Page Replacement

→ Page replacement completes separation between logical memory & physical memory - a large virtual memory can be provided on a smaller physical memory.

→ Required when a page fault occurs & there is no free frame available in the physical memory.

→ Page replacement steps are:

1- Find location of the desired page on disk.

2- Find a free frame:

- if there is a free frame, use it.
- if there is no free frame, use a page replacement algorithm to select a victim frame.

3- Read the desired page into the (newly) free frame. Update page & frame table.

4- Restart the process.

→ To implement demand paging two algorithms are required :

* Frame - allocation algorithm - if we have multiple processes in memory, how many frames should be allocated to each process.

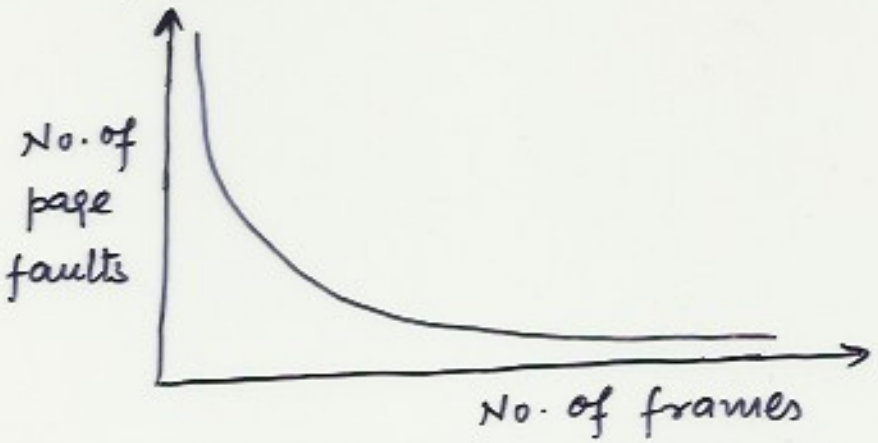
* Page - Replacement algorithm - which frame should be replaced, when there is page fault.

Page Replacement Algorithm :

→ Algorithm should lower the page fault rate.

→ Evaluate algorithms by running it on a particular ~~reference~~ string of memory references (reference string) & computing the no. of page faults on that string.

→ More no. of frames, lesser no. of page faults.



(1-) FIFO Algorithm :

→ Replace a page, which has been brought first into memory.

Reference string :

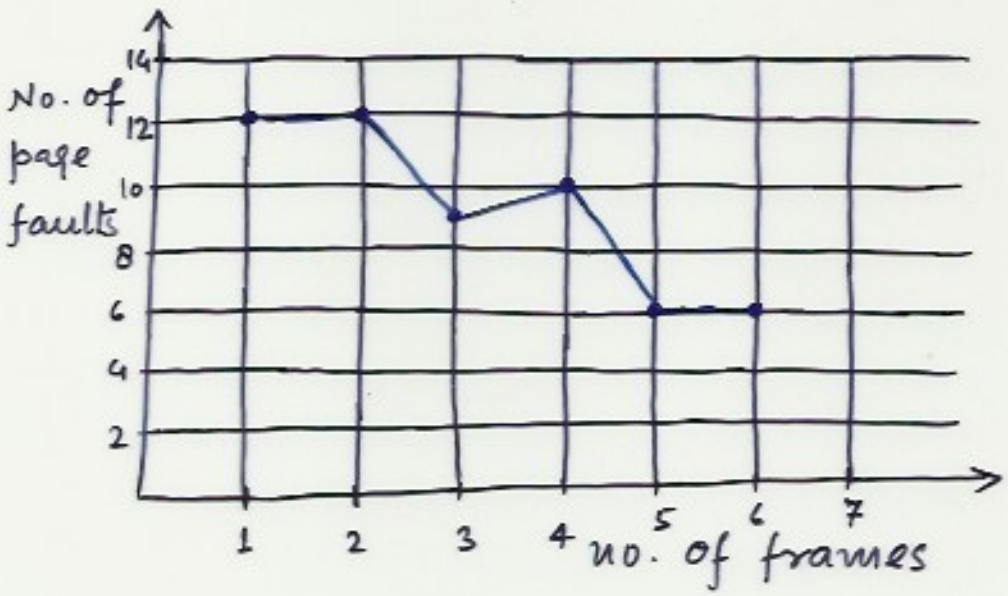
7	0	1	2	0	3	0	4	2	3	0	3	2	1	2
7	7	7	2		2	2	4	4	4	0			0	0
	0	0	0		3	3	3	2	2	2			1	1
		1	1		1	0	0	0	3	3			3	2

→ Algorithm is easy to understand.

