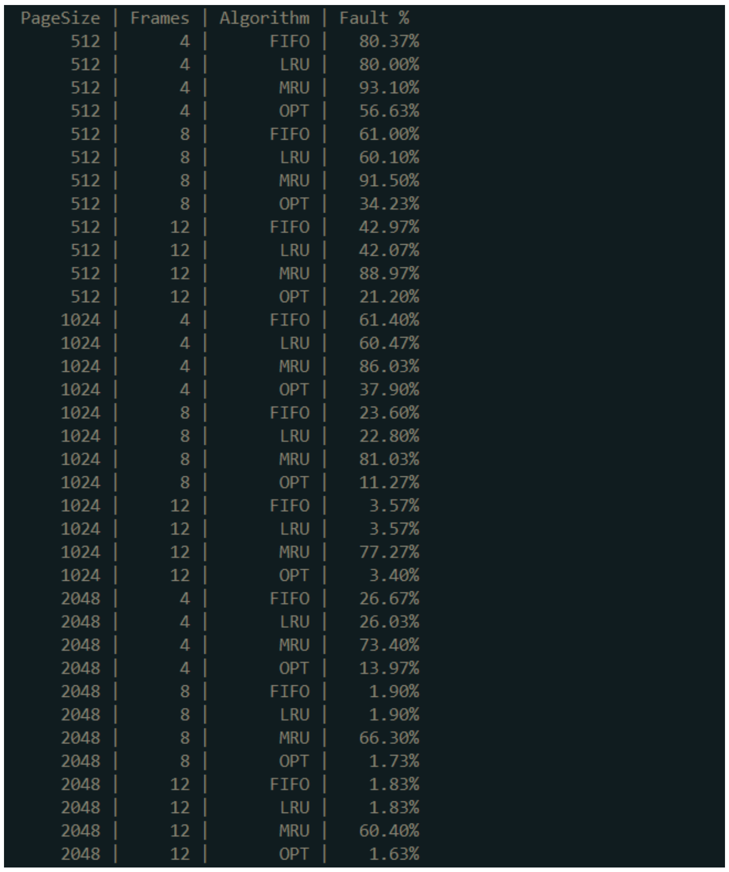
The virtual memory management problem required the simulation of four different replacement algorithms that are used in virtual memory management. I was given a file titled “Sample input data files.txt”, which contained 3000 memory addresses and was tasked to convert them into page numbers for different page sizes and then to run the four replacement algorithms on them. The replacement algorithms were FIFO (First In First Out), LRU (Least Recently Used), MRU (Most Recently Used), and OPT (Optimal). The purpose of running the algorithms on the different page numbers was to find and analyze the percentage of page faults faced with each algorithm to attempt to determine which algorithm was best in different scenarios. The page sizes tested were 512 bytes, 1024 bytes, and 2048 bytes. The frame counts tested were 4, 8, and 12 pages. When running the program’s executable file “VMEMMAN.jar”, the results are printed to the console, as shown below, and can be compared to the “Sample output file.txt” to ensure accuracy.



```
PageSize | Frames | Algorithm | Fault %
     512 |      4 |      FIFO |   80.37%
     512 |      4 |       LRU |   80.00%
     512 |      4 |       MRU |   93.10%
     512 |      4 |       OPT |   56.63%
     512 |      8 |      FIFO |   61.00%
     512 |      8 |       LRU |   60.10%
     512 |      8 |       MRU |   91.50%
     512 |      8 |       OPT |   34.23%
     512 |     12 |      FIFO |   42.97%
     512 |     12 |       LRU |   42.07%
     512 |     12 |       MRU |   88.97%
     512 |     12 |       OPT |   21.20%
    1024 |      4 |      FIFO |   61.40%
    1024 |      4 |       LRU |   60.47%
    1024 |      4 |       MRU |   86.03%
    1024 |      4 |       OPT |   37.90%
    1024 |      8 |      FIFO |   23.60%
    1024 |      8 |       LRU |   22.80%
    1024 |      8 |       MRU |   81.03%
    1024 |      8 |       OPT |   11.27%
    1024 |     12 |      FIFO |    3.57%
    1024 |     12 |       LRU |    3.57%
    1024 |     12 |       MRU |   77.27%
    1024 |     12 |       OPT |    3.40%
    2048 |      4 |      FIFO |   26.67%
    2048 |      4 |       LRU |   26.03%
    2048 |      4 |       MRU |   73.40%
    2048 |      4 |       OPT |   13.97%
    2048 |      8 |      FIFO |    1.90%
    2048 |      8 |       LRU |    1.90%
    2048 |      8 |       MRU |   66.30%
    2048 |      8 |       OPT |    1.73%
    2048 |     12 |      FIFO |    1.83%
    2048 |     12 |       LRU |    1.83%
    2048 |     12 |       MRU |   60.40%
    2048 |     12 |       OPT |    1.63%
```

Figure 1: A screenshot of the terminal after running the executable VMEMMAN.jar.

There are several clearly visible trends in the data. The MRU algorithm tends to have the highest fault rate at any page size and frame count. Conversely, OPT consistently has the lowest fault rate in all cases. While arguably negligible, LRU tends to be slightly more efficient than, or at worst equally efficient to, the FIFO algorithm in all cases. It is also reasonable to assume an inverse relationship between the number of page frames and page faults; consistently, when the number of page frames increases, the page faults decrease.

Page size and the number of frames have clear impacts on the number of page faults. It is evident that, for example, in the runs for the page size of 512 bytes, as the number of frames increases, the faults decrease. With 4 frames, we see FIFO hitting a fault 80.37% of the time, at 8 frames, we see it hit only 61.00% of the time, and at 12 we see even fewer hits at 42.97%. This is consistent across all runs and all page sizes. Regarding page size, the FIFO run at 512 bytes and 4 frames is 80.37% whereas at 1024 bytes and 4 frames, it only hits 61.40% and even less at 2048 bytes and 4 frames, with a fault rate of 26.67%. It stands to reason that the page size and frame number improve fault rates, this is backed by how page sizes and frame numbers affect memory allocation. When the page is larger or has more frames, there are fewer references to pages and, as such, fewer faults.

The OPT algorithm, expectedly, always has the lowest fault rate and performs incredibly well with large page sizes and high frame counts. At a page size of 2048 bytes and 12 frames, its fault rate dips as low as only 1.63%. LRU and FIFO have similar results, but the LRU algorithm is more successful in some cases, such as, when running at a page size of 512 bytes and a frame count of 4, LRU hits marginally fewer faults at 80.00% whereas FIFO hits a fault 80.37% of the time. LRU and FIFO both perform according to their expected behavior as a middle ground between MRU and OPT in performance. The MRU algorithm performed as expected as well, consistently hitting the most faults and overall acting as the worst-performing algorithm. MRU has high fault rates even when provided the best chance for success. This can be clearly seen at a page size of 2048 bytes and 12 pages; the MRU algorithm still hits a fault 60.40% of the time.

Overall, the simulation resulted in the confirmation of what was expected of each replacement algorithm’s performance. The OPT algorithm has the lowest fault percentage, making it the best case scenario. LRU and FIFO performed similarly, with LRU being slightly better and serving as the most effective algorithm for practical use. MRU, as expected, performed the worst in every instance with a high fault rate, even in the best of circumstances, making MRU the least ideal algorithm tested. A higher frame count and page size lead to consistently better results, which supports the idea of a relationship between fewer page references and fewer faults. The results of the experiment perfectly aligned with what was expected and reinforced which replacement algorithms are ideal for different circumstances.