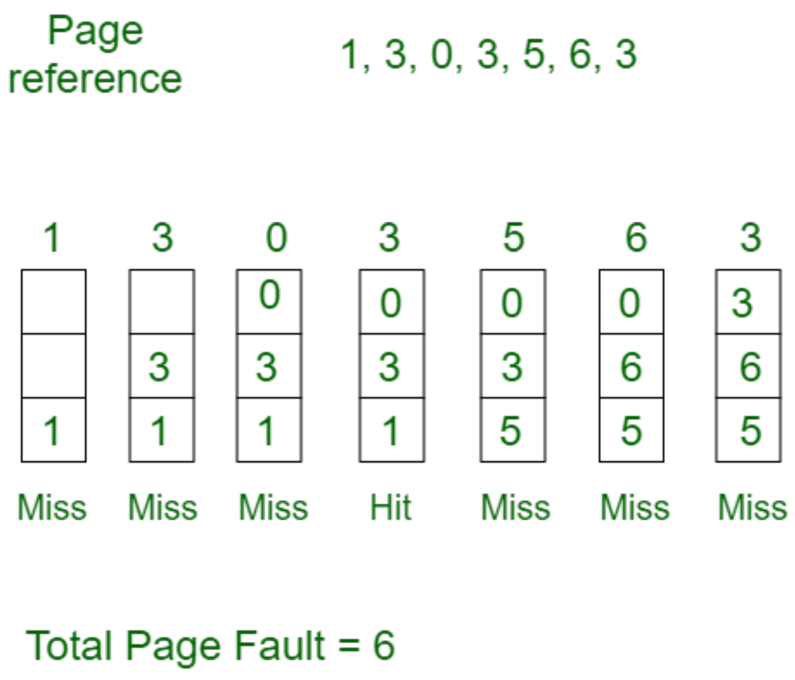
Page Replacement Algorithms in Operating Systems

First in First Out (FIFO):
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 page in the front of the queue is selected for removal.

Example Question: Consider the page reference string 1, 3, 0, 3, 5, 6 with 3-page frames. Find the number of page faults. Show your work.



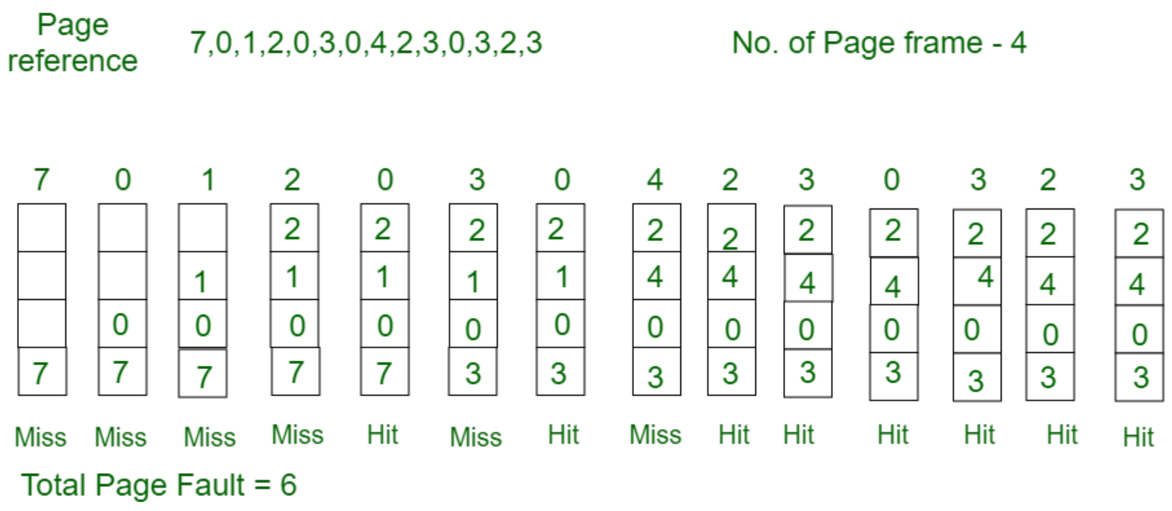
- Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots which creates **3 Page Faults**.
- When 3 comes, it is already in memory so **0 Page Faults**
- Then 5 comes, it is not available in memory, so it replaces the oldest page slot (#1) for **1 Page Fault**
- 6 comes, it is also not available in memory, so it replaces the oldest page slot (#3) for **1 Page Fault**
- Finally, when 3 come it is not available, so it replaces 0 for **1 Page Fault**

Belady's anomaly – Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm. For example, if we consider reference string 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4 and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10 Page Faults.

