Nathan Ng

COEN 177 Lab 3

Wednesday 2:15

11/18/16

4 Page Replacement Algorithms

For my implementation of LRU and LFU, I used a deque with a circular, doubly-linked list. Also for the 4th page algorithm, the one of my choice, I chose to implement random that also uses the deque. New elements are entered at the front of the list for all three algorithms. For LRU and LFU, the most recent elements are kept at the front of the list. The random page replacement algorithm is effective because it randomly chooses which page to delete from the cache, a random index is selected using rand(), seeded by the time at runtime. For the 3rd page replacement algorithm, we had to implement the second-chance algorithm. I chose to do so through using the clock algorithm. This included a circular array of nodes with the data and also has a flag that determines if the node has already been visited or not.

All the algorithms were tested using the numbers.txt test file. Each algorithm was run with cache sizes of 10, 50, 100, 250, and 500 with 10000 lines read. For the random algorithm, each cache size was tested 3 times, with the average recorded. A table and graph of the data are in this folder under the Data and Graph excel spreadsheet. From this, we can tell that the LFU algorithm is the worst performing one of the all four. This also generated the most recorded page faults. The other three algorithms, LRU, second chance, and random were pretty much equal in performance and efficiency.