CS 452 Operating Systems

Paging

Dr. Denton Bobeldyk

Page Replacement

- Basic Page Replacement
- FIFO Page Replacement
- Optimal Page Replacement
- LRU Page Replacement
- Page Buffering Algorithms

Page Replacement

- Basic Page Replacement
- FIFO Page Replacement
- Optimal Page Replacement
- LRU Page Replacement
- Page Buffering Algorithms

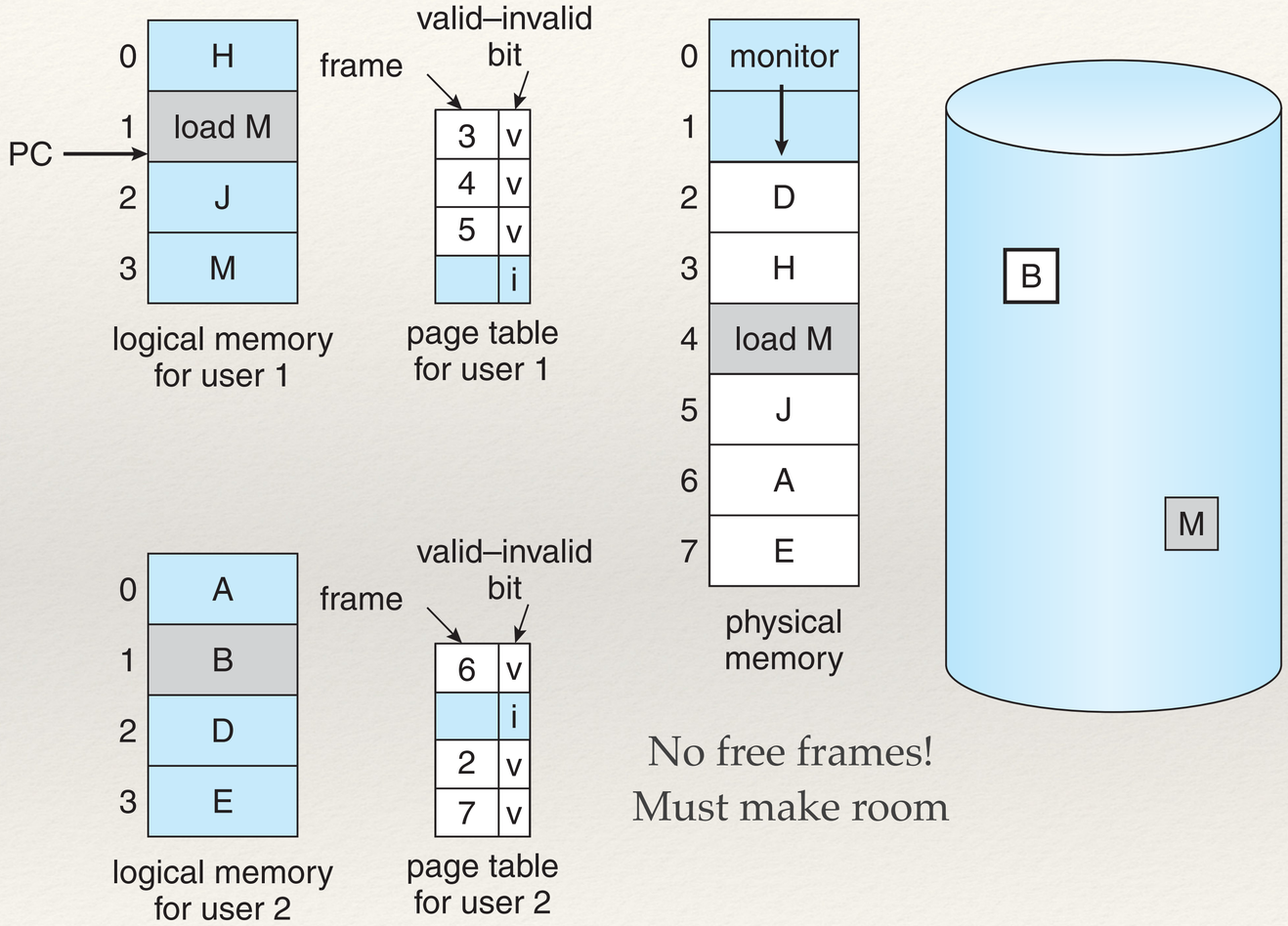
Page Replacement Motivation

- Only part of the program is loaded into memory
  - Saves I/O
  - Saves memory space
  - Can over-allocate available memory
- What to do when a program page faults and there are no available free frames?
  - Terminate another program to free up memory
  - Replace pages of other programs

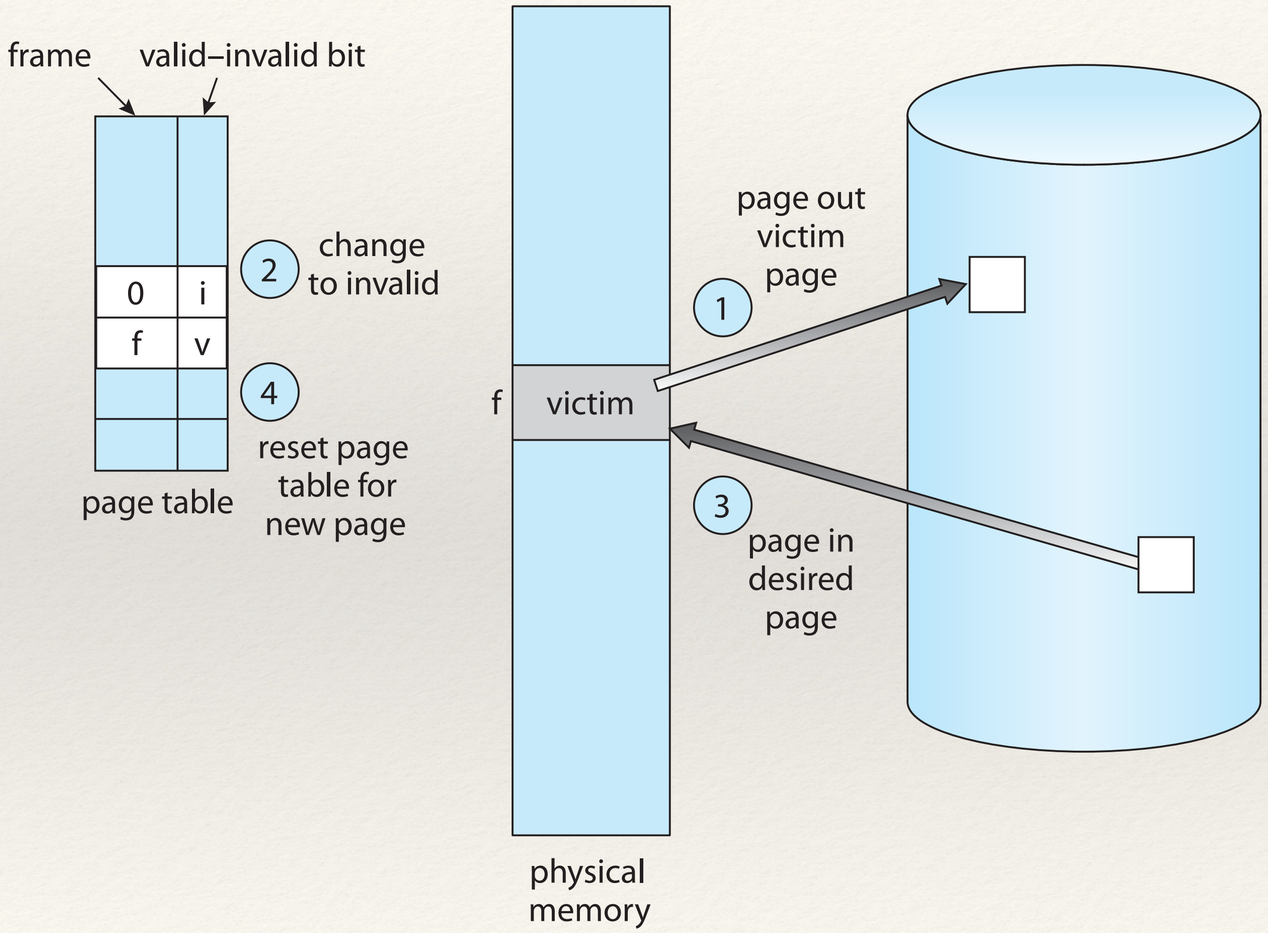
Basic Page Replacement

- If no frame is free, find one that isn't being used and replace it
- The frame being replaced is known as a **victim frame**
  - Write the victim frame to disk (Page out)
  - Load the desired page into memory (Page in)

