Page Replacement Algorithms

Written By:

Shariya George

CS451 (Operating Systems Course)

INTRODUCTION: Understanding algorithms can be very confusing and difficult to understand at times. Thus, I created this explanation of understanding one my favorite types of algorithms in Computer Science : The Page Replacement Algorithm.

Our professor assigned us with the task of explaining the 3 different page replacement algorithms: **FIFO Page Replacemnt Algorithm, Optimal Page Replacement Algorithm, and LRU Page Replacement Algorithm.**

Through this task hopefully you will be able to understand Page Replacement Algorithms a little more !!

ENJOY !!

Page Replacement Algorithms

Task: *Choose your own sequence of 12 pages, using page numbers 1 through 5 (you must use all 5 numbered pages at least once in your sequence). Using 3 frames for the process, apply 3 different page replacement algorithms; two must be the optimal and LRU (the third is your choice).*

Legend: X = Page Faults h = Hits

Sequence of Pages: 1 2 4 3 3 5 1 3 5 2 4 2

FIFO Page Replacemnt Algorithm

Ref. String	1	2	4	3	3	5	1	3	5	2	4	2
f1	1	1	1	3	3	3	3	3	3	2	2	2
f2		2	2	2	2	5	5	5	5	5	4	4
f3			4	4	4	4	1	1	1	1	1	1
Page faults/ Hits	x	x	x	x	h	x	x	h	h	x	x	h

Total Page Faults: 8 **Total Hits:** 4

FIFO Description:

In this FIFO algorithm, we have three frames (f1, f2, f3) and the reference string: 1 2 4 3 3 5 1 3 5 2 4 2. In First IN, First Out, we are looking at the first page that came in. Thus, **the oldest page gets swapped out.**

- Page 1 is loaded into frame 1. Page 1 was not available in the main memory. Thus, we have our first page fault.
- Second, page 2 is loaded into fame 2. Page 2 was not available in the main memory, so we have our 2nd page fault.
- Third, page 4 is loaded into frame 3. Page 4 was not available in our main memory, so we have our 3rd page fault.
- Fourth, Page 3 is not available in our main memory, so the operating system must swap out one page from main memory to secondary memory.

Using the FIFO algorithm, the oldest page (the page that first came in) gets swapped out. Page 1 is the oldest page. Thus, page 1 gets swapped out for page 3. This is the fourth page fault.

- Fifth, page 3 is available in our main memory, so there is no swapping being done and no page fault. This is a hit.
- Next, is page number 5. Page 5 is not available in our main memory, so this is our fifth page fault. The operating system will need to swap out page 2 for page 5. Using the FIFO algorithm, page 2 comes first, followed by page 3 and page 4. In other words, page 2 is our oldest page.
- Next, the CPU requests page 1. Page 1 is not in the main memory, so we have our sixth page fault. Using FIFO, we will swap page 4 out (the oldest page) and replace it with page 1.
- Next is page 3. Page 3 is already available, so this is a hit.
- Next is page 5. Page 5 is already available, so this is a hit. No swapping is taking place.
- Then comes page 2. Page 2 is not available in the main memory, so this is the 7th page fault. Using the FIFO algorithm, page 3 gets swapped out for page 2.
- The next required page is page 4. Page 4 is not available in the main memory, so this is our 8th page fault. Using the FIFO algorithm, page 5 gets swapped out for page 4 and loaded into the frame.
- Lastly, page 2 is required. Page 2 is available in the main memory, so this is a hit.

Optimal Page Replacement Algorithm

Legend: X = Page Faults h = Hits

Sequence of Pages: 1 2 4 3 3 5 1 3 5 2 4 2

Ref. String	1	2	4	3	3	5	1	3	5	2	4	2
f1	1	1	1	1	1	1	1	1	1	2	2	2
f2		2	2	2	2	5	5	5	5	5	5	5
f3			4	3	3	3	3	3	3	3	4	4
Page faults/ Hits	x	x	x	x	h	x	h	h	h	x	x	h

Total Page Faults: 7 **Total Hits:** 5

Optimal Description:

In this Optimal Page Replacment algorithm, we have three frames (f1, f2, f3) and use the reference string: 1 2 4 3 3 5 1 3 5 2 4 2. We will be replacing the page that will not be used for the longest amount of time in the future.

- Page 1 is loaded into frame 1. Page 1 was not available in the main memory. Thus, we have our first page fault.
- Page 2 is loaded into frame 2. Page 2 is not available in the main memory, so we have our 2nd page fault.
- Page 4 is loaded into frame 3. Page 4 is not available in our main memory, so we have our 3rd page fault.
- Next, we have page 3. Page three is not currently available in our main memory, so we have our fourth page fault. Using the optimal algorithm, we see that of the three pages (1, 2, and 4), page 4 will not be needed for the longest. Thus, page 4 gets swapped in frame 3.
- Next, the CPU requests page 3. Page 3 is already available in the main memory, so this is our first hit.
- Next, is page number 5. Page 5 is not available in our main memory, so this is our 5th page fault. The operating system will need to swap out page 2 for

page 5. Using the optimal algorithm, out of the pages (1, 2, and 3), page 2 will not be used for the longest time.

