OS LAB ASSIGNMENT 8:

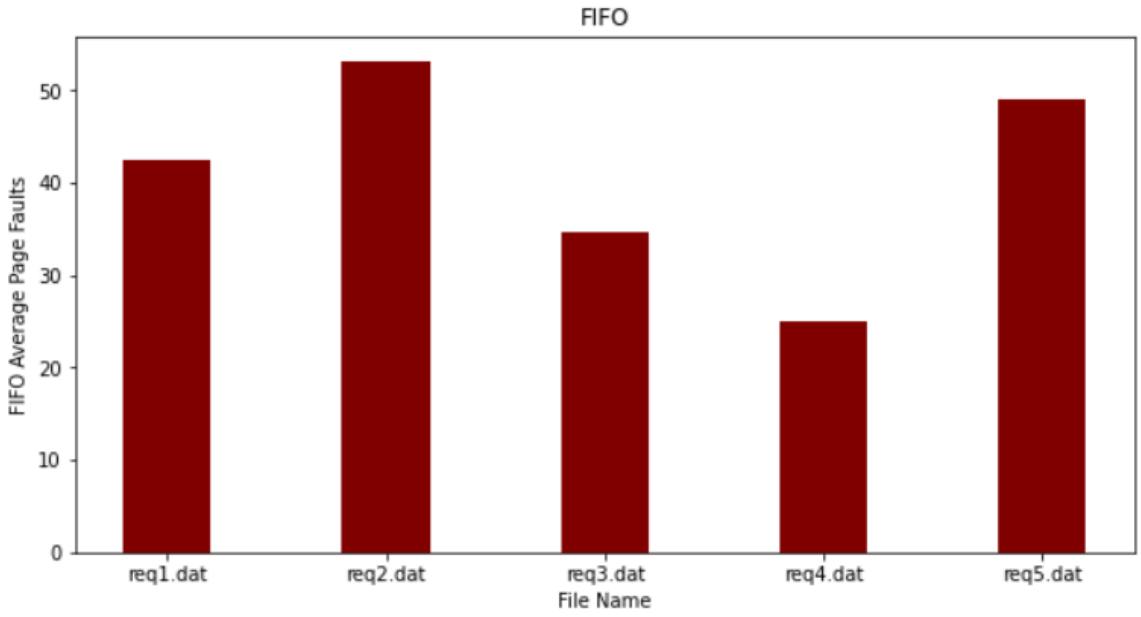
- Chidaksh (200010046)

Observations and Analysis:

1) FIFO:

No matter how often it is called, the first page that has been taken will always be the first to be replaced. If something has remained for a while, it must leave. It will be switched out first. The simplest algorithm for replacing pages is this one. The operating system maintains a queue for all of the memory pages in this method, with the oldest page at the front of the queue. The page in front of the queue is chosen for removals when a page needs to be changed.

Below is a bar graph showing the results of FIFO implemented on various files.