Paging Overview



Page Replacement Algorithms



Paging Overhead

- Page out victim page
- Page in desired page

Paging Overhead

- If the victim page wasn't modified, does it need to be written back to the data store?
- Track if it was modified by using a 'modify or dirty bit'
  - If the bit isn't set, no need to write/save it to the data store

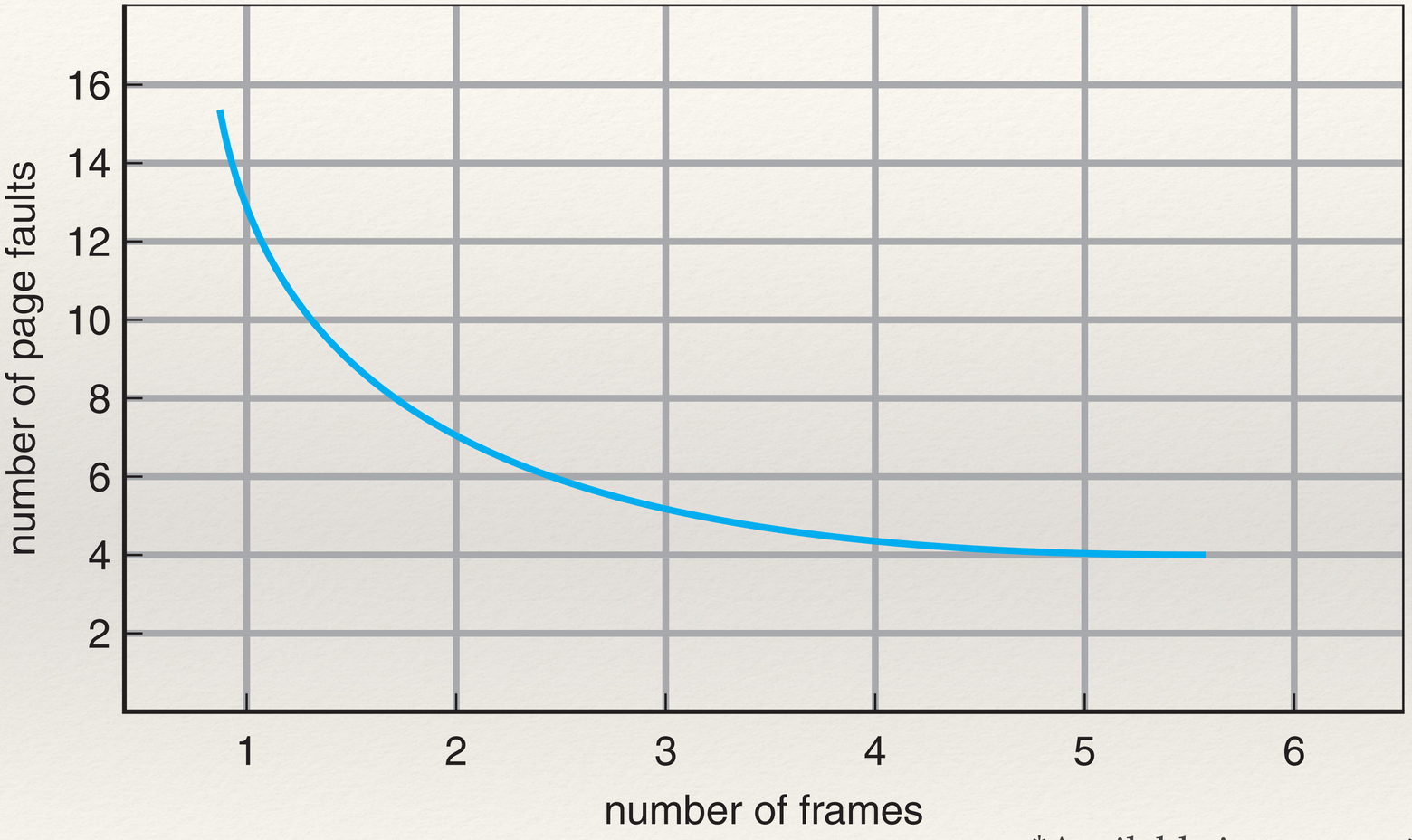
Demand Paging

- Requires
  - Frame allocation algorithm
    - How many frames does each process get in memory?
  - Page-replacement algorithm
    - Which frames/pages will be replaced.

Page Replacement Algorithms

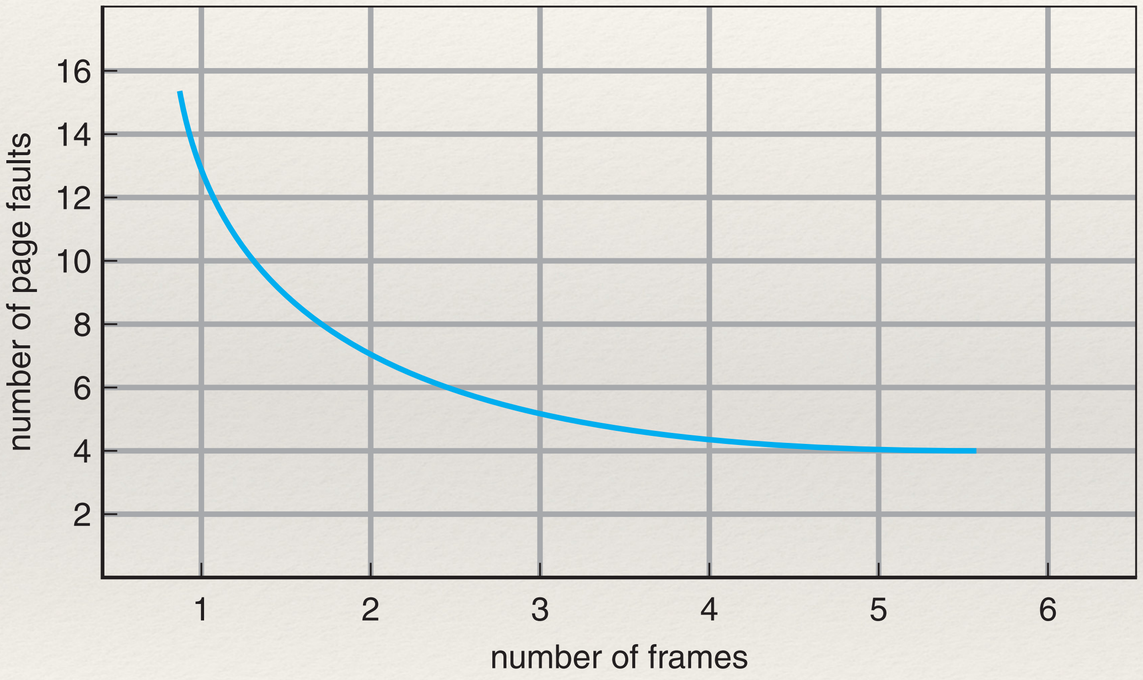
- What makes a good one?
  - Lowest page-fault rate
- Evaluate an algorithm by running it against a string of memory references
  - The string of memory references is called a **reference string**
  - Can be generated artificially

