MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Operating Systems

Mini Project Report

TITLE
Page Replacement Algorithms

Submitted by:

Name	Reg No	Section	Roll No
Yashas Kamath	200905132	D	20
Preetham M	200905138	D	21
Sarthak Arora	200905145	D	22
Siddhartha Bhat	200905010	D	3

ABSTRACT

Page replacement algorithms choose pages to swap out from the memory when a new page needs memory for allocation. A cluster of algorithms have developed for page replacement. Each algorithm has the objective to minimize the number of page faults. With minimum page faults, the performance of the process is increased.

Keywords: - Memory Management, Virtual memory, Page-fault.

PROBLEM STATEMENT

To reduce the number of page faults to efficiently utilize main memory using page replacement algorithms.

Description:

- Demand paging suggests keeping all pages of the frames in the secondary memory until they are required. In other words, it says that do not load any page in the main memory until it is required.
- To solve two major problems to implement demand paging, we must develop:
    - Frame-Allocation Algorithm
    - Page-Replacement Algorithm
- Here, we are taking up some of the important algorithms that have been developed to reduce the page-fault rate.

INDIVIDUAL CONTRIBUTIONS

YASHAS KAMATH:

1. LRU – Least Recently Used
2. 2Q LRU – 2 Queue Least Recently Used

PREETHAM M:

1. NRU – Not Recently Used
2. LDF – Longest Distance First

B L SIDDHARTHA BHAT:

1. OPTIMAL
2. LFU – Least Frequently Used

SARTHAK ARORA:

1. FIFO – First In, First Out
2. CLOCK

ALGORITHMS AND THEIR METHODOLOGIES

LRU PAGE REPLACEMENT ALGORITHM:

- A Least Recently Used (LRU) Cache organizes items in order of use, allowing you to quickly identify which item hasn't been used for the longest amount of time.
- A Least Recently Used (LRU) Cache organizes items in order of use, allowing you to quickly identify which item hasn't been used for the longest amount of time.

Simple LRU

2Q LRU PAGE REPLACEMENT ALGORITHM:

- Sub-optimality in evictions: LRU algorithm works with a single dimension - recency - as it removes the pages from the buffer on the basis of recent accesses. Since it does not really consider any other factor, it can actually evict a warmer page and replace it with a colder one - a page that could and would be accessed just once.

- 2Q addresses the above-illustrated issues by introducing parallel buffers and supporting queues. Instead of considering just recency as a factor, 2Q also considers access frequency while making the decision to ensure the page that is really warm gets a place in the LRU cache. It admits only hot pages to the main buffer and tests every page for a second reference.

- The golden rule that 2Q is based on is - *Just because a page is accessed once does not entitle it to stay in the buffer. Instead, it should be decided if it is accessed again then only keep it in the buffer.*

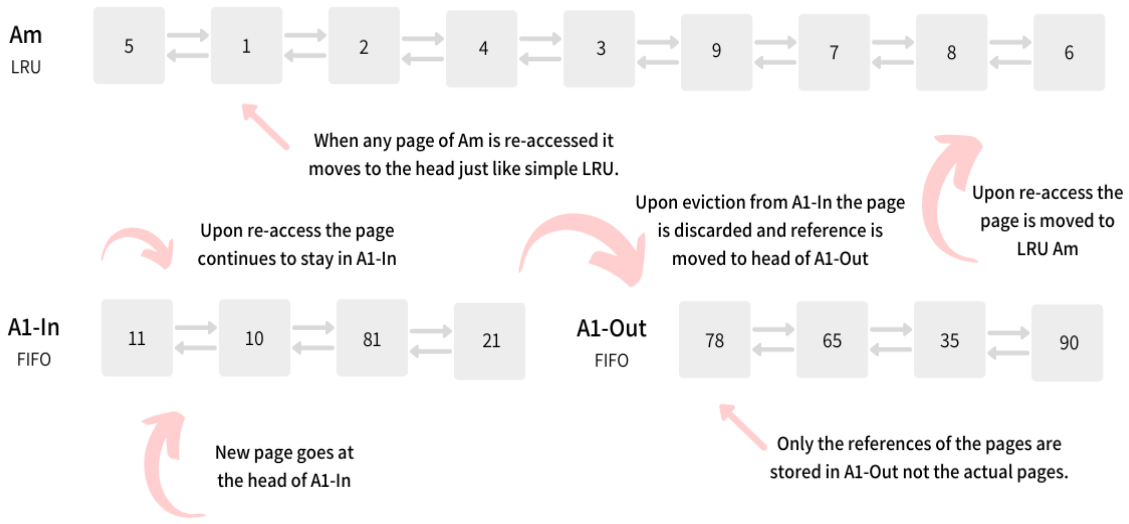
- 2Q algorithm works with two buffers: the primary LRU buffer - Am and a secondary FIFO buffer - A1. New faulted pages first go to the secondary buffer A1 and then when the page is referenced again, it moves to the primary LRU buffer Am.

