COEN_177_Assignment 3

FiFo:

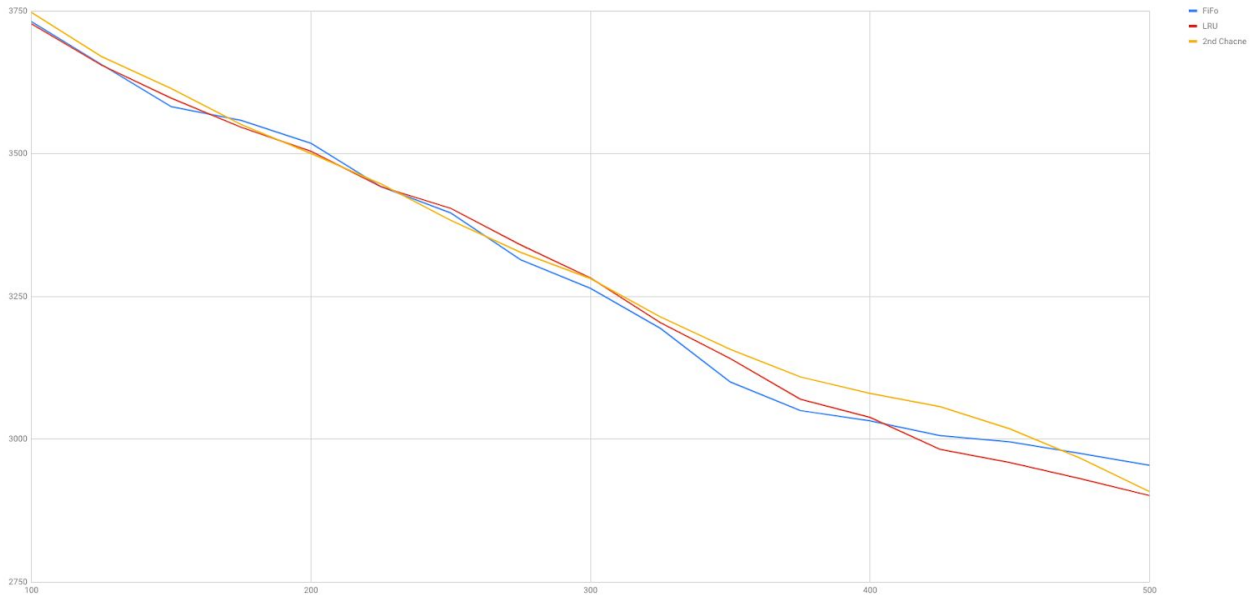
For Fifo algorithm, I created an array and an integer variable called fifo_loc that keeps track of the index of the oldest inserted page or the next empty spot in the array. After taking in a value, I sequentially search the array for that value. If it's found, nothing changes, but if it isn't, then the page at index fifo_loc is replaced, and fifo_loc is updated to the index of the new oldest page (which is the subsequent index, unless it has reached the end of the array). Lastly, the newly inserted page that caused the page fault is printed to standard out.

LRU

For LRU, I again used an array and when given a value, sequentially search it for that page. I used a variable 'found' to both determine whether or not the page was found and if it is, at what index was it found. Using this information, if the page is not found, I shift all items in the array right one index, and put the new page at index 0. This works because this way the most recently used pages will be near the front of the array and the least recently used will be at the end, and will be knocked out of the array by the right shift. Additionally, if the page is found, I copy its value, shift all pages in the page table array, from index 0 to found index, one to the right, and place the found page in location 0. This means the table that was just accessed is now the most recently used in the table. i

2nd Chance

For the second chance algorithm, I used 2 arrays: one as the page table and the other as a flag array for if a page has been referenced. First I set all reference bits to zero in the 'ref' array. I again perform a sequential search for a given page. If it is found, its reference bit is set to one. If not found in the table, then starting from the current index location (variable containing index of 'current' location) in the ref array, the program iterates through the array until it sees a zero. While it passes bits with ref=1, it sets these to zero, that way if it is a 1 next passthrough, that page must have been referenced, and if still zero, it was not. Once the first page with ref=0 is found, it is replaced with the newly added page, and the current 'pointer' is set to the index one above where the new page was added.



Graph of behaviors

They are all pretty even for awhile but in the end LRU and 2nd chance pull away as being more efficient, with LRU being the most efficient by just a bit.

input	FiFo	LRU	2nd Chacne
100	3731	3727	3747
125	3656	3655	3670
150	3582	3597	3614
175	3558	3546	3551
200	3518	3504	3500
225	3442	3442	3447
250	3396	3404	3383
275	3314	3340	3327
300	3264	3282	3281
325	3194	3204	3214
350	3100	3141	3157
375	3050	3070	3109
400	3032	3038	3080
425	3006	2982	3057
450	2995	2959	3018
475	2975	2931	2967
500	2954	2901	2908

Hit rates for three algorithms on data sets of 10,000 given different input page table sizes.