Optimal Page replacement:

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

Example Question: Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, with 4-page frames. Find the number of page faults. Show your work.



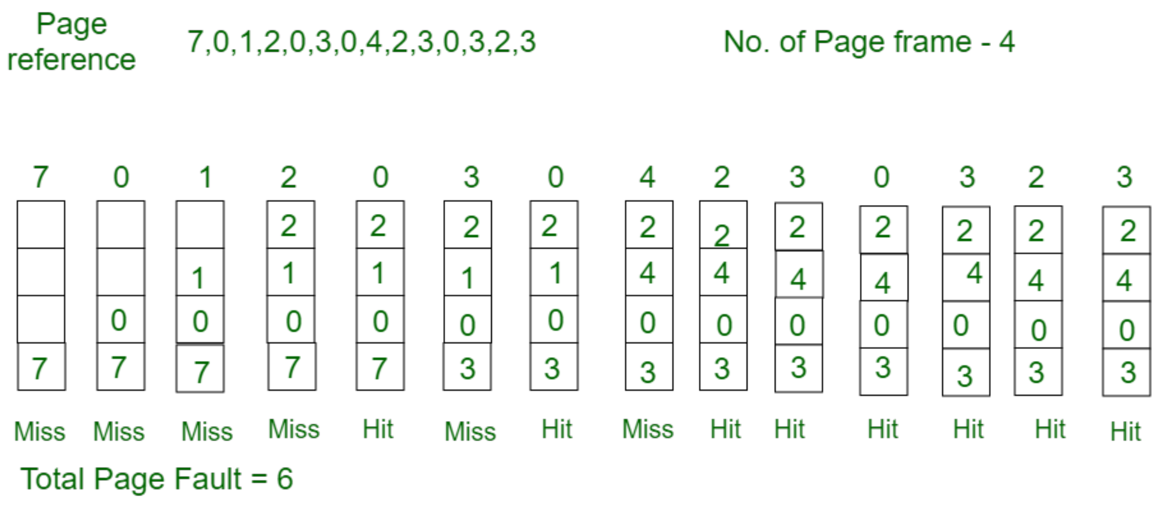
- Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots this creates **4 Page Faults**
- 0 is already there so, **0 Page Fault.**
- When 3 comes in, it will take the place of 7 because it is not used for the longest duration of time in the future which creates **1 Page Fault.**
- 0 is already there so, **0 Page Fault.**
- 4 will take the place of 1 for **1 Page Fault.**
- Now for the further page reference string, **0 Page fault** because they are already available in the memory.

Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

Least Recently Used:

In this algorithm the page will be replaced which is least recently used.

Example Question: Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4-page frames. Find the number of page faults. Show your work.



Here LRU has same number of page fault as optimal but it may differ according to question.

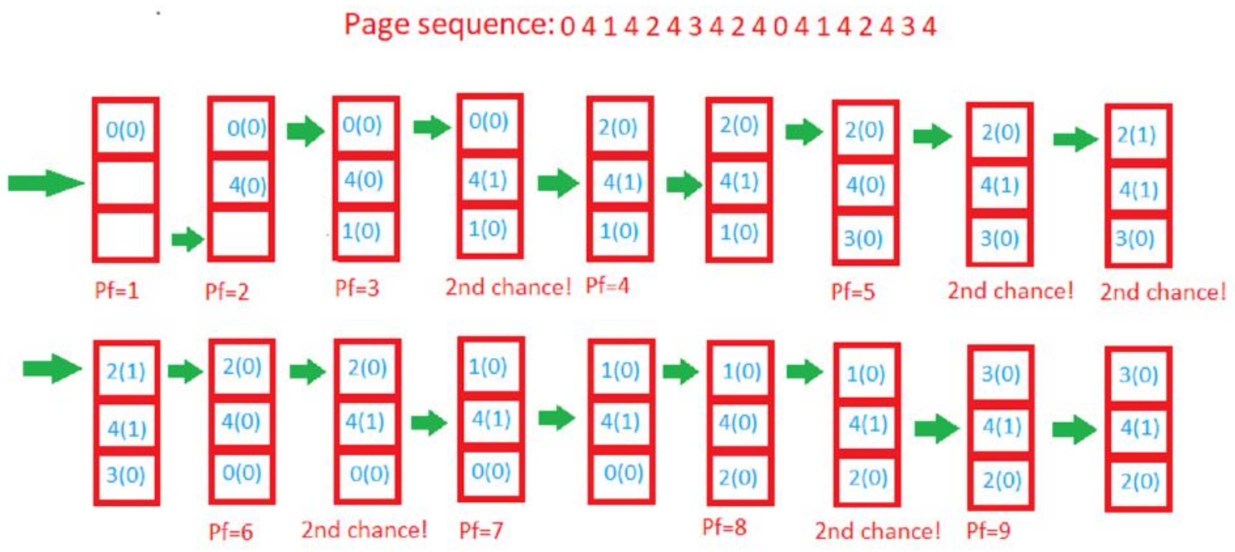
- Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots this creates **4 Page Faults**
- 0 is already present so, **0 Page Fault.**
- When 3 comes in, it will take the place of 7 because it is least recently used, for **1 Page Fault**
- 0 is already in memory so, **0 Page Fault**.
- 4 will takes place of 1 for **1 Page Fault**
- Now for the further page reference string, **0 Page fault** because they are already available in the memory.