This ensures that the page that moves to the primary LRU buffer is hot and indeed requires to be cached.

- If the page residing in A1 is never referenced again, it eventually gets discarded, implying the page was indeed cold and did not deserve to be cached. Thus, this 2Q provides protection against the two listed sub-optimality of the simple LRU scheme by adding a secondary buffer and testing pages for a second reference.

- The 2Q algorithm splits the secondary buffer A1 into two buffers A1-In and A1-Out, where the new element always enters A1-In and continues to stay in A1-In till it gets accesses ensuring that the most recent first accesses happen in the memory.

- Once the page gets old, it gets thrown off the memory but its disk reference is stored in the A1-Out buffer. If the page, whose reference is, residing in A1-Out is accessed again the page is promoted to Am LRU implying it indeed is a hot page that will be accessed again and hence required to be cached.



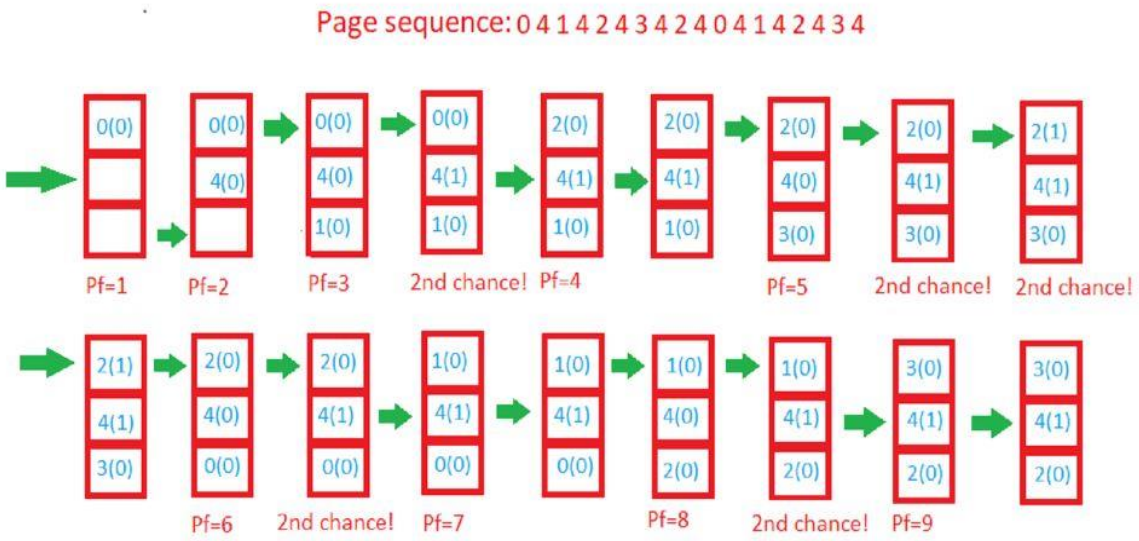
2Q - Full Version

Second chance (Clock)

- Apart from LRU, OPT and FIFO page replacement policies, we also have the second chance/clock page replacement policy.

- In the Second Chance page replacement policy, the candidate pages for removal are considered in a round robin matter, and a page that has been accessed between consecutive considerations will not be replaced.

- The page replaced is the one that, when considered in a round robin matter, has not been accessed since its last consideration.