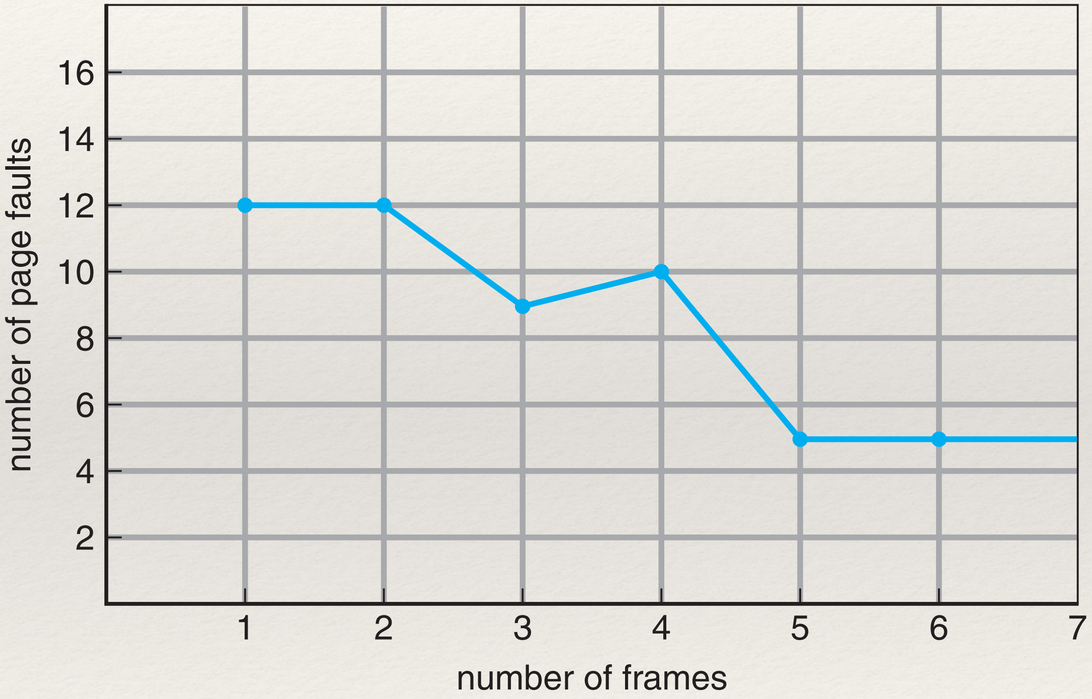
Page Replacement Algorithms



Available in memory

Belady's Anomaly





Paging Performance

- Belady's anomaly: for some page replacement algorithms, the page-fault rate may increase as the number of allocated frames increases.
  - Note: You would normally expect this to decrease

Page Replacement

- Basic Page Replacement
- FIFO Page Replacement
- Optimal Page Replacement
- LRU Page Replacement
- Page Buffering Algorithms

FIFO Page Replacement

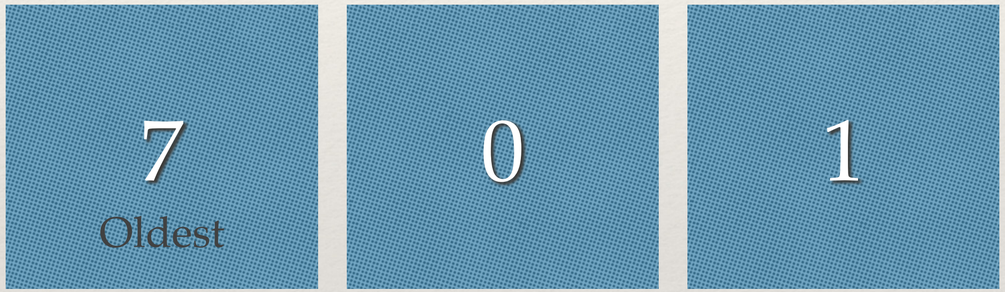
- Associate a time with each page, implement First-in, First-out algorithm for replacement
- Keep track of pages using a queue

FIFO Page Replacement

- Given the following reference string, let's illustrate the FIFO behavior
  - 7, 0, 1, 2, 0, 3, 0, 4, 2, 3
  - Three frames initially empty are filled with 7, 0, 1

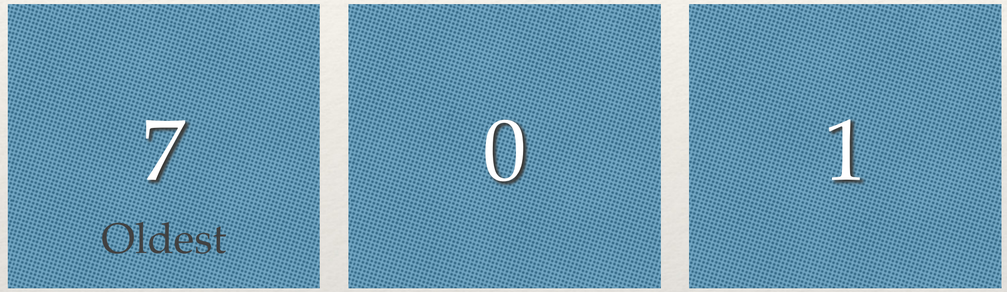
FIFO Page Replacement

- Reference String: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3



FIFO Page Replacement

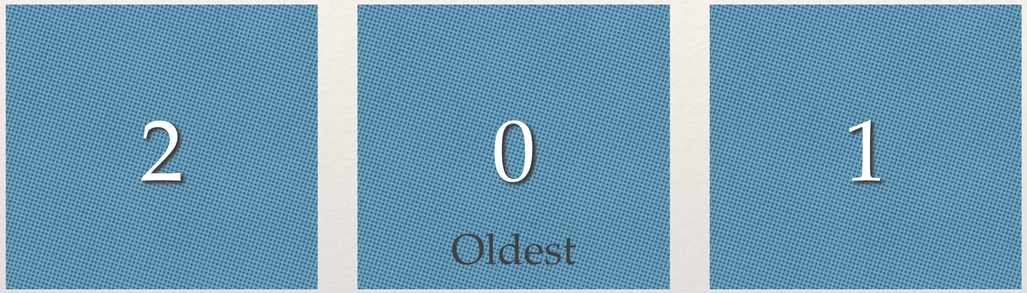
- Reference String: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3



2 replaces the 7

FIFO Page Replacement

- Reference String: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3

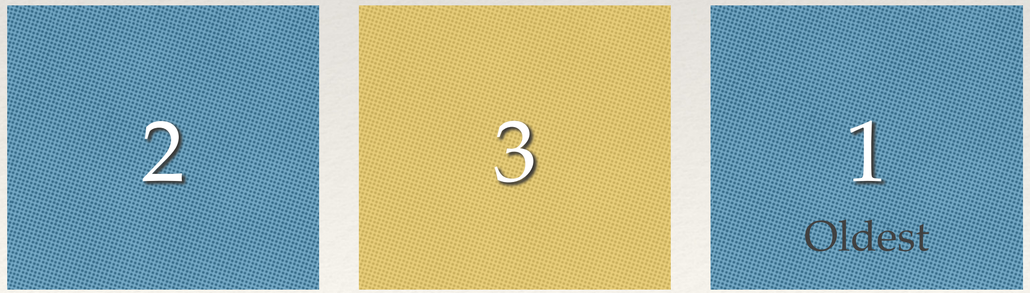


0 already in memory, no change

FIFO Page Replacement

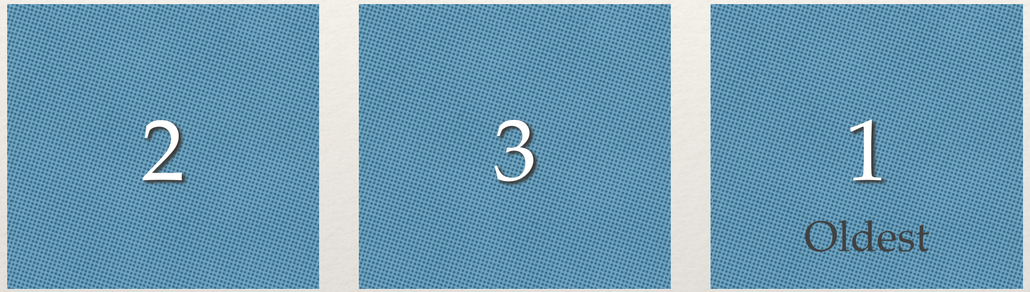
- Reference String: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3