→ Performance is not always good.

→ For reference string - 1 2 3 4 1 2 5 1 2 3 4 5

curve for page faults Vs available frame is :



→ No. of page faults is more when 4 frames were there than no. of page faults when there were three frames.

→ Effect is called Belady's anomaly.

Optimal Algorithm :

→ Replace the page that will not be used for longest period of time.

Reference string

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2

Total page faults = 8

→ Drawback :

- Requires future knowledge of reference string.

→ Used for measuring how well other algorithms perform.

Least Recently Used (LRU) algorithm :

→ Replace the page that has not been used for longest period of time.

Reference string

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2
7	7	7	2		2		4	4	4	0			1	
	0	0	0		0		0	0	3	3			3	
		1	1		3		3	2	2	2			2	

Total page faults : 10

→ Problem is how to find least recently used frame.

→ Two methods are there :

- Counter
- Stack

→ Using Counter

- Each entry in page table has a time of use field.
- Whenever a page is referenced, this field value is set equal to a counter value associated with CPU.

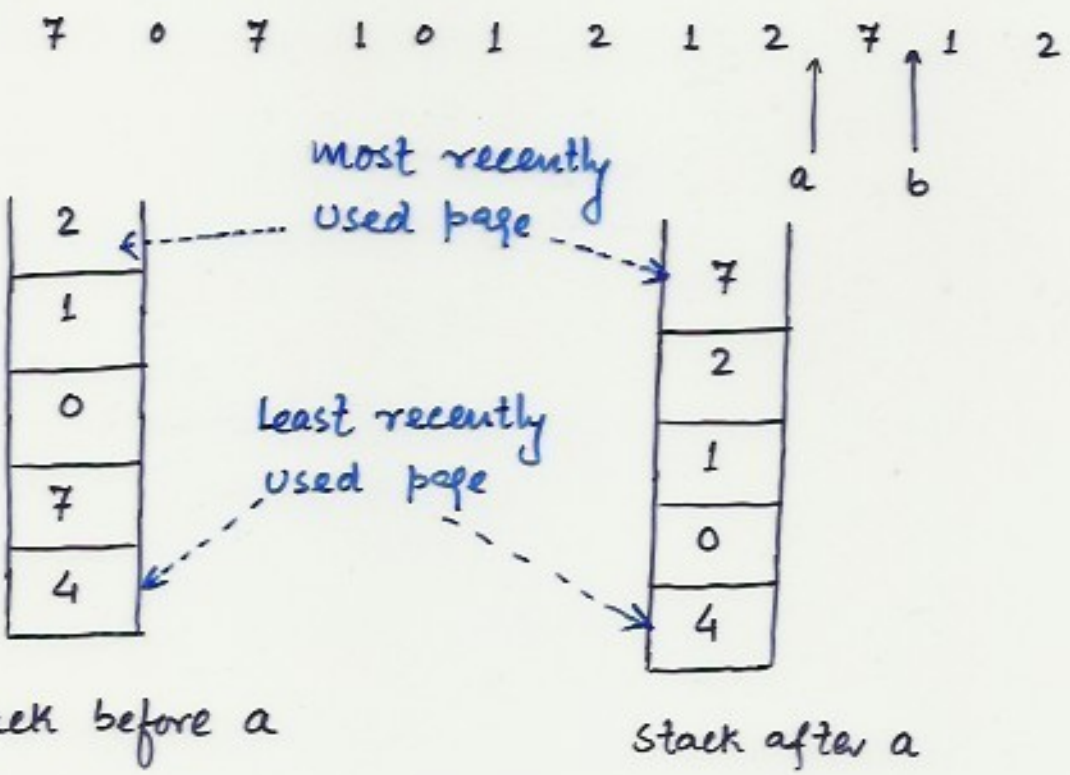
- Page table entry which has smallest value for time of use field is replaced.

→ Using stack :

- Keeps a stack of page nos.
- When a page is referenced, that page is moved at the top of stack.
- top of stack represents most recently used page.
- bottom of stack represents least recently used page.

reference string

4 7 0 7 1 0 1 2 1 2 7 1 2



Stack before a

Stack after a

LRU Approximation Algorithm

→ Some systems provide no hardware support for LRU.

→ Many system use reference bit as only support for LRU.

→ Reference bit :

- is set to 1 whenever the page is referenced.
- is associated with each page table entry.