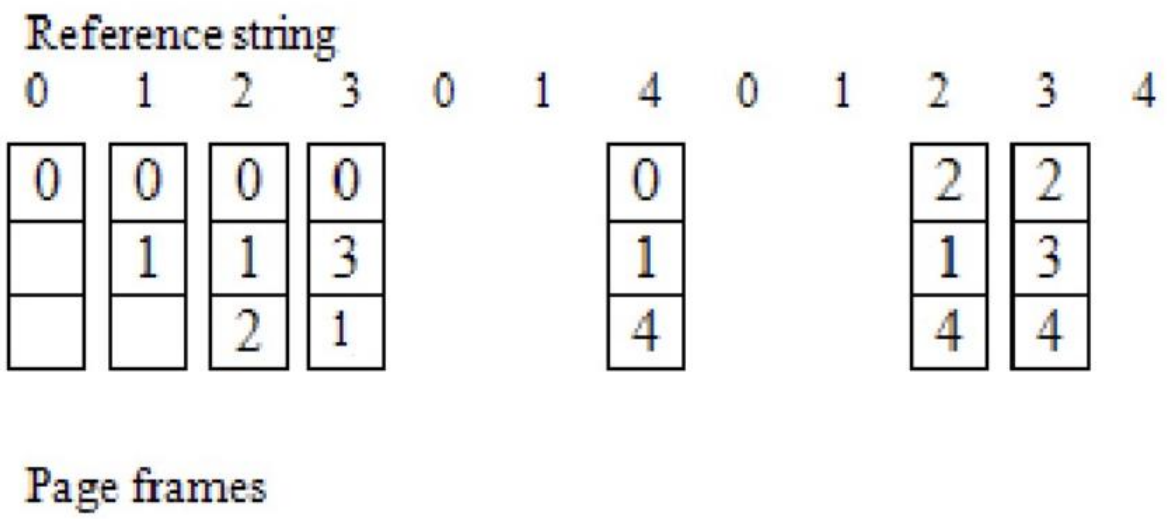
FIFO Page replacement

- This is the simplest page replacement algorithm.
- In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue.
- When a page needs to be replaced, the page in the front of the queue is selected for removal.

0	2	1	6	4	0	1	0	3	1	2	1
0	0	0	0	4	4			4	4	2	
	2	2	2	2	0		hit	0	0	0	
		1	1	1	1	hit		3	3	3	
			6	6	6			6	1	1	hit

LDF PAGE REPLACEMENT ALGORITHM

- The basic idea behind this algorithm is Locality of Reference, and locality is based on the distance not on the used references.
- The page which is on the longest distance from the current page is replaced.
- Distance calculation:
  - For the calculation of distance of a page from the current page, arrange page reference numbers in circular form and count how many number of pages it is away from the current page in both directions, clockwise and anti-clockwise.
  - From these two distances minimum distance will be taken.
- If two pages are on same distance then the page which is next to current page in anti-clockwise rotation will get replaced.

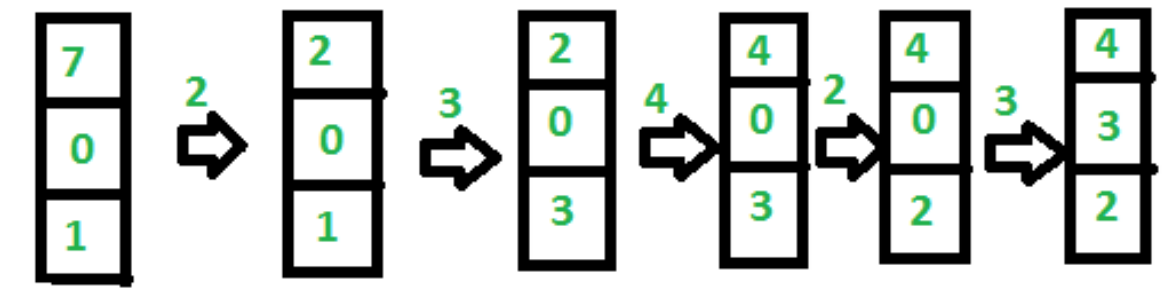


NRU Page Replacement Algorithm:

- NRU page replacement algorithm favours keeping pages in memory that have been recently used.
- Working principle:
  - when a page is referenced, a referenced bit is set for that page, marking it as referenced.
  - when a page is modified (written to), a modified bit is set.
- When a needs to be replaced, operating system divides page into 4 classes:
  - 3. Referenced, modified
  - 2. Referenced, not modified
  - 1. Not referenced, modified
  - 0. Not referenced, not modified

