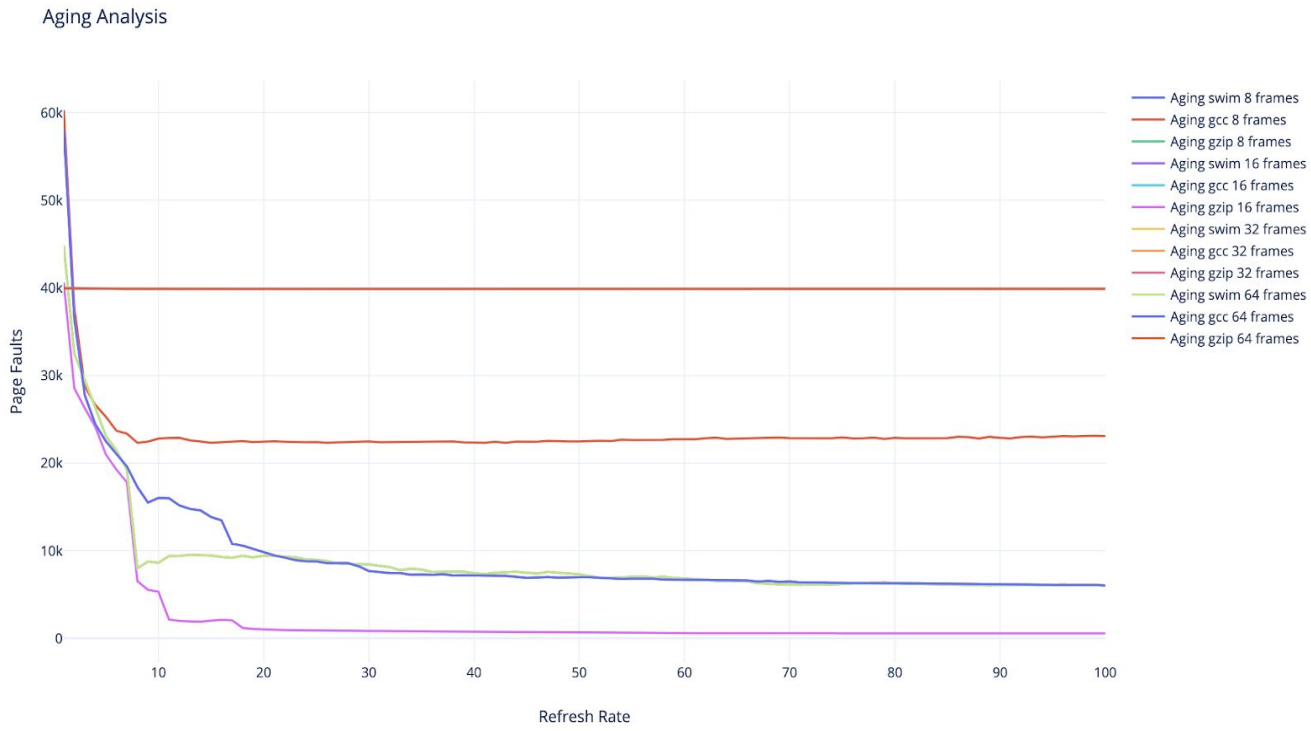
Name: Abbas Ahmed

Section: MW 3 PM - 4 PM – Dr. Sherif Khattab

Recitation: Tuesday 4 PM

CS1550 Project 3 - Write Up

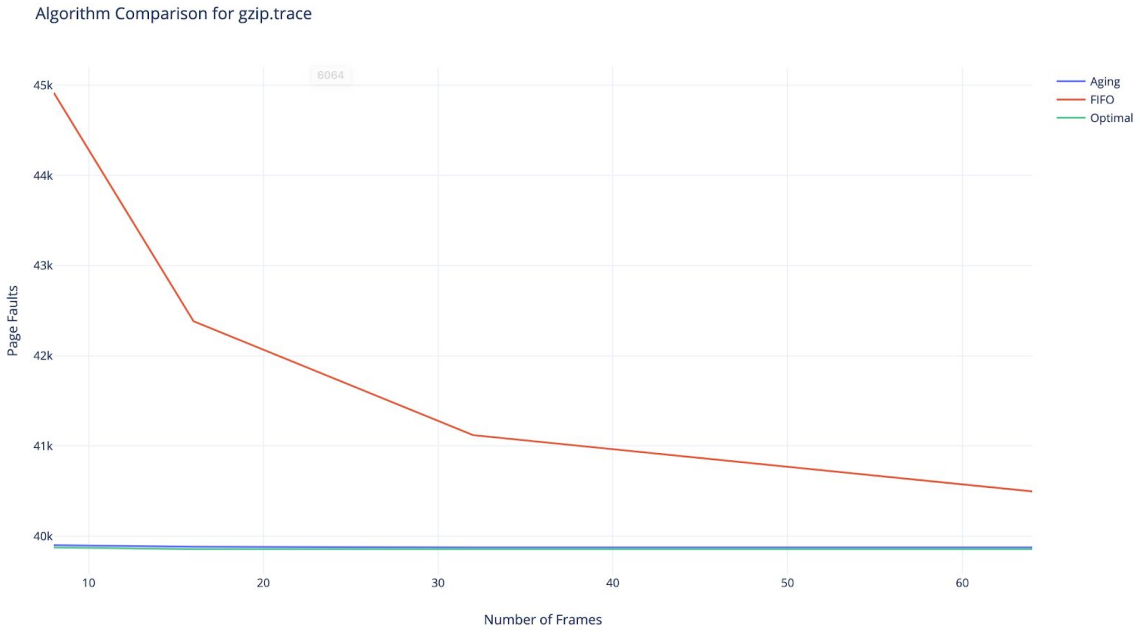
Optimal Value of Refresh Parameter

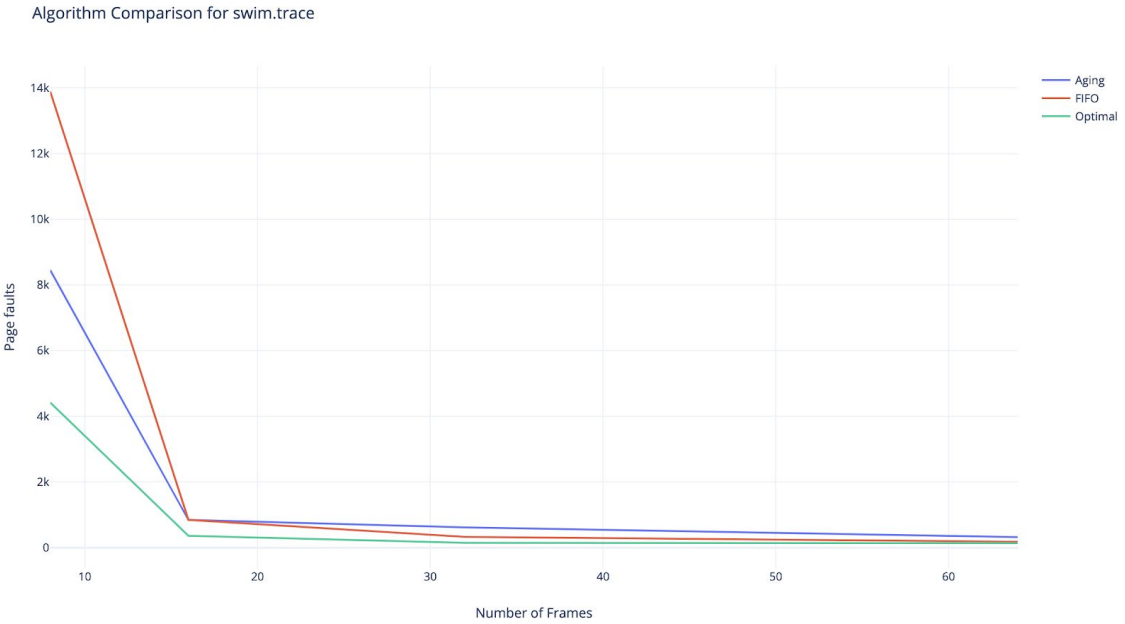


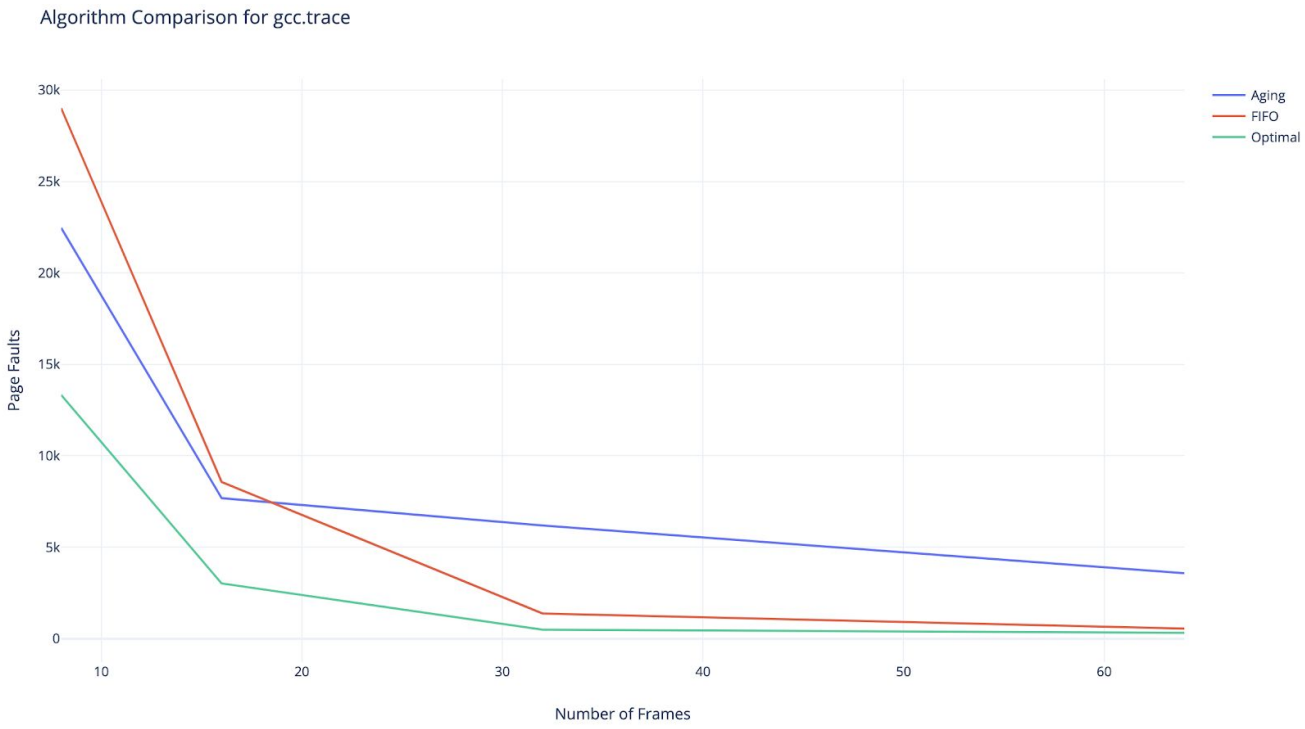
For the optimal value of the refresh parameter for Aging Algorithm, I chose the optimal value to be 20. I first ran the aging algorithm for every file for 4 different number of frames(8, 16, 32, 64), and for refresh parameter in the range of 1 to 100, both inclusive. After obtaining the results, and plotting in the graph shown above I realized that for around for each file and frame number, there's a steep decrease in page faults until the refresh rate is around 10 then continues to decrease until around when the refresh rate is 30. After that point, the rate of decrease in the number of page faults is very low. Hence, which is why I chose 30 as the optimal refresh rate. 30 seems to be a good refresh rate because of a higher refresh rate, the

aging will take place much slower and the number of page faults still decreases but very slowly and at which point is approximately close to what it was at the refresh rate of 30.