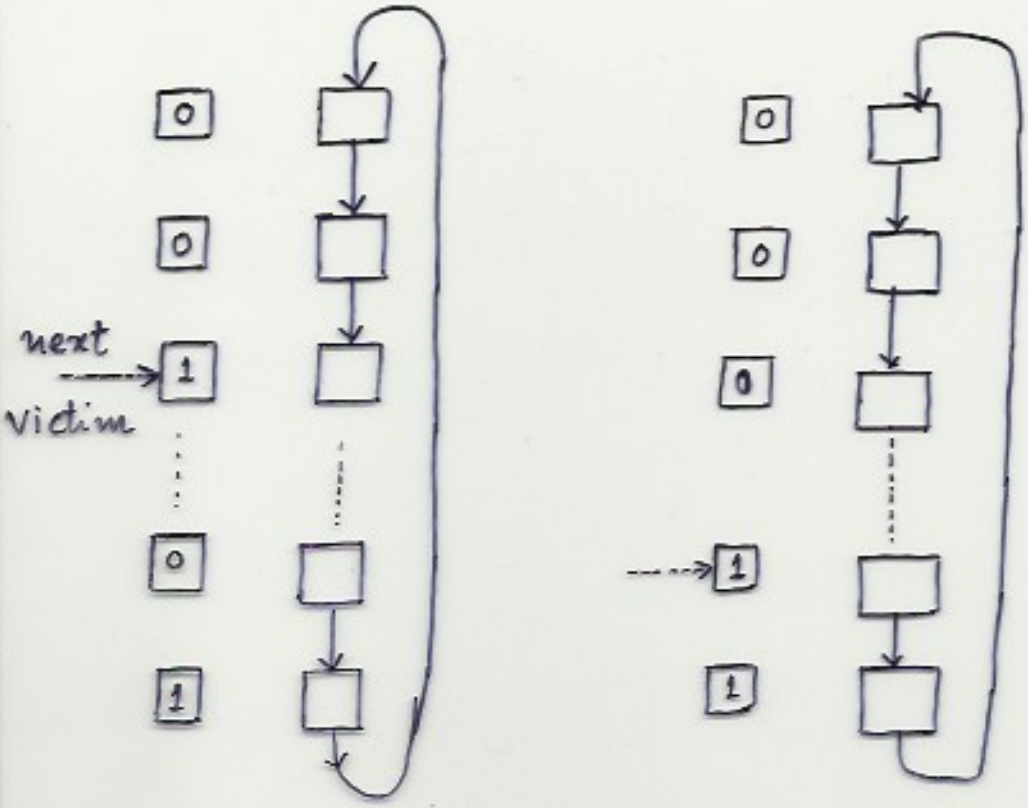
Second chance Algorithm :

→ Replace a page if its reference bit is set to 0.

→ If reference bit is 1, we give that page a second chance & move on to select next FIFO page.

→ When given second chance, reference bit is set to 0 & its arrival time is reset to the current time.

→ One way to implement second-chance algorithm is circular queue.



Circular queue of pages

Counting Algorithms:

→ Keep a counter of the number of references that have been made to each page.

→ Two algorithms are common:

- Least Frequently used (LFU) Algorithm.
- Most Frequently used (MFU) Algorithm.

LFU Algorithm :

- Page with smallest count is replaced.
- Suffers from situation when a page is heavily during initial phase but then never used.
- One solution is to shift the counts right by 1 bit at regular intervals.

MFU Algorithm :

- Page with largest count is replaced.
- Based on the argument that the page with the smallest count was probably just brought in & has yet to be used.

Allocation of frames

→ Each process needs minimum no. of frames for its execution.

→ Two major allocation schemes:

- fixed allocation
- Priority Allocation.

Fixed Allocation

→ Easiest way to split m frames among n processes is to give everyone equal share m/n frames.

→ Scheme is called Equal Allocation.

→ drawback is - smaller processes may be allocated more frames.

→ Proportional Allocation :

- Above problem is solved out.
- Allocate frames according to the size of the processes.
- s_i : size of process p_i

  S : $\sum_i s_i$

m : total no. of available frames.

frames allocated to process p_i :

$$a_i = s_i / S * m$$

a_i : an integer, greater than the minimum no. of frames required.

→ Drawback of fixed allocation -

- A higher priority process is treated the same as a lower priority process.

Priority Allocation

→ Uses a proportional allocation scheme, using priorities rather than size.

→ A combination of priority & size can also be used.

Global Vs Local Allocation

→ With multiple processes competing for frames, frame allocation can be either

- Global allocation.

or

- Local allocation.

Global Allocation

→ Allows a process to select a replacement frame from the set of all frames.

→ One process can take a frame from another.

→ A process can not control its own page fault rate.

→ Set of pages in memory for a process not only depends on that process but also on other processes.

Local Allocation

→ Each process select frame from its own set of allocated frames.

→ No. of frames allocated to a process does not change.