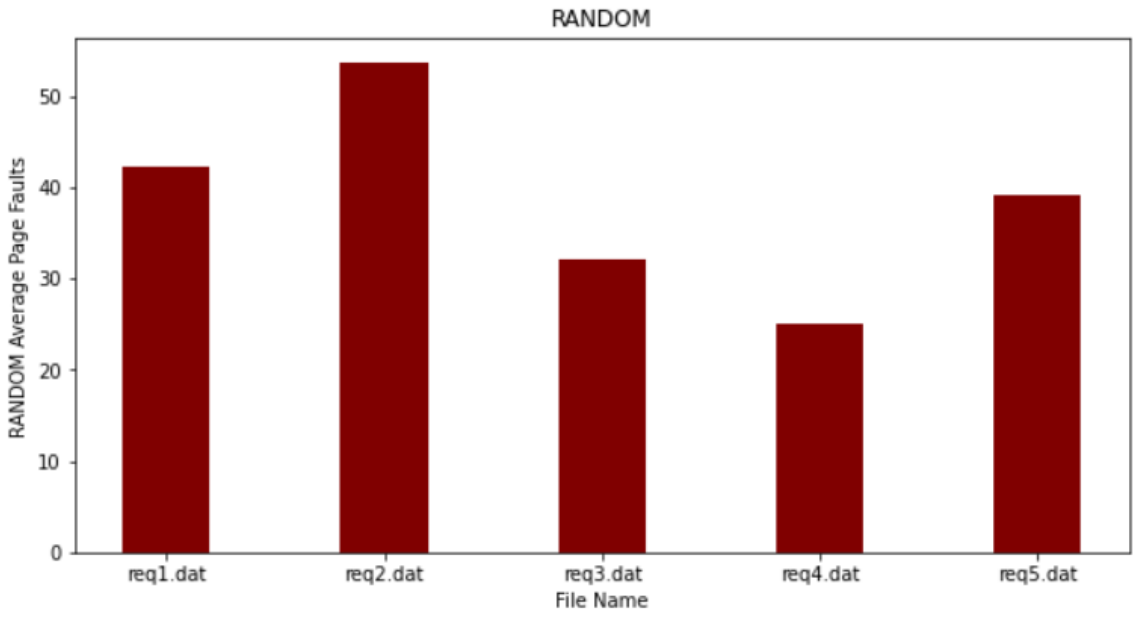


2) RANDOM:

RANDOM is a page replacement algorithm that evicts a random page from memory when there is not enough space. This algorithm works on the principle that any page in memory can be evicted since the likelihood of the next page access is unpredictable.

This is a straightforward solution that isn't very effective because it simply substitutes existing pages at random without taking anything into consideration. I use the following methodology to put the algorithm into practice.

Below is a bar graph showing the results of RANDOM implemented on various files.

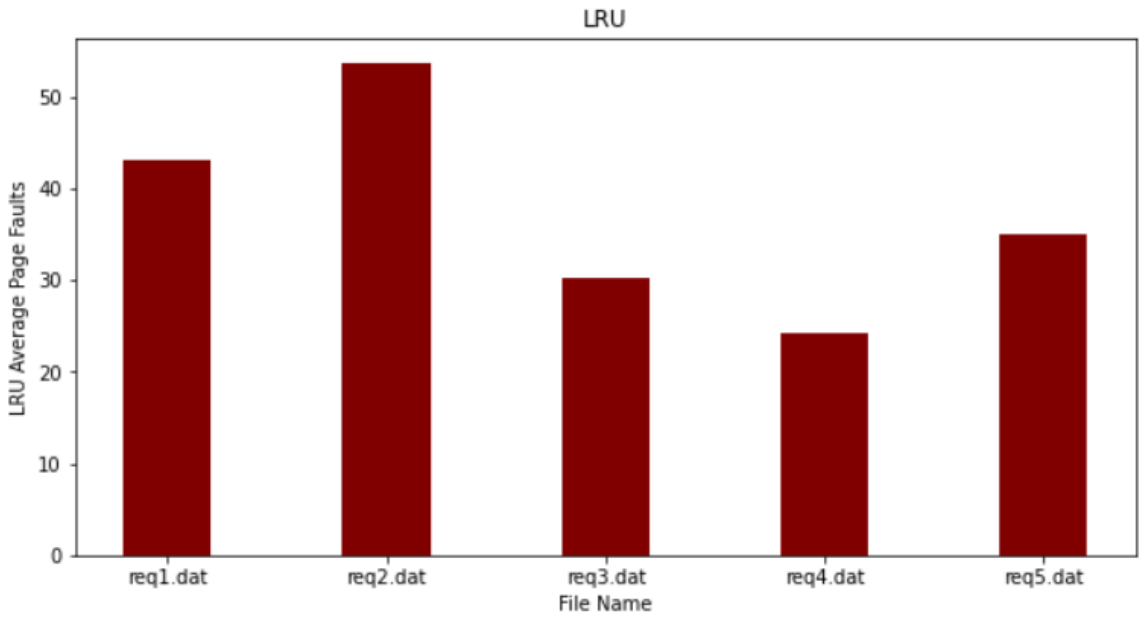


3) LRU:

LRU algorithm works on the principle that the pages that have been used recently are more likely to be used again in the near future. Therefore, when a page needs to be evicted, the page that was least recently used is chosen.

Here, the algorithm is greedy. The most recently used page is the one that has to be replaced. Based on the idea's locality of reference, it is unlikely that the least recently used page will be chosen.

Below is a bar graph showing the results of LRU implemented on various files.