Second Chance:

In the Second Chance page replacement policy, the candidate pages for removal are considered in a round robin 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 way, has not been accessed since its last consideration.

It can be implemented by adding a “second chance” bit to each memory frame-every time the frame is considered (due to a reference made to the page inside it), this bit is set to 1, which gives the page a second chance, as when we consider the candidate page for replacement, we replace the first one with this bit set to 0 (while zeroing out bits of the other pages we see in the process). Thus, a page with the “second chance” bit set to 1 is never replaced during the first consideration and will only be replaced if all the other pages deserve a second chance too.

Example: Consider the page reference string **0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4** with 3-page frames. Find the number of page faults. Show your work.



- Initially, all frames are empty so after first **3 passes** they will be filled with {0, 4, 1} and the second chance array will be {0, 0, 0} as none have been referenced yet. Also, the pointer will cycle back to 0.
- **Pass-4:** Frame={0, 4, 1}, second_chance = {0, 1, 0} [4 will get a second chance], pointer = 0 (No page needed to be updated so the candidate is still page in frame 0), pf = 3 (No increase in page fault number).
- **Pass-5:** Frame={2, 4, 1}, second_chance= {0, 1, 0} [0 replaced; it's second chance bit was 0, so it didn't get a second chance], pointer=1 (updated), pf=4
- **Pass-6:** Frame={2, 4, 1}, second_chance={0, 1, 0}, pointer=1, pf=4 (No change)
- **Pass-7:** Frame={2, 4, 3}, second_chance= {0, 0, 0} [4 survived but it's second chance bit became 0], pointer=0 (as element at index 2 was finally replaced), pf=5
- **Pass-8:** Frame={2, 4, 3}, second_chance= {0, 1, 0} [4 referenced again], pointer=0, pf=5
- **Pass-9:** Frame={2, 4, 3}, second_chance= {1, 1, 0} [2 referenced again], pointer=0, pf=5
- **Pass-10:** Frame={2, 4, 3}, second_chance= {1, 1, 0}, pointer=0, pf=5 (no change)
- **Pass-11:** Frame={2, 4, 0}, second_chance= {0, 0, 0}, pointer=0, pf=6 (2 and 4 got second chances)
- **Pass-12:** Frame={2, 4, 0}, second_chance= {0, 1, 0}, pointer=0, pf=6 (4 will again get a second chance)
- **Pass-13:** Frame={1, 4, 0}, second_chance= {0, 1, 0}, pointer=1, pf=7 (pointer updated, pf updated)
- **Page-14:** Frame={1, 4, 0}, second_chance= {0, 1, 0}, pointer=1, pf=7 (No change)
- **Page-15:** Frame={1, 4, 2}, second_chance= {0, 0, 0}, pointer=0, pf=8 (4 survived again due to 2nd chance!)
- **Page-16:** Frame={1, 4, 2}, second_chance= {0, 1, 0}, pointer=0, pf=8 (2nd chance updated)

- **Page-17:** Frame={3, 4, 2}, second_chance= {0, 1, 0}, pointer=1, pf=9 (pointer, pf updated)
- **Page-18:** Frame={3, 4, 2}, second_chance= {0, 1, 0}, pointer=1, pf=9 (No change)

In this example, the second chance algorithm does as well as the LRU method, which is much more expensive to implement in hardware.