Final Exam Report

Tanmay Goyal
AI20BTECH11021

While taking input, **we use -1 as an EOF character.** Since we do not know the number of references, we store them in a vector. Each reference is divided by the page size to get the page number that is being accessed. Finally, this vector, as well as the number of frames are passed as arguments for the FIFO, LRU and OPT page replacement algorithms.

For the FIFO algorithm, we maintain a queue for the elements, as well as an normal array containing the elements that are still in the queue. Every time we see a reference, we first check if it is in the list. If it is, we continue to the next reference, else we check if the queue is at it's maximum limit. If it is, we pop the first element, and ensure to change the element in the list as well. This is done by turning the list element to -1, to say that is no longer a valid reference. In case the queue is not at maximum limit, we simply add the reference to the queue and also update the list of elements in the queue.

For the LRU algorithm, we work with an array of size equal to the number of frames, and another array to keep track of which frame was the most recently accessed. Every time we see a reference, we first check if it is in the list. If it is, we continue to the next reference, and update it as most recent reference. Else we check if the array is at its maximum limit. If it is, we find the least recently used element and replace it. Else, we just add it to the array and update the reference to the most recent reference. The time of the references is stored as the index of reference in the reference string. The lesser the index, the higher probability of the page being selected as a victim.

For the OPT algorithm, we again work with an array of size equal to the number of frames. Every time we see a reference, we first check if it is in the list. If it is, we continue to the next reference. Else we check if the array is at its maximum limit. If it is, for each element in the array, we find its next reference in time. This reference in time is again stored as the index of reference in the reference string. The lesser this index, the lower the probability of the page being selected as a victim page. Once we know the time in the future when each page is going to be referenced, we select the victim page and replace it. In case the array is not at maximum limit, we just add it to the array.

Finally, we return the number of faults for each page replacement algorithm and print it to the terminal.