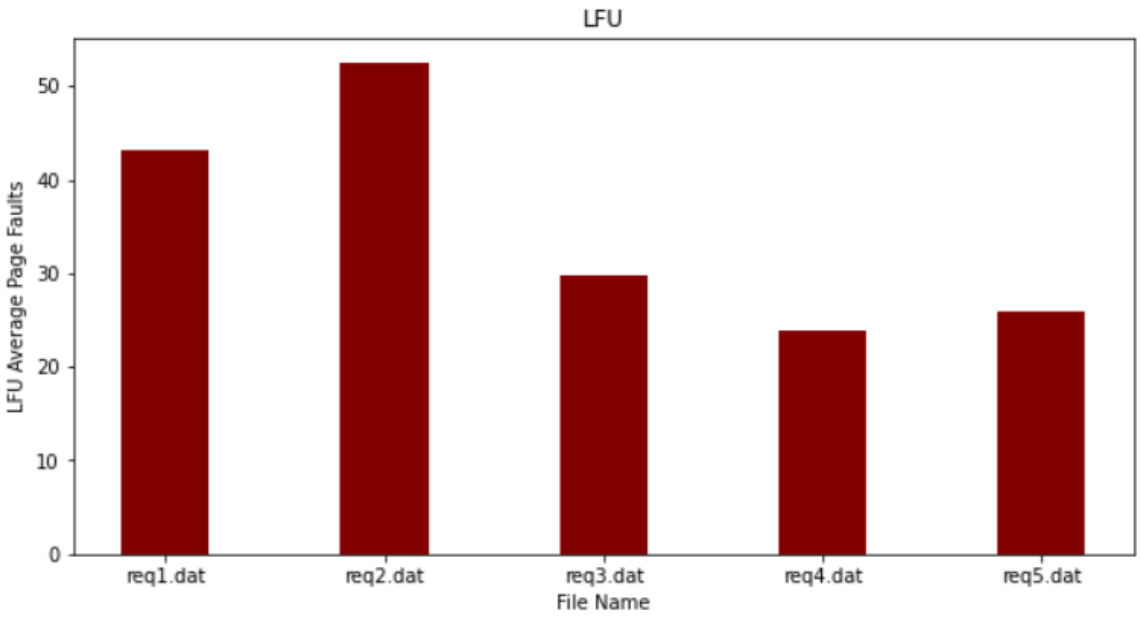
4) LFU:

LFU is a page replacement algorithm that evicts the least frequently used page from memory when there is not enough space. This algorithm works on the principle that the pages that

have been used the least number of times are less likely to be used again in the near future.

LFU operates by monitoring a counter that counts the number of times each memory page has been read. The page with the smallest counter value is chosen when a page needs to be changed. The page that was brought into memory first is picked for replacement if numerous pages have the same lowest counter value.

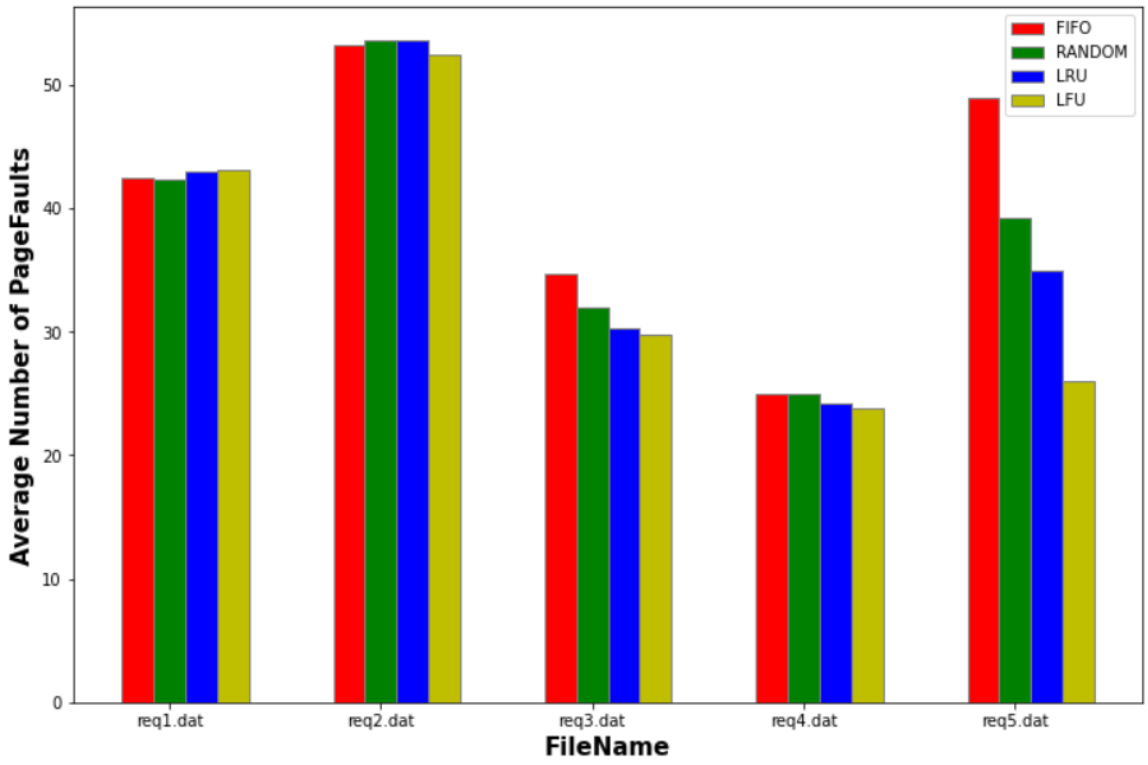
Below is a bar graph showing the results of LFU implemented on various files.