Reference String-

7 0 1 2 0 3 0 4 2 3

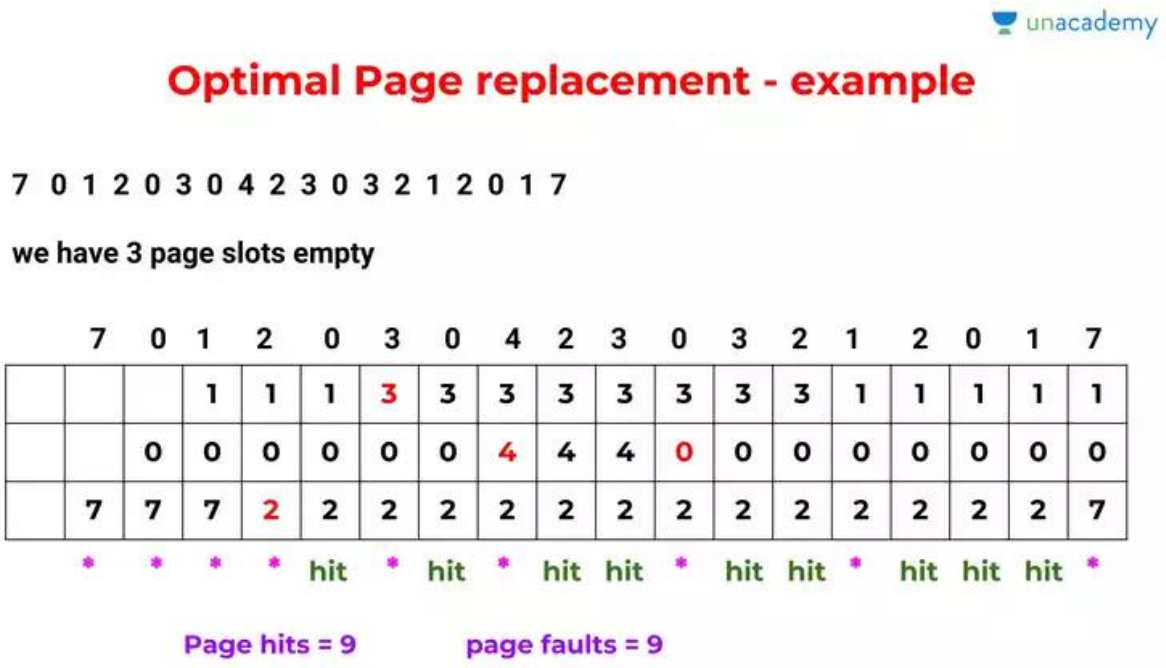


Page Fault = 8

Fault Rate = 8/10 = 4/5

OPTIMAL Page Replacement Algorithm:

- This algorithm works on the principle that when a page needs to be swapped in, the operating system swaps out the page who's next use will occur farthest in future.
- This algorithm is not practical and cannot be implemented in the general-purpose operating systems.
- However, it is possible to implement it on the second run using page reference information collected on the first run.
- It can offer near optimal performance and can be used as an reference for other page replacement algorithms.



LFU Page Replacement Algorithm:

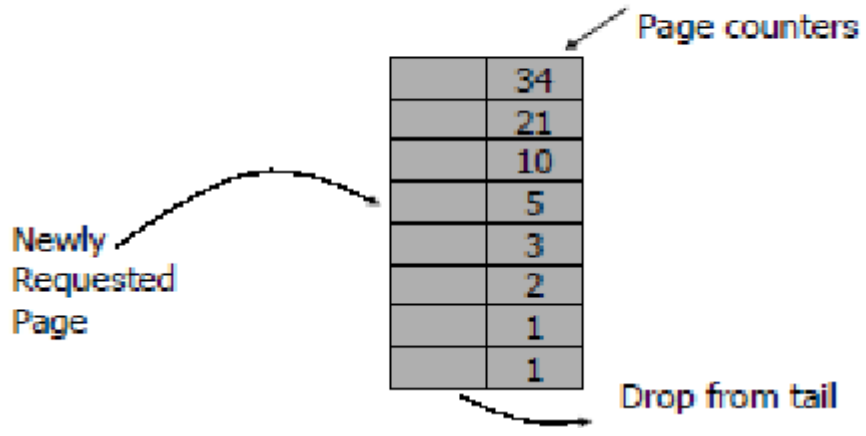
- LFU is a page replacement policy in which least frequently used pages are replaced.
- If the frequency of the pages are same, then pages are replaced in the fifo order.
- Operating system keeps track of all pages in the memory in a queue.
- This algorithm uses two data structures, one to store the pages and another keeps track of frequency of the pages that has occurred

1	2	3	4	5	1	3	1	6	3	2	3

1	1	1	1	5	5	5	5	5	5	2	2
	2	2	2	2	1	1	1	1	1	1	1
		3	3	3	3	3	3	3	3	3	3
			4	4	4	4	4	6	6	6	6
M	M	M	M	M	M	H	H	M	H	M	H

M = Miss
H = Hit

Least Frequently Used



REFERENCES

- https://en.wikipedia.org/wiki/Page_replacement_algorithm
- https://www.researchgate.net/
- https://www.academia.edu/
- https://www.geeksforgeeks.org/not-recently-used-nru-page-replacement-algorithm
- https://arpitbhayani.me/blogs/2q-cache
- https://github.com/EnergeticQuanta17/OS_MiniProject_Final
- Textbook:

  Operating System Concepts by Abraham Silberschatz, Greg Gagne, Peter B. Galvin 9th Edition