- Next is page 1. Page 1 is already available in the main memory, so this is a hit.
- Next is page 3. Page 3 is already available, so this is a hit. No swapping is taking place.
- Next is page 5. Page 5 is already available, so this is a hit.
- Next is page 2. Page 2 is not available in the main memory, so we have our 6th page fault. Out of the pages 1, 5, and 3), none of these pages will be requested in the future, so we can replace them with any. I applied the FIFO page replacement algorithm. Out of pages (1, 5 and 3) page 1 is the oldest. Thus, page 2 gets swapped in frame 1.
- Next is page 4. Page 4 is not available in the main memory, so we have our 7th page fault. Out of the pages (2, 5, and 3), only page 2 will be requested in the future, so we can replace the page with any. I applied the FIFO page replacement algorithm. Out of pages (2, 5, and 3), page 3 is the oldest. Thus, page 4 gets swapped in frame 3.
- Last, page 2 is requested. Page 2 is already available in the main memory, so this is a hit. No swapping is taking place.

LRU Page Replacement Algorithm

LRU Description:

The Least Recently Used Page Replacment algorithm, replaces the page that has not been used for the longest amount of time.

Legend: X = Page Faults h = Hits

Sequence of Pages: 1 2 4 3 3 5 1 3 5 2 4 2

Ref. String	1	2	4	3	3	5	1	3	5	2	4	2
f1	1	1	1	3	3	3	3	3	3	3	4	4
f2		2	2	2	2	5	5	5	5	5	5	5
f3			4	4	4	4	1	1	1	2	2	2
Page faults/ Hits	x	x	x	x	h	x	x	h	h	x	x	h

Total Page Faults: 8 **Total Hits:** 4

Stack Demonstration:

The most recently used (MRU) page is always at the top of the stack. LRU is always at the bottom of the stack.

MRU	1	2	4	3	3	5	1	3	5	2	4	2
		1	2	4	4	3	5	1	3	5	2	4
LRU			1	2	2	4	3	5	1	3	5	5

- Page 1 is loaded into frame 1. Page 1 was not available in the main memory. Thus, we have our first page fault. Page 1 is MRU, so it is at the top of the stack.
- Second, page 2 is loaded into fame 2. Page 2 was not available in the main memory, so we have our 2nd page fault. Page 2 is at the top of the stack, and Page 1 gets pushed down.

- Page 4 is loaded into frame 3. Page 4 was not available in our main memory, so we have our 3rd page fault. Page 4 is MRU, so it is at the top of the stack, and Page 2 gets pushed down, and Page 1 gets pushed down to the bottom of the stack. Page 1 is now the LRU.
- Next, page 3 is swapped into frame one. Page 3 was not available in the main memory, so we have our fourth page fault. Page 3 gets pushed to the top of the stack (MRU). Page 4 is pushed down, and Page 2 gets pushed down to the bottom of the stack. Page 2 is now the LRU.

- Next is page 3. Page 3 is already available, so this is a hit. No swapping is taking place. Also, the stack stays the same.

- Next, page 5 is swapped into frame two. Page 5 was not available in the main memory, so we have our fifth page fault. Page 5 gets pushed to the top of the stack (MRU). Page 3 is pushed, and Page 4 gets pushed down to the bottom of the stack. Page 4 is now the LRU.

- Next, page 1 is swapped into frame three. Page 1 was not available in the main memory, so we have our sixth page fault. Page 1 gets pushed to the top of the stack (MRU). Page 3 gets pushed down to the bottom of the stack, then page 5. Page 3 is now the LRU.

- Next is page 3. Page 3 is already available, so this is a hit. No swapping is taking place. However, the stack changes. Page 3 is now at the top of the stack (MRU). Page 5 gets pushed down to the bottom of the stack, then page 1. Page 5 is now the LRU.

- Next is page 5. Page 5 is already available, so this is a hit. No swapping is taking place. However, the stack changes. Page 5 is now at the top of the stack, (MRU). Page 1 gets pushed down to the bottom of the stack, then page 3. Page 1 is now the LRU.

- Next, page 2 is swapped into frame three. Page 2 was not available in the main memory, so this is the seventh page fault. Page 2 gets pushed to the top of the stack (MRU). Page 3 gets pushed down to the bottom of the stack, then page 5. Page 3 is now the LRU.

- Page 4 is loaded into frame 1. Page 4 was not available in our main memory, so this is the eighth page fault. Page 4 is MRU, so it is at the top of

the stack, and Page 2 gets pushed down, and Page 5 gets pushed down to the bottom of the stack. Page 5 is now the LRU.

- Last, page 2 is requested. Page 2 is already available in the main memory, so this is a hit. However, the stack changes. Page 2 is now at the top of the stack (MRU). Page 5 gets pushed down to the bottom of the stack, then page 4. Page 5 is now the LRU.

Conclusion: The Optimal Page replacement algorithm is the best for the sequence: 1 2 4 3 3 5 1 3 5 2 4 2. There were seven page faults and five hits in total. This was the most efficient page replacement algorithm, with the fewest page faults and the most hits.