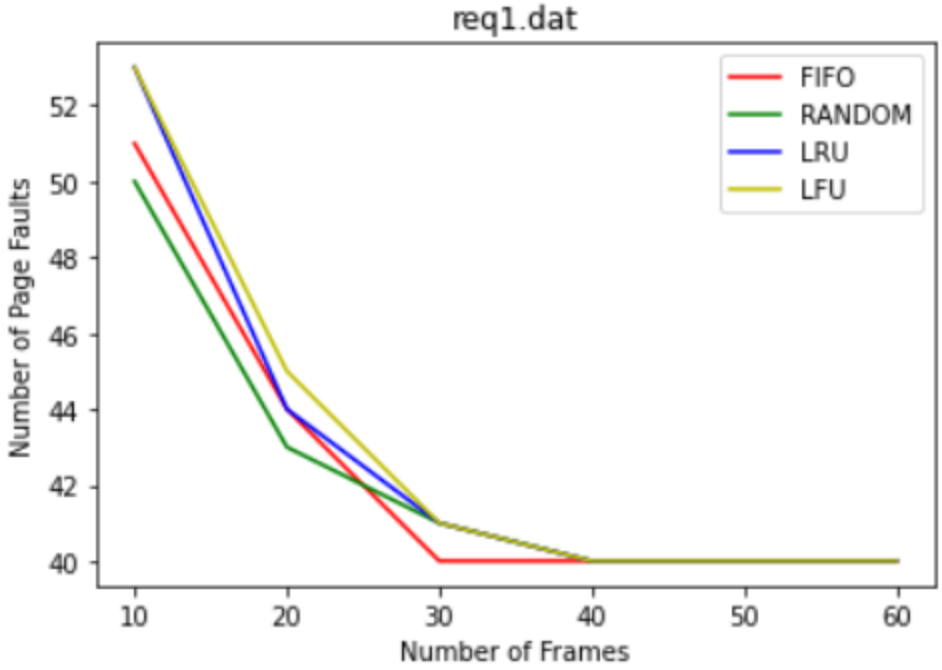


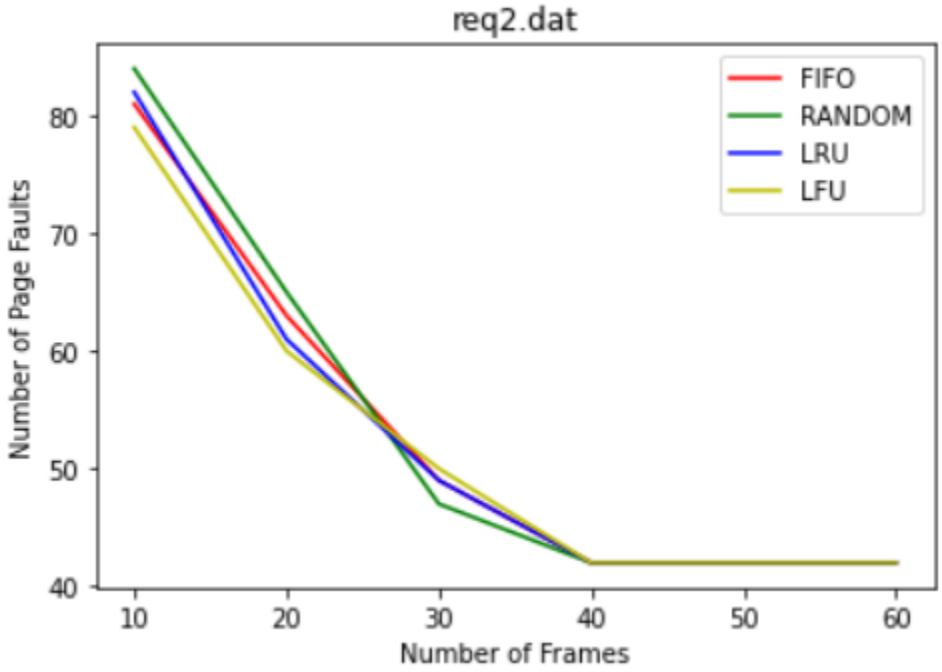
Results and Comparisons:

