PAGE REPLACEMENT ALGORITHMS

Page Replacement Algorithms:

When a page fault occurs, the operating system has to choose a page to remove from memory to make room for the page that has to be brought in. The page replacement is done by swapping the required pages from backup storage to main memory and vice-versa. If the page to be removed has been modified while in memory, it must be rewritten to the disk to bring the disk copy up to date. If, however, the page has not been changed (e.g., it contains program text), the disk copy is already up to date, so no rewrite is needed. The page to be read in just overwrites the page being evicted. *A page replacement algorithm is evaluated by running the particular algorithm on a string of memory references and compute the page faults.Referenced string is a sequence of pages being referenced. Page fault is not an error.* Contrary to what their name might suggest, page faults are not errors and are common and necessary to increase the amount of memory available to programs in any operating system that utilizes virtual memory, including Microsoft Windows, Mac OS X, Linux and Unix.

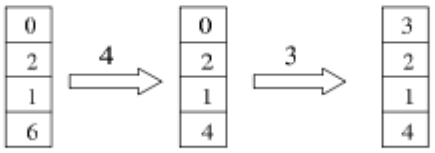
Each operating system uses different page replacement algorithms. To select the particular algorithm, the algorithm with lowest page fault rate is considered.
1.Optimal page replacement algorithm
2.Not recently used page replacement
3.First-In, First-Out page replacement
4.Second chance page replacement
5.Clock page replacement
6.Least recently used page replacement

The Optimal Page Replacement Algorithm:

The algorithm has lowest page fault rate of all algorithm. This algorithm state that: Replace the page which will not be used for longest period of time i.e future knowledge of reference string is required.
•Often called Balady's Min Basic idea: Replace the page that will not be referenced for the longest time.
•Impossible to implement

Optimal Page Replacement

•Consider the following reference string: 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1



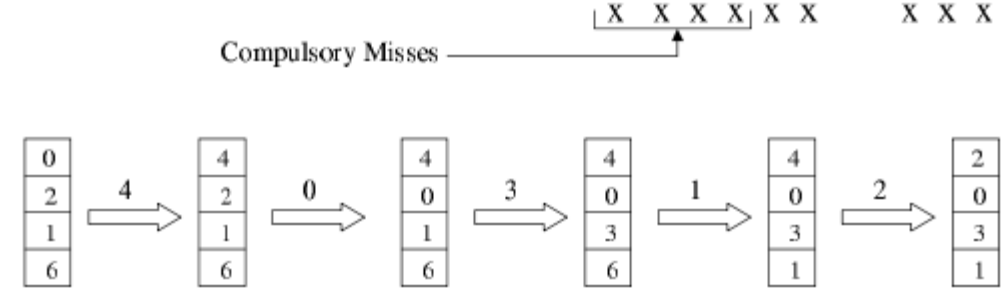
•Fault Rate = 6 / 12 = 0.50
•With the above reference string, this is the best we can hope to do

FIFO: (First In First Out)

•The oldest page in the physical memory is the one selected for replacement.
•Very simple to implement.
- Keep a list
On a page fault, the page at the head is removed and the new page added to the tail of the list

FIFO

•Consider the following reference string: 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1



•Fault Rate = 9 / 12 = 0.75

Issues:
•poor replacement policy
•FIFO doesn't consider the page usage.

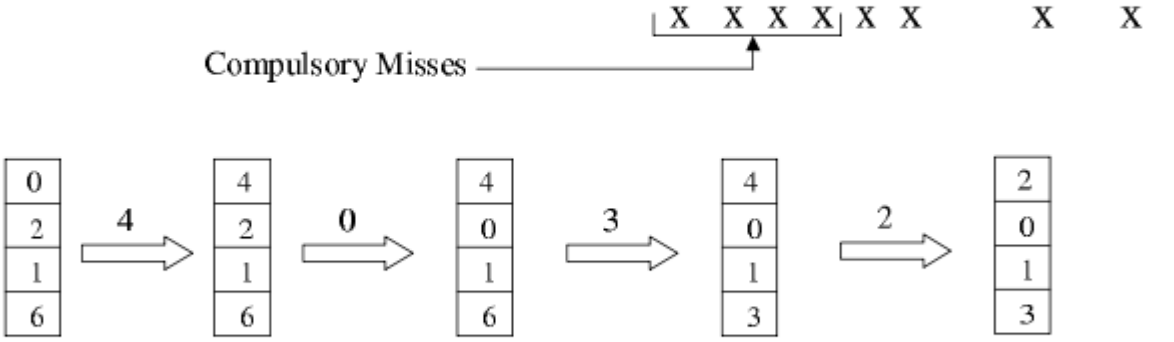
LRU(Least Recently Used):

In this algorithm, the page that has not been used for longest period of time is selected for replacement.

