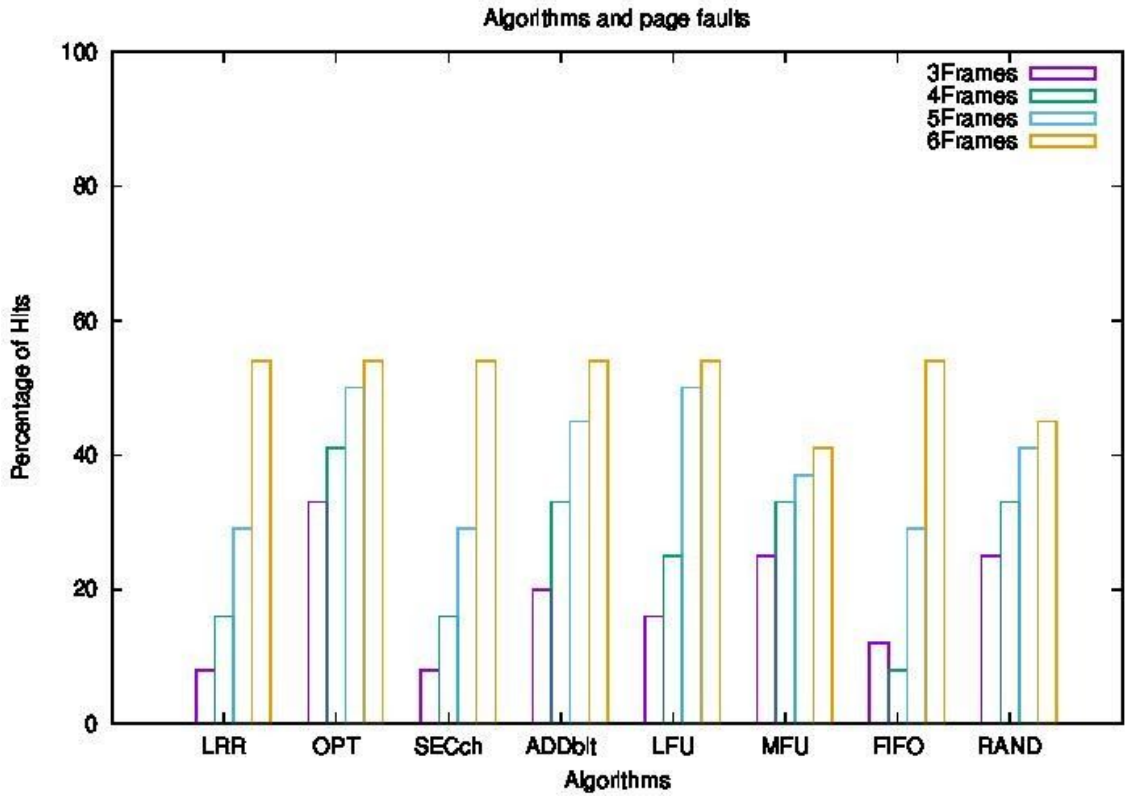

Algorithms and page faults
Percentage of Hits
Algorithms
3Frames
4Frames
5Frames
6Frames
LRR
OPT
SECch
ADDbit
LFU
MFU
FIFO
RAND
0
20
40
60
80
100

Fig 7

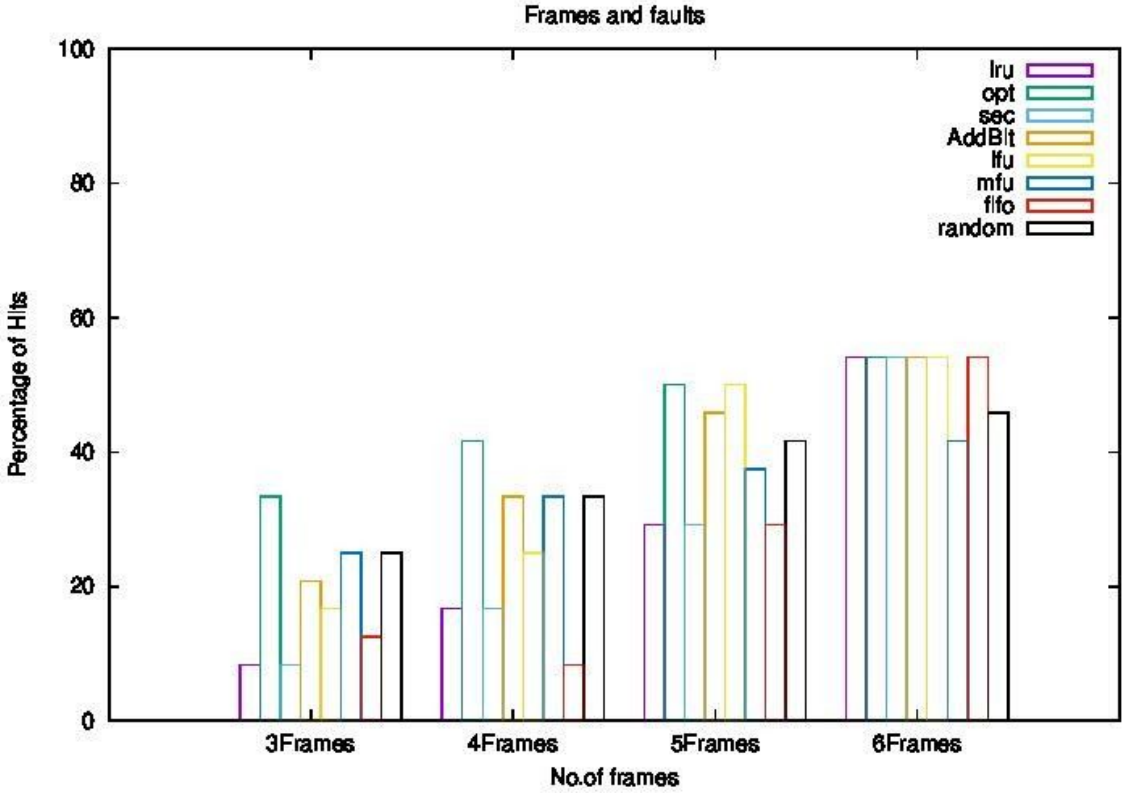

Frames and faults
Percentage of Hits
No.of frames
lru
opt
sec
AddBit
lfu
mfu
fifo
random
3Frames
4Frames
5Frames
6Frames
0
20
40
60
80
100

Fig 8

1.7 CONCLUSION

We Since all the algorithms will not behave identically for a reference string. It is purely the decision of the designer of the Operating System on what algorithm is going to be used by the Operating System. Using the optimized algorithm is highly impossible since computing the pages repeating next is an impossible task in an operating system. Depending on the need of the operating system and its efficiency in computing the algorithm is chosen. From our analysis after the optimized algorithm the additional reference bit algorithm and the LFU has a good hit ratio. But computing the references of the pages which are replaced is als a difficult job. Normally linux kernels would use LRU or second chance algorithms since they are easy to compute.

1.8 BIBLIOGRAPHY

1.8.1 WEBSITES

1. https://www.tutorialspoint.com/perl/index.htm
2. https://www.perltutorial.org/

1.8.2 BOOKS

1. Silberschatz Abraham, Galvin P B, Gagne G (2011), *Operating System Concepts*, John Wiley & Sons. INC