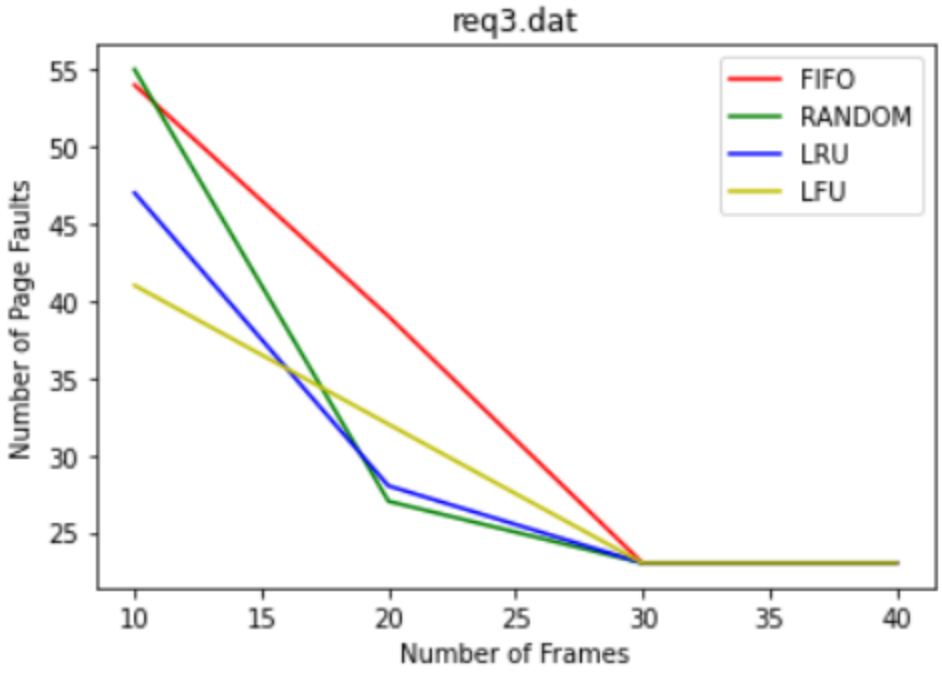
The following table shows the average of the number of page faults generated when varying the frame queue size (across all the sizes considered)

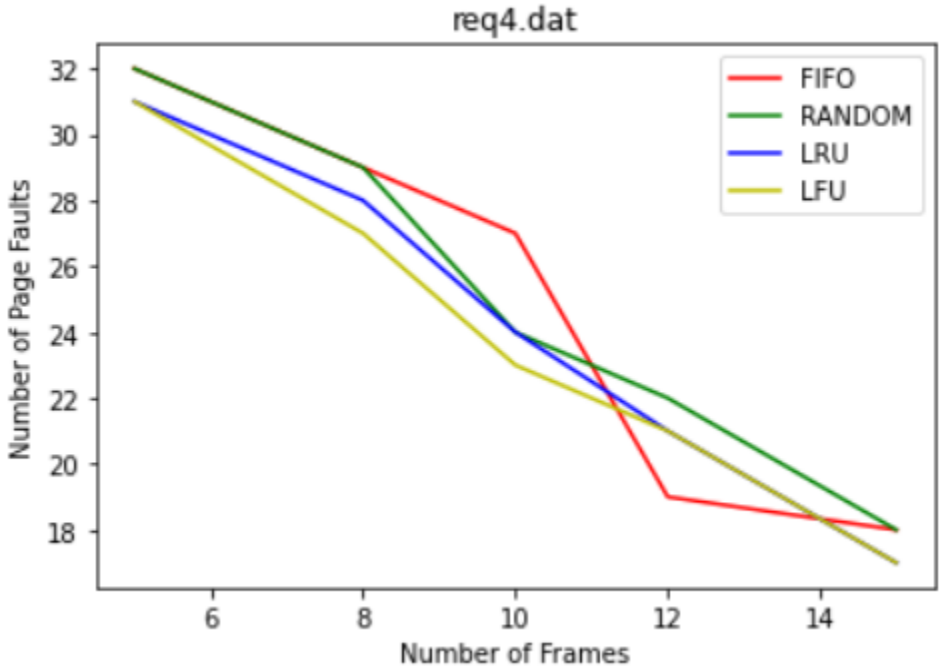
Algorithm	req1.dat	req2.dat	req3.dat	req4.dat	req5.dat
FIFO	42.5	53.17	34.75	25.0	49.0
RANDOM	42.33	53.67	32.0	25.0	39.2
LRU	43.0	53.67	30.25	24.2	35.0
LFU	43.17	52.5	29.75	23.8	26.0

