OPERATING SYSTEMS
CODING ASSIGNMENT 2

ANIKET GOYAL	17114011
BHAVYE JAIN	17114020
KAUSTUBH TRIVEDI	17114044
ROODRAM PANERI	17114066
SAURABH SINGH	17114068
TWARIT WAIKER	17114074
SHIVANSH BINDAL	17115088

Complexity Analysis

Time Complexity

- **LRU Counter**: O(n*f)

  Here ‘n’ is the length of the string and ‘f’ is the number of frames.

- **LRU Stack**: O(n*f)

  Here ‘n’ is the length of the string and ‘f’ is the number of frames.

- **LRU Aging Clock**: O(n*f*r)

  Here ‘n’ is the length of the string, ‘f’ is the number of frames and ‘r’ is the register size.

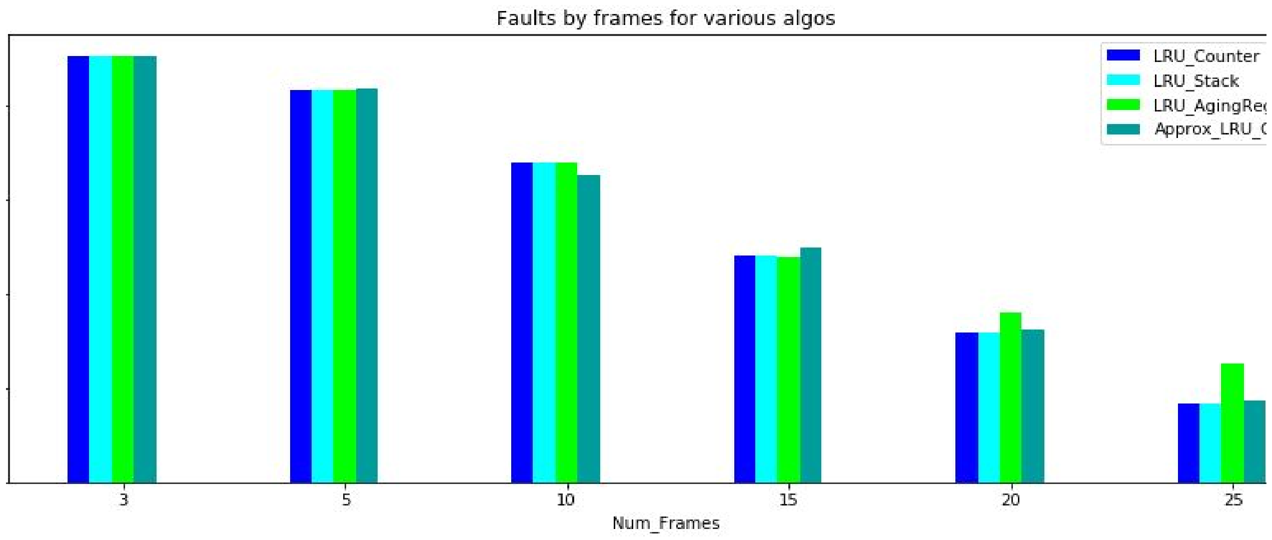
- **Approximate LRU Clock**: O(n*f)

  Here ‘n’ is the length of the string and ‘f’ is the number of frames.

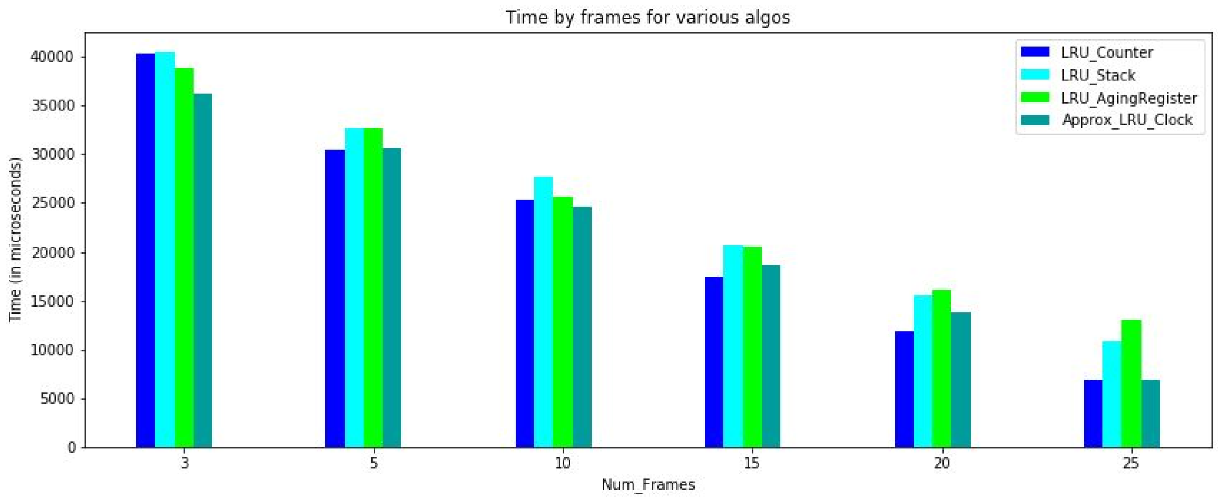
Now, as the number of frames increases, then the number of page fault decreases. We added a time delay of 10000 microseconds to account for every page fault. The explanation for the trends seen in the graphs is done in the critical analysis.

Graphs

The graph for the number of frames vs. the number of page faults



The graph for the average time taken to execute the algorithms for different corner cases



Critical Analysis of Graphs:

Proceeding to analyse the behaviour of various algorithms when the number of frames is changed:

LRU_Counter: LRU_Counter is shown in blue in the graph. As per the time complexity, the time taken to execute the code must increase as the number of frames increase. But, there will be more page faults when the number of frames is less so the number of times a page needs to replaced and moved to primary memory from the secondary memory will be more. So, more delay is observed in the code and more time is taken when the number of frames is less. Therefore the time decreases with an increase in the number of frames.

LRU_Stack: LRU_Stack is shown in light-blue colour in the graph. As per the time complexity, the time taken to execute the code must increase as the number of frames increase. But, there will be more page faults when the number of frames is less so the number of times a page needs to replaced and moved to primary memory from the secondary memory will be more. So, more delay is observed in the code and more time is taken when the number of frames is less. Therefore the time decreases with an increase in the number of frames. Time for LRU_Stack is more than LRU_Counter even though the time complexity for both is the same because, in Stack, additional time is needed to implement and update the Stack every time.

LRU_AgingRegister: LRU_AgingRegister is shown in green in the graph. As a register is used in Aging Register, it can help us predict with an almost optimal performance which page to replace. As a register is used and the performance is good, we can see in the trend that the Aging Register is comparatively stable and is less prone to abrupt time changes due to change in the number of page frames.

Approx_LRU_Clock: Approx_LRU_Clock is shown in dark green. A circular list of pages resident in memory is made. All pages that have not been refrenced in a clock cycle are replaced. As the number of page frames increases, the number of page faults decreases and there is a reduction in the time taken for the program execution.