Algorithms Comparison



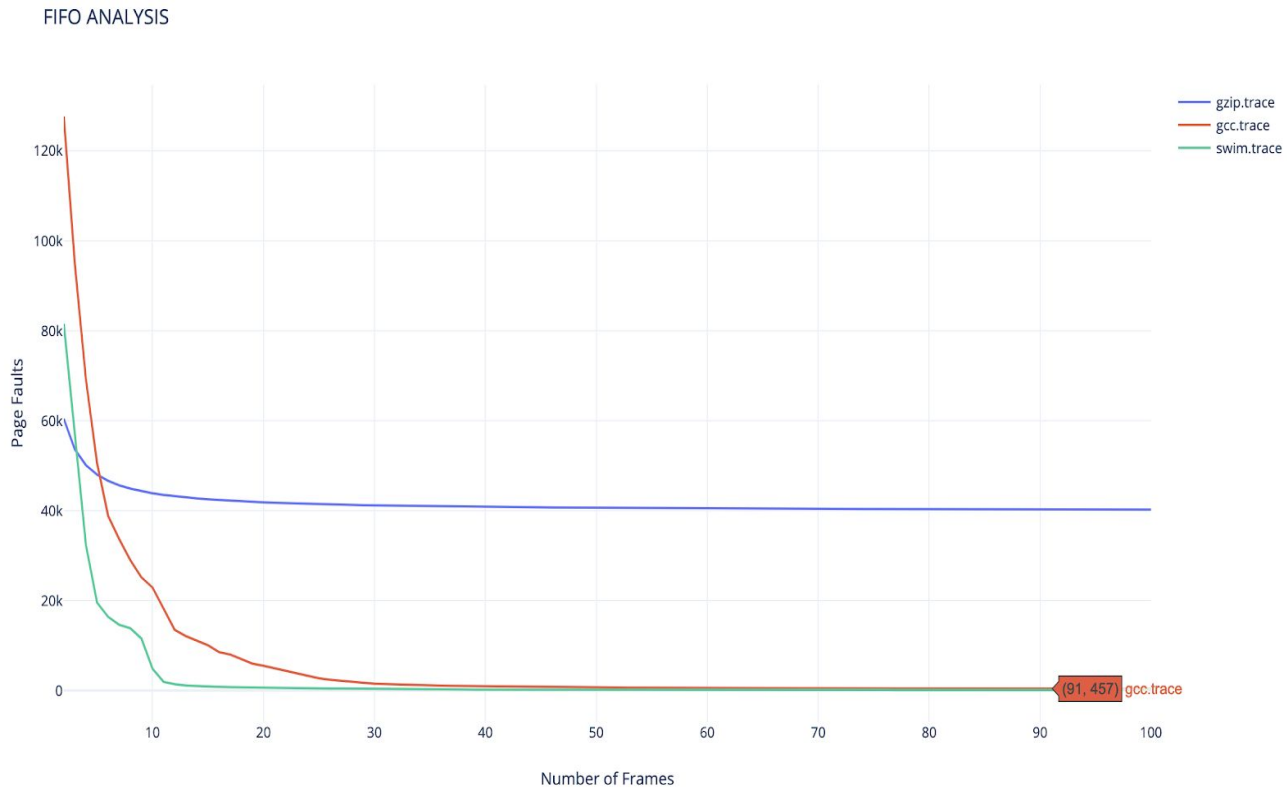


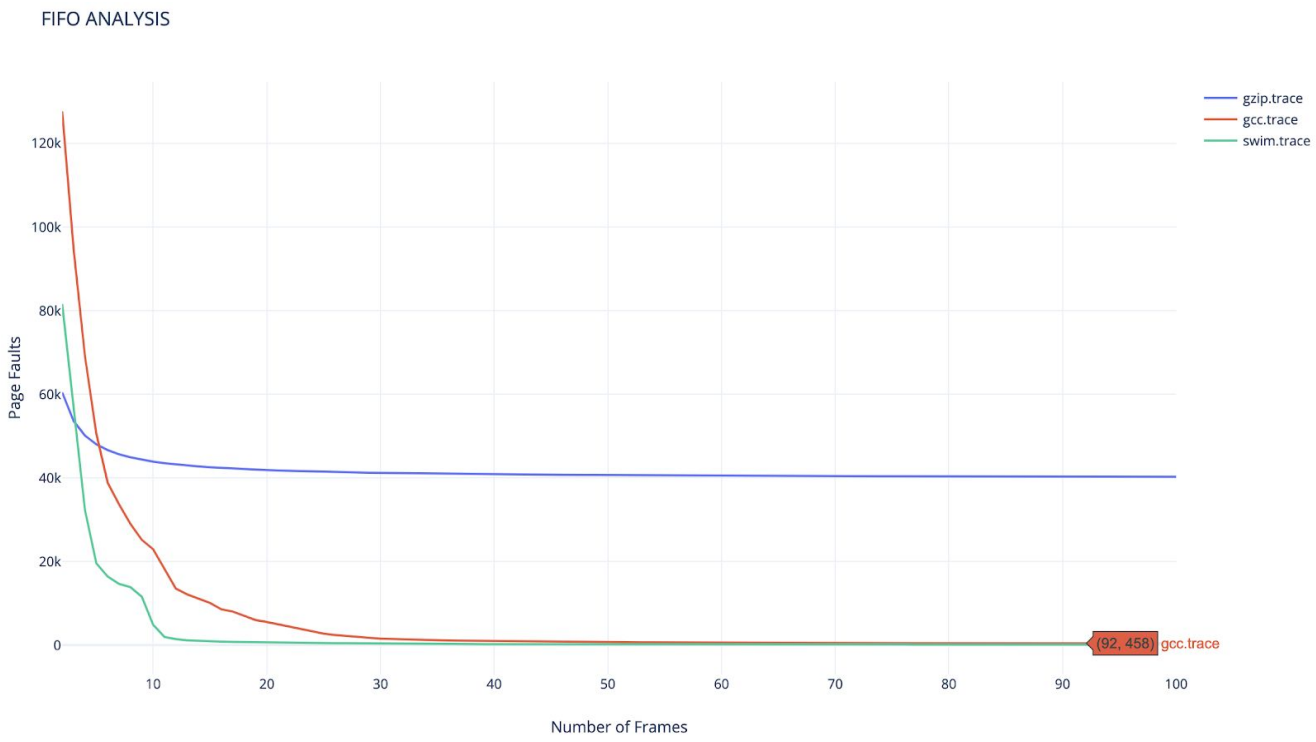


Above, I have provided 3 graphs, each containing the performance of each algorithm on each file. In all the cases, Optimal algorithm provides the best performance but there's a catch with this algorithm. The way the Optimal algorithm is implemented is that it needs to know which page is being requested in the future and at what times, which is impossible to do in real life since the OS has to deal with a request during run-time as given, and cannot predict which pages will be requested. Hence, the optimal algorithm can only provide us with a baseline for comparing our other algorithms for the purpose of this project. In some cases, such as for gzip.trace, the aging algorithm works as efficiently as the Optimal algorithm, and the FIFO algorithm performs better with a higher number of page frames but still far off from the optimal and aging. However, in swim.trace and gcc.trace we see that FIFO algorithm outperforms the aging algorithm in case of a higher number of frames and causes fewer page faults, and somewhat close to the optimal algorithm's performance. However, as the number of frames increases, the number of page faults amongst FIFO and aging decrease steadily and get closer

to the number of page faults the same as to optimal algorithm. Even though FIFO seems to have better results in the above analysis, we can have efficient and stable results with the aging algorithm and in aging algorithm higher number of frames eventually gives lower page faults(close to optimal). Thus, in real life, the OS would have a higher number of frames than what's been taken into account in this analysis, which is why the aging algorithm would be the best to use amongst these tree algorithms.

FIFO Analysis





I noticed a Belady's anomaly in the gcc.trace file at the intervals of [91, 93]. Belady's Anamoly points out that the increasing number of frames does not necessarily mean fewer page faults, and we can see here that at 91 frames for the gcc.trace file, the number of page faults is 457, and at 92 and 93 frames, the number of page faults is 458. Here, we see a decrease in performance of the FIFO algorithm. This concludes that increasing the number of frames does not necessarily result in fewer page faults, and which is why other pages replace algorithms from FIFO were studied further, to replace it.