3 replaces 0

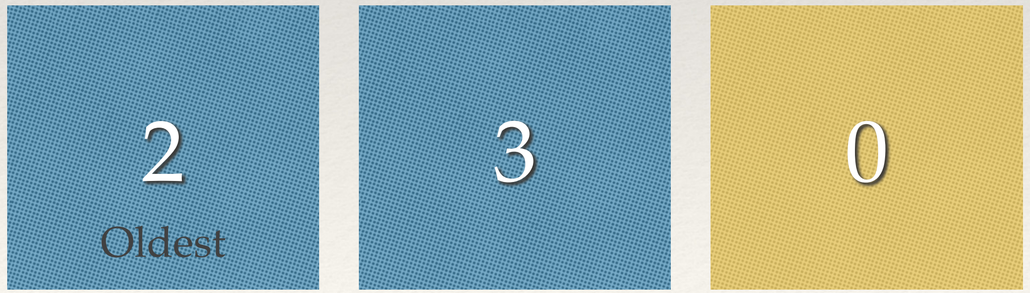


FIFO Page Replacement

- Reference String: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3

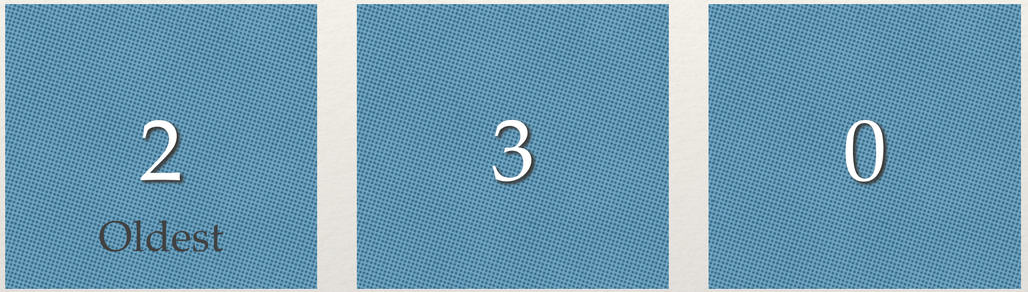


0 replaces 1



FIFO Page Replacement

- Reference String: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3

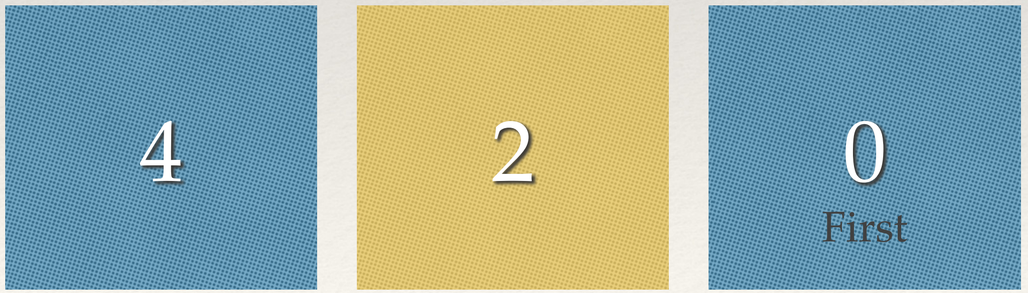


4 replaces 2

FIFO Page Replacement

- Reference String: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3

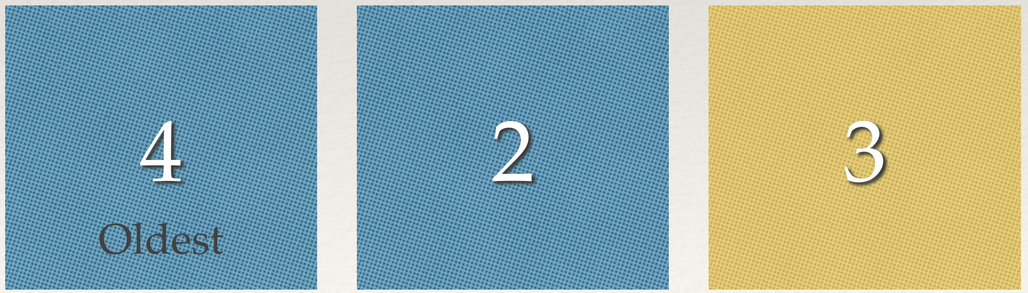
2 replaces 3



FIFO Page Replacement

- Reference String: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3

3 replaces 0



Reference Strings

- Different reference strings will return different results
- Certain strings may generate situations where a page that was just swapped out will need to be swapped back in again.

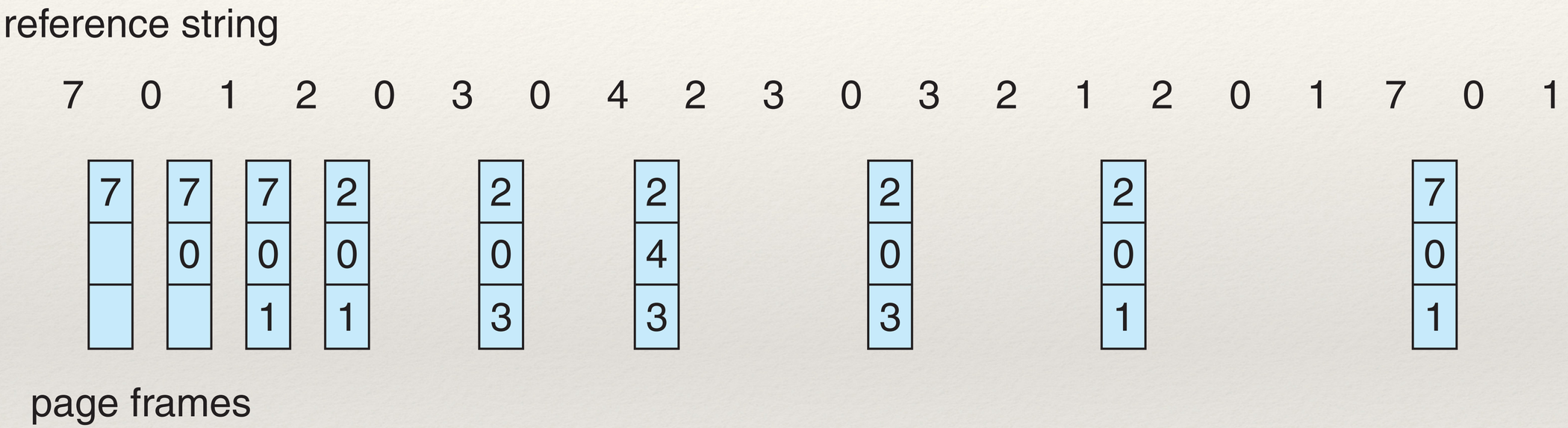
Page Replacement

- Basic Page Replacement
- FIFO Page Replacement
- Optimal Page Replacement
- LRU Page Replacement
- Page Buffering Algorithms