Although LRU is theoretically realizable, it is not cheap. To fully implement LRU, it is necessary to maintain a linked list of all pages in memory, with the most recently used page at the front and the least recently used page at the rear. The difficulty is that the list must be updated on every memory reference. Finding a page in the list, deleting it, and then moving it to the front is a very time-consuming operation, even in hardware (assuming that such hardware could be built).

LRU

•Consider the following reference string: 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1



•Fault Rate = 8 / 12 = 0.67

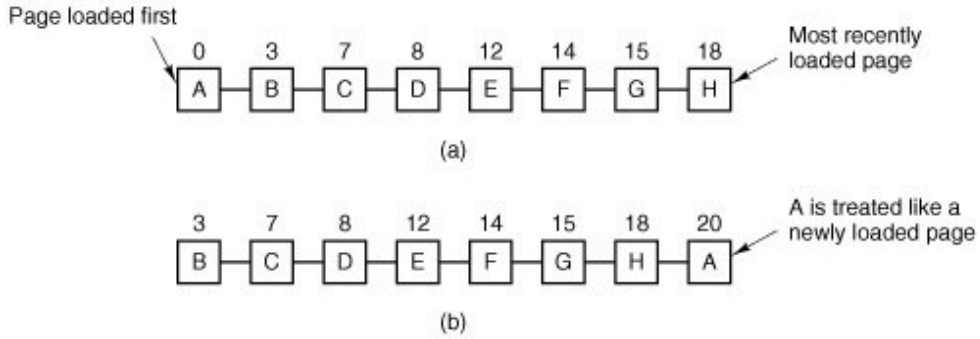
The Not Recently Used Page Replacement Algorithm

Two status bit associated with each page. R is set whenever the page is referenced (read or written). M is set when the page is written to (i.e., modified).

When a page fault occurs, the operating system inspects all the pages and divides them into four categories based on the current values of their R and M bits:
Class 0: not referenced, not modified.
Class 1: not referenced, modified.
Class 2: referenced, not modified.
Class 3: referenced, modified.
The NRU (Not Recently Used) algorithm removes a page at random from the lowest numbered nonempty class.

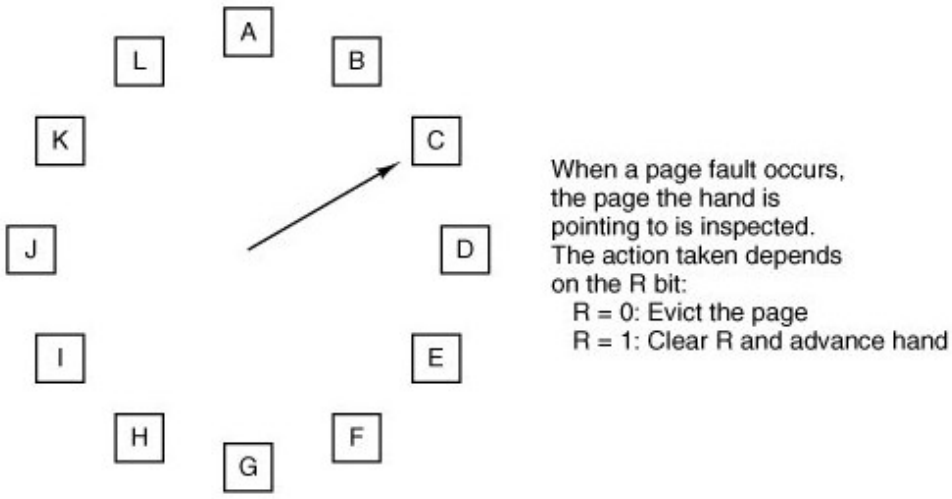
The Second Chance Page Replacement Algorithm:

A simple modification to FIFO that avoids the problem of heavily used page. It inspects the R bit If it is 0, the page is both old and unused, so it is replaced immediately. If the R bit is 1, the bit is cleared, the page is put onto the end of the list of pages, and its load time is updated as though it had just arrived in memory. Then the search continues.



The Clock Page Replacement Algorithm

keep all the page frames on a circular list in the form of a clock, as shown in Fig. A hand points to the oldest page.



When a page fault occurs, the page being pointed to by the hand is inspected. If its R bit is 0, the page is evicted, the new page is inserted into the clock in its place, and the hand is advanced one position. If R is 1, it is cleared and the hand is advanced to the next page. This process is repeated until a page is found with R = 0

Source : http://dayaramb.files.wordpress.com/2012/02/operating-system-pu.pdf