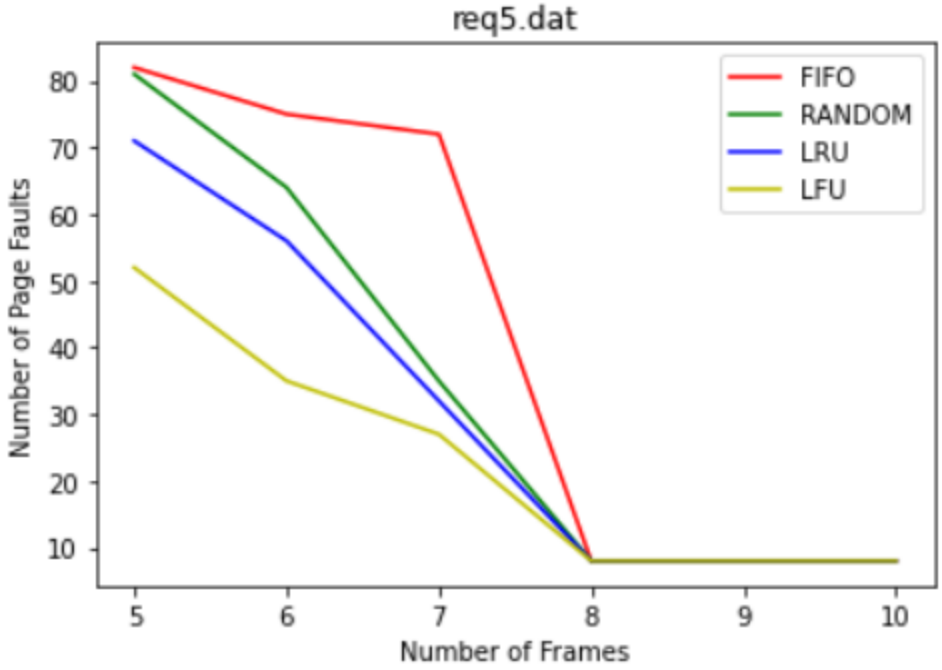



req1.dat
FIFO
RANDOM
LRU
LFU
Number of Page Faults
52
50
48
46
44
42
40
10
20
30
40
50
60
Number of Frames


req2.dat
FIFO
RANDOM
LRU
LFU
Number of Page Faults
80
70
60
50
40
10
20
30
40
50
60
Number of Frames


req3.dat
FIFO
RANDOM
LRU
LFU
Number of Page Faults
Number of Frames
55
50
45
40
35
30
25
10
15
20
25
30
35
40


req4.dat
FIFO
RANDOM
LRU
LFU
Number of Page Faults
Number of Frames
32
30
28
26
24
22
20
18
6
8
10
12
14



Graphical Analysis:

We see that there are numerous page flaws at the beginning of the varied size from 20 to 60. This is due to the fact that only a maximum of 20 pages will be reached, which is a fairly small quantity. Nevertheless, after we gradually raise it to 60, we can observe how there are now significantly fewer page errors. This is so that, since even the system will only support a maximum of 60, the greater number will result in more pages existing there themselves and fewer page faults.
Conclusion: Less page faults, more pages accommodate frames, and more frames.

Which Algorithm is Best?

Comparing these algorithms, LRU and LFU are both based on the usage history of pages and are more likely to keep pages in memory that are frequently used. LRU is better suited for applications that have temporal locality, where pages that have been accessed recently are more likely to be accessed again in the near future. LFU is better suited for applications

that have spatial locality, where pages that are accessed together are more likely to be accessed again together in the near future. FIFO is a simple algorithm that does not require any history or usage information, but it can be inefficient in cases where a page that was loaded a long time ago is still being used frequently. RANDOM is the simplest algorithm that is easy to implement, but it may not perform well in cases where pages with high usage or locality need to be kept in memory.