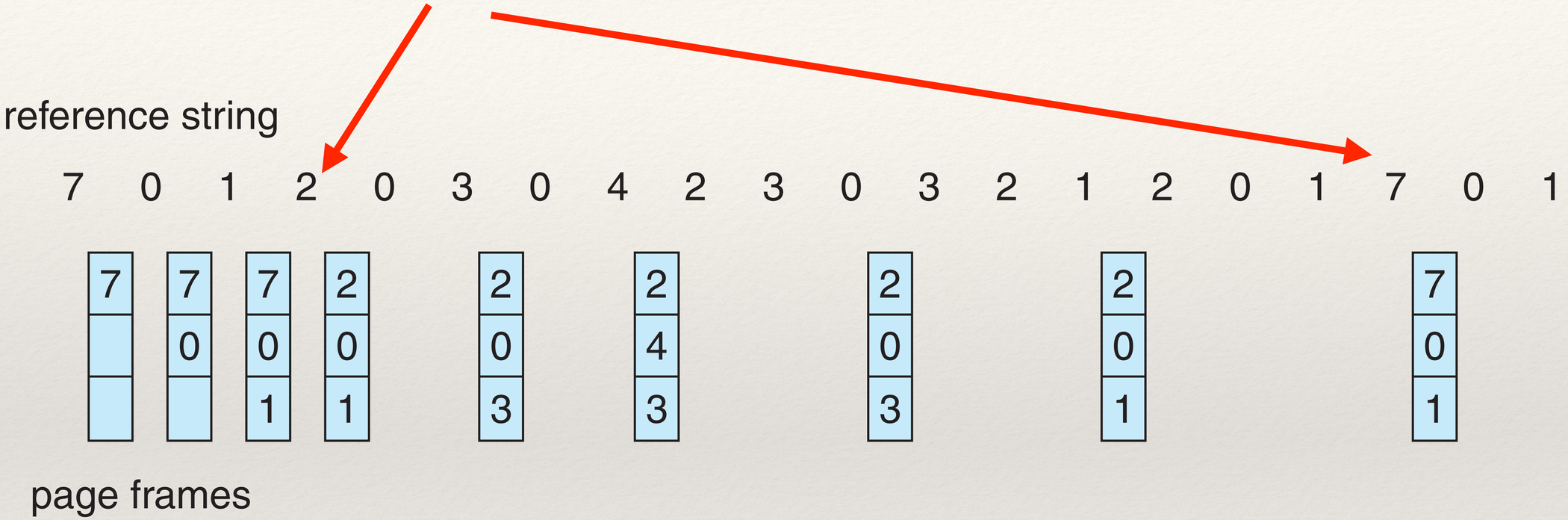
Optimal Page Replacement

- Guaranteed lowest possible page-fault rate
- Replace the page that will not be used for the longest period of time
- Difficult to actually implement as you need to know the future.
- Mainly used for comparison studies
- Similar situation to SJF CPU scheduling

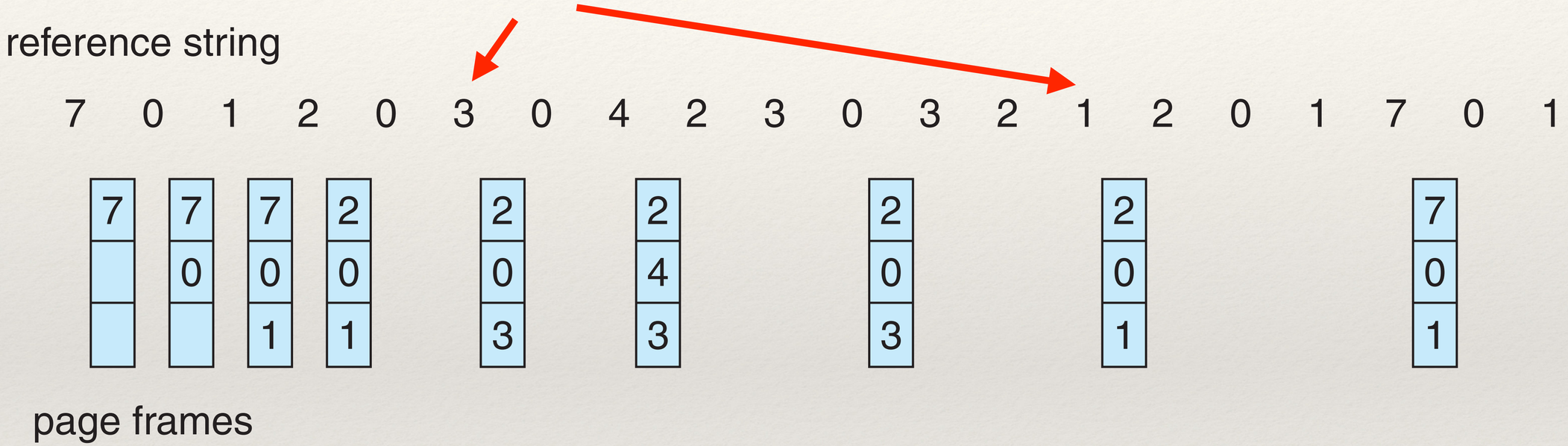
Optimal Page Replacement



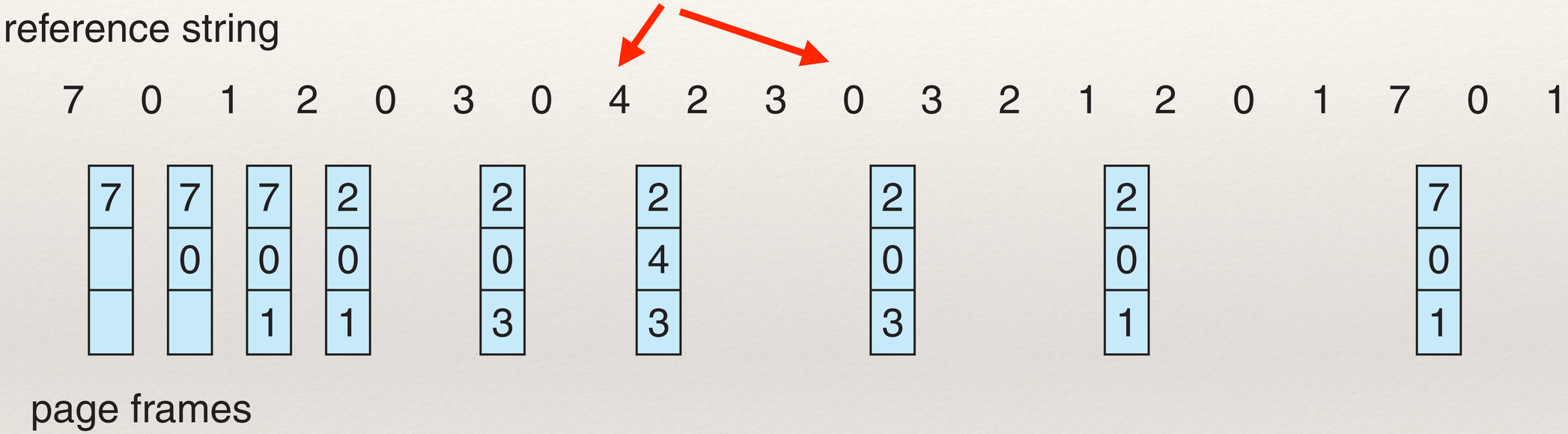
Optimal Page Replacement



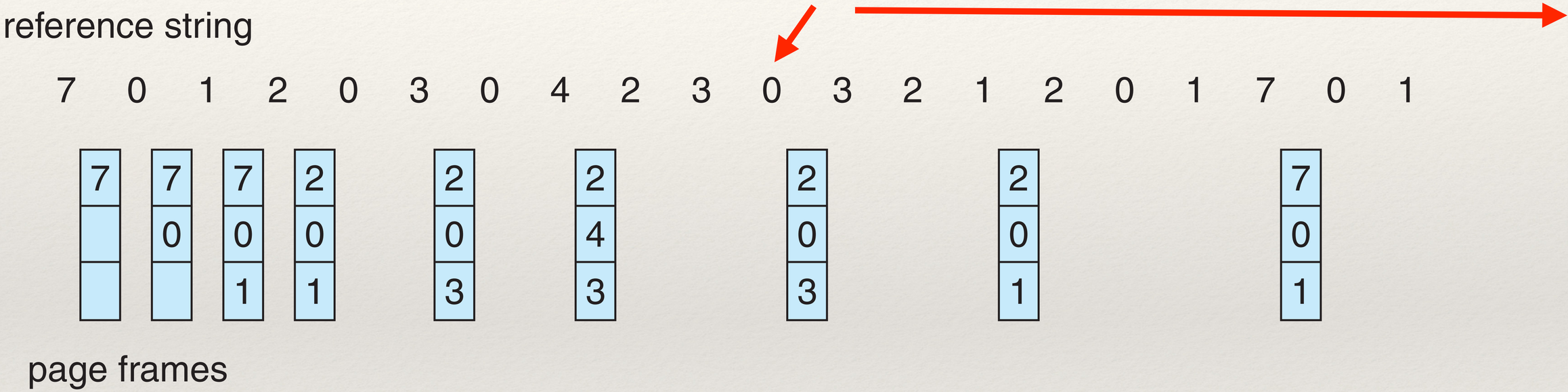
Optimal Page Replacement



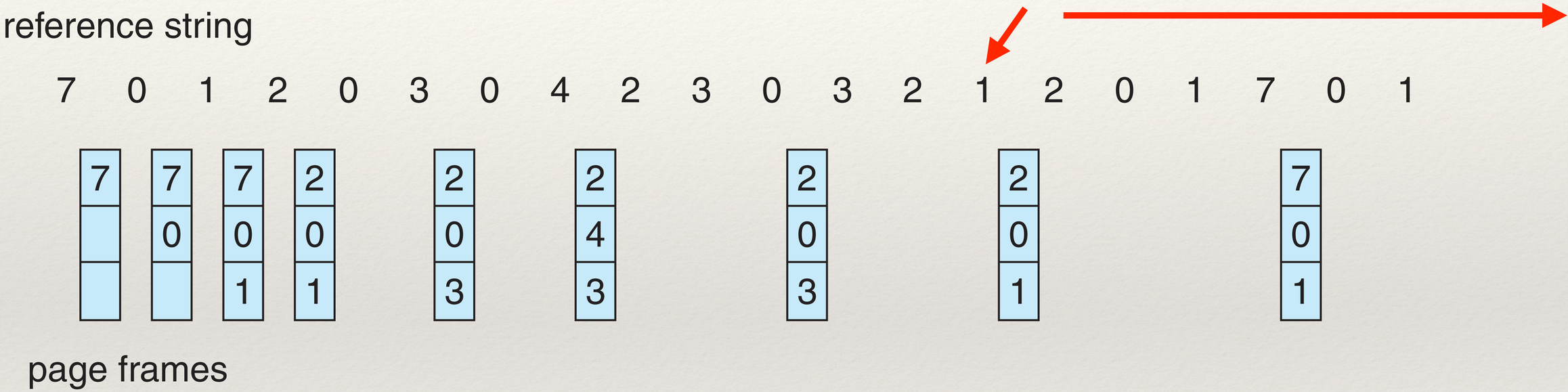
Optimal Page Replacement



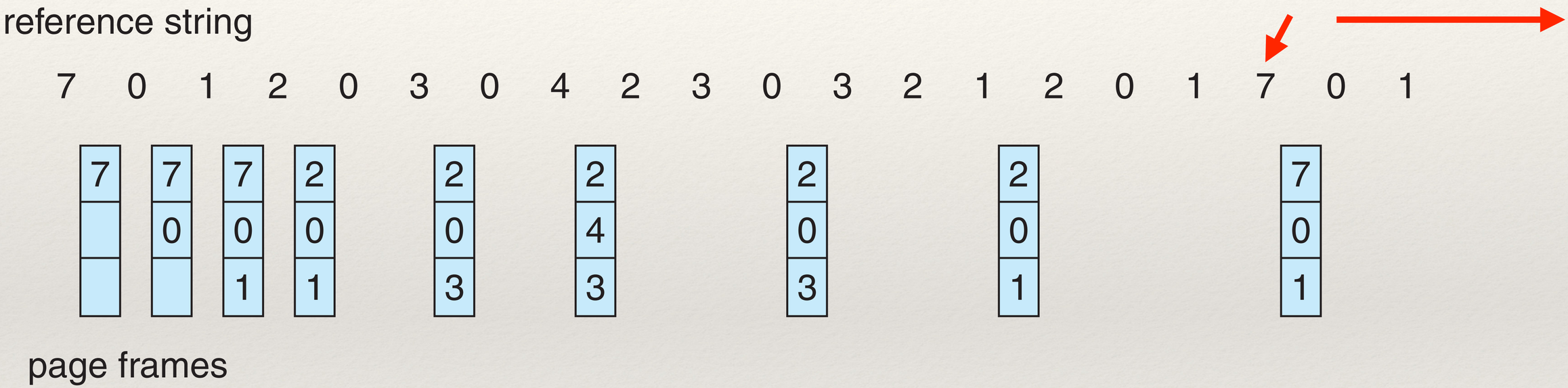
Optimal Page Replacement



Optimal Page Replacement



Optimal Page Replacement



Page Replacement

- Basic Page Replacement
- FIFO Page Replacement
- Optimal Page Replacement
- LRU Page Replacement
- Page Buffering Algorithms

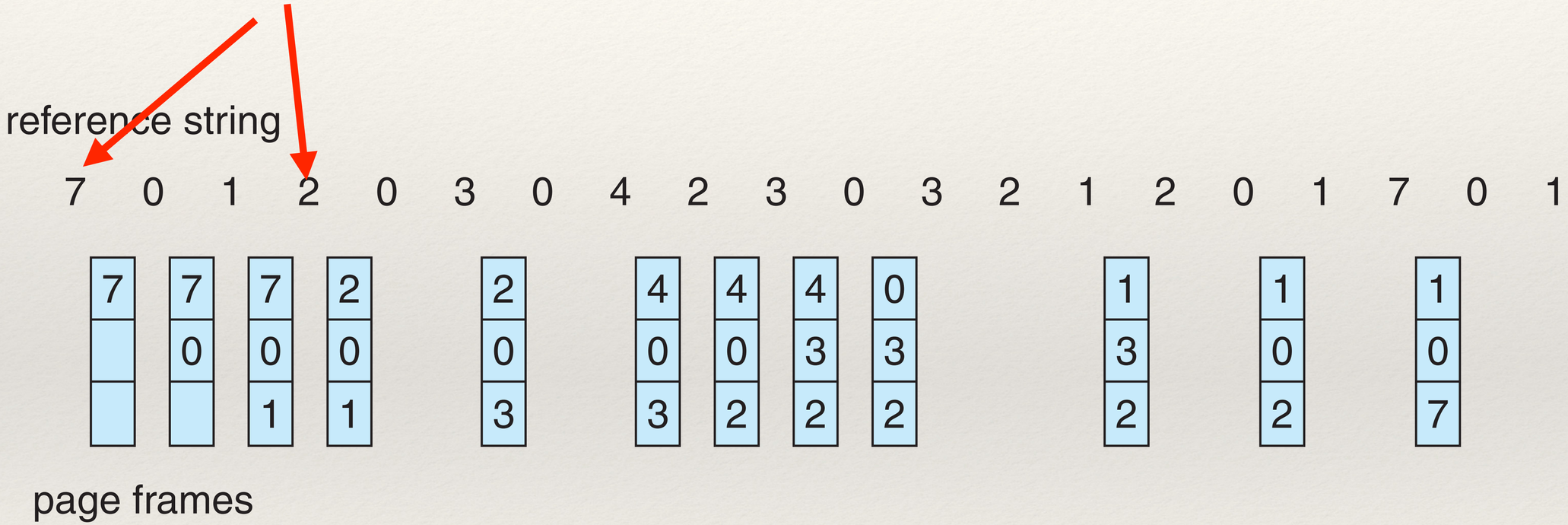
LRU Page Replacement

- Replace the page that has not been used for the longest
  - Looking backwards instead of forward (Optimum)
  - Don’t need to foresee the future, just the past

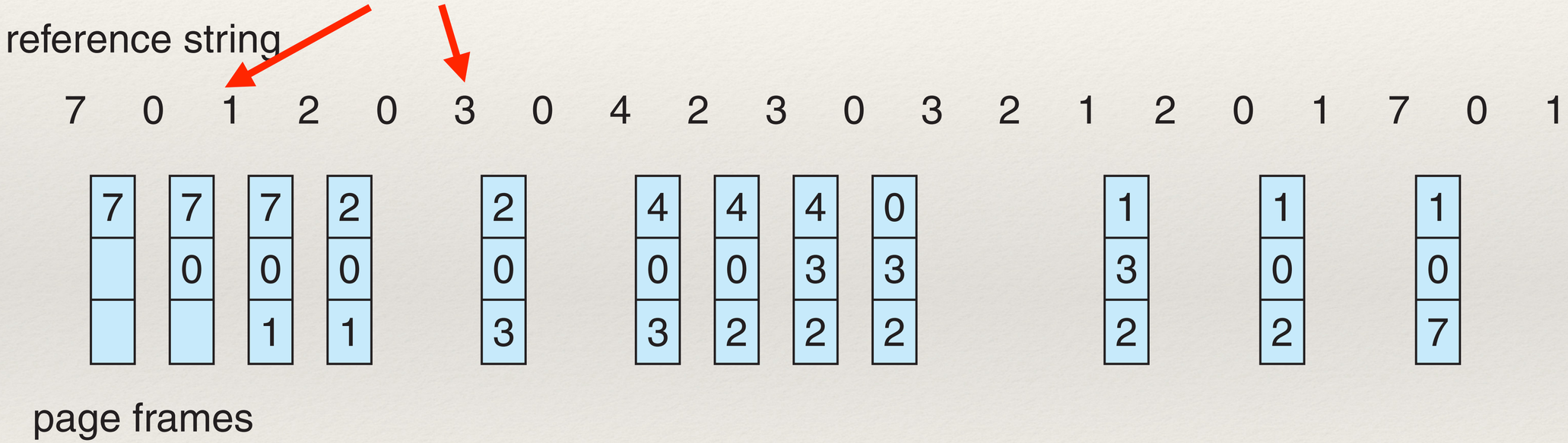
LRU Page Replacement

- Need to track when the page was last used
- Swap out the least recently used page

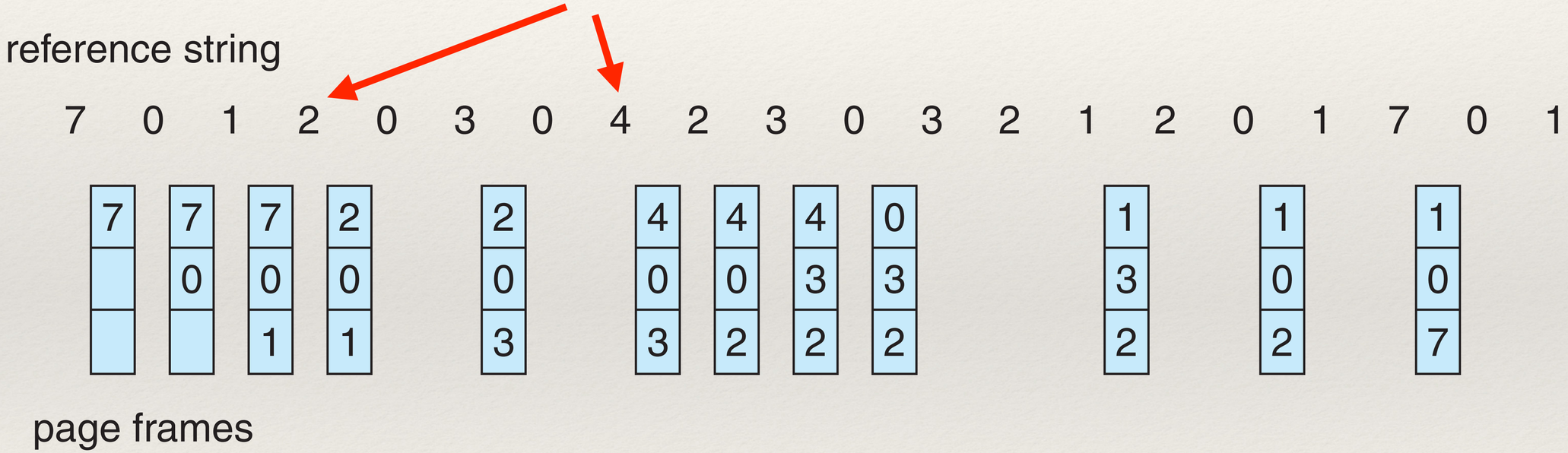
LRU Page Replacement



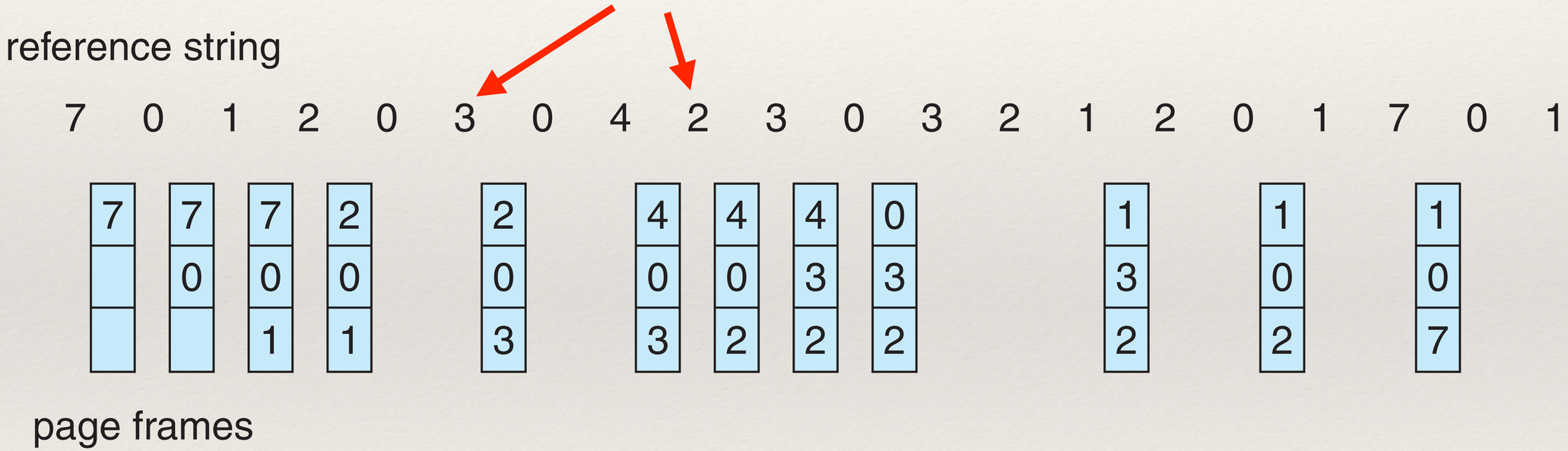
LRU Page Replacement



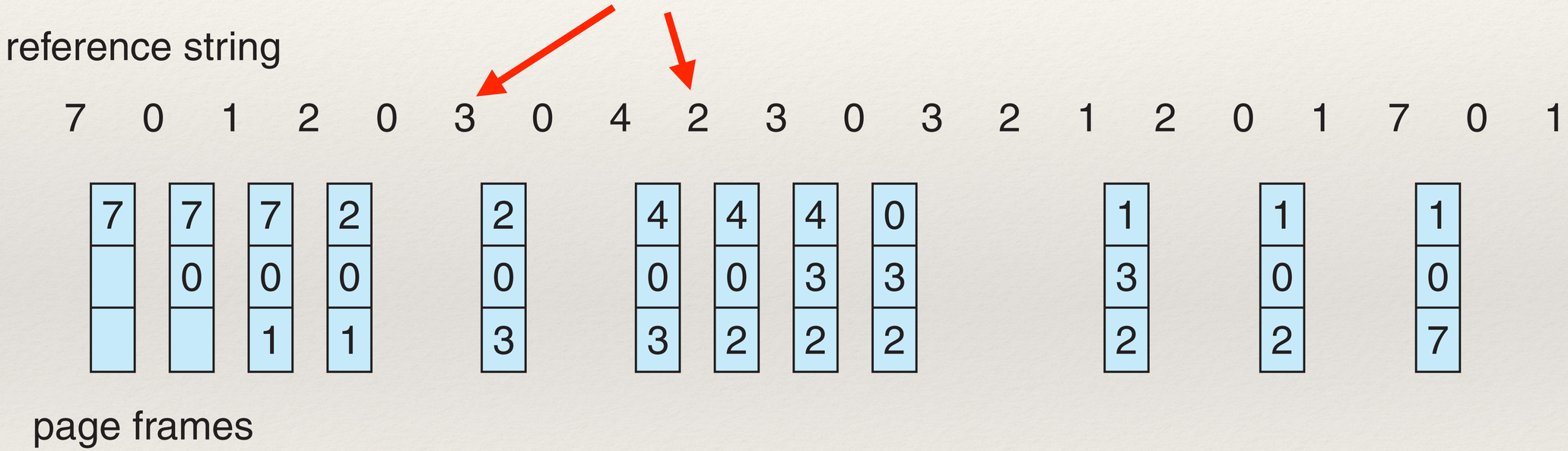
LRU Page Replacement



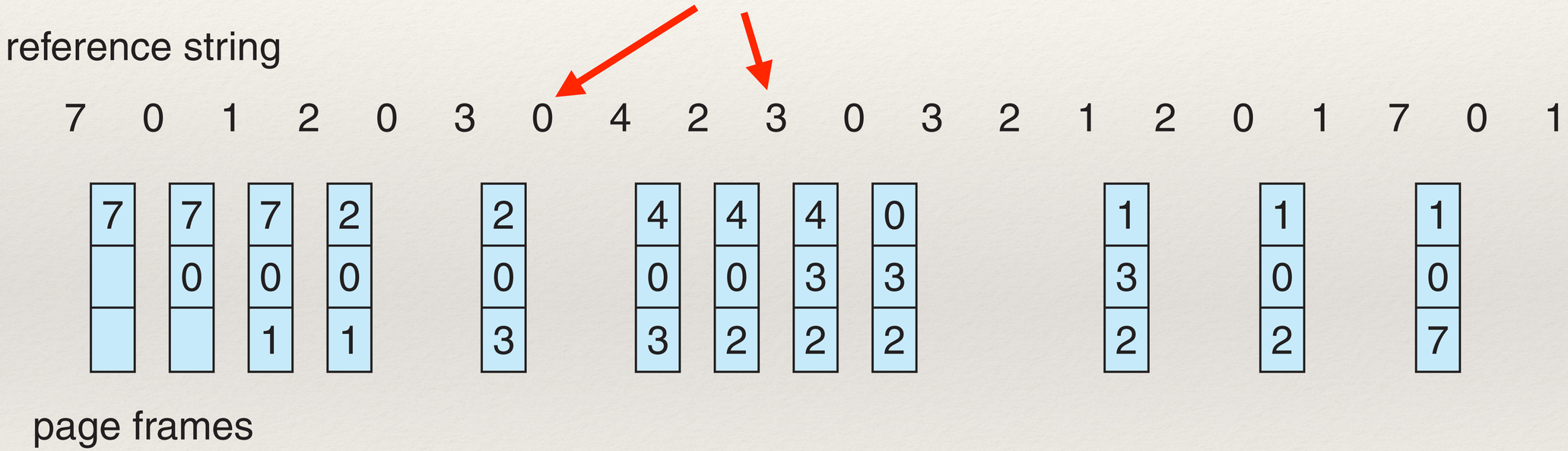
LRU Page Replacement



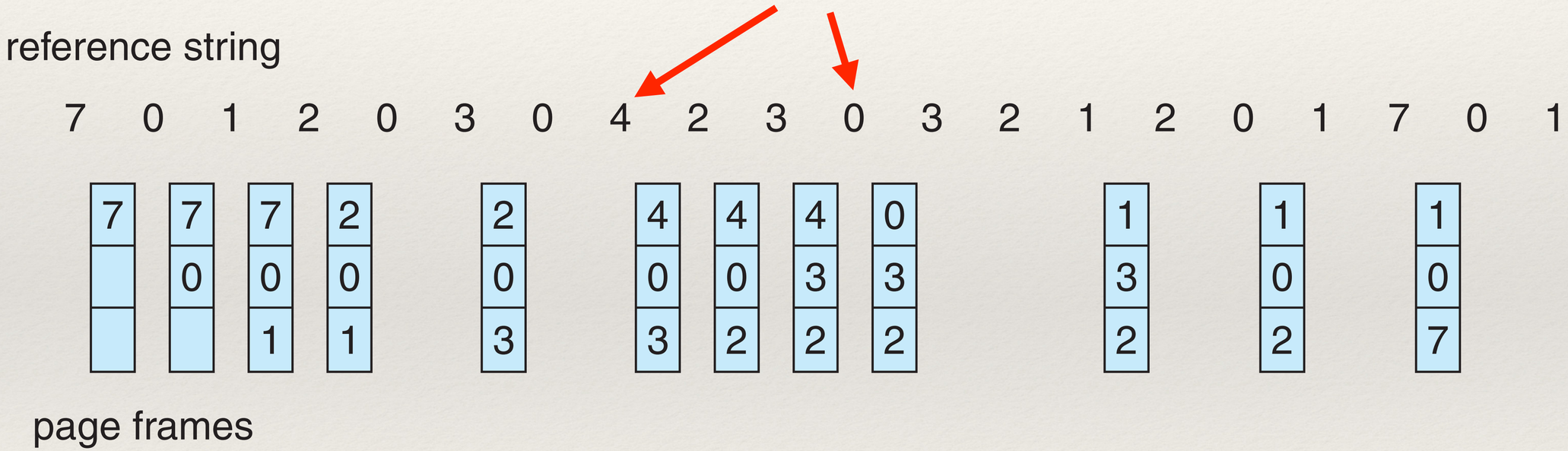
LRU Page Replacement



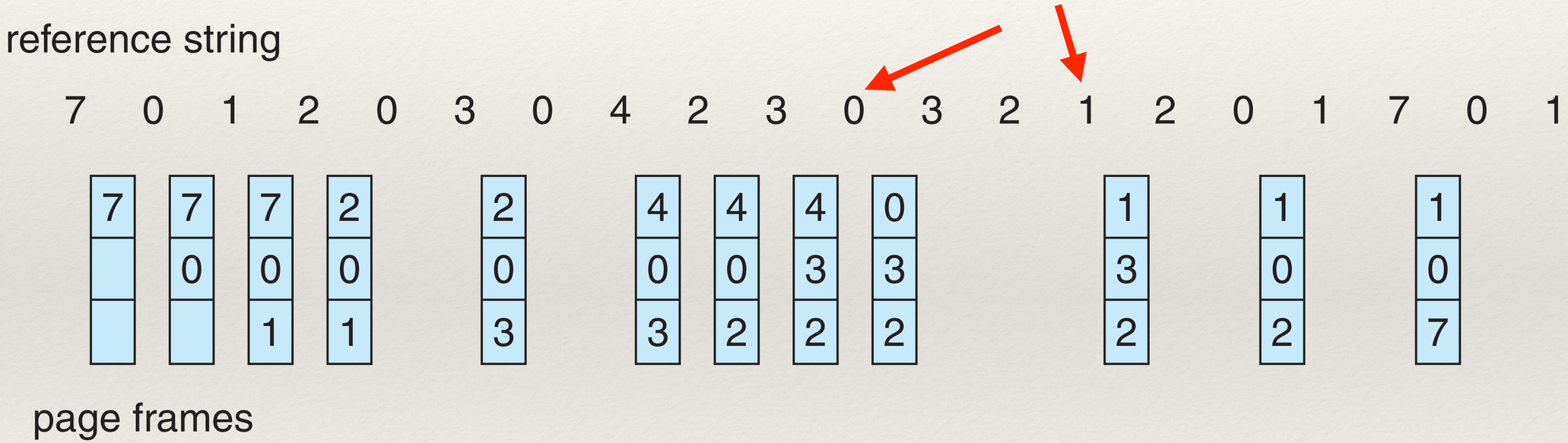
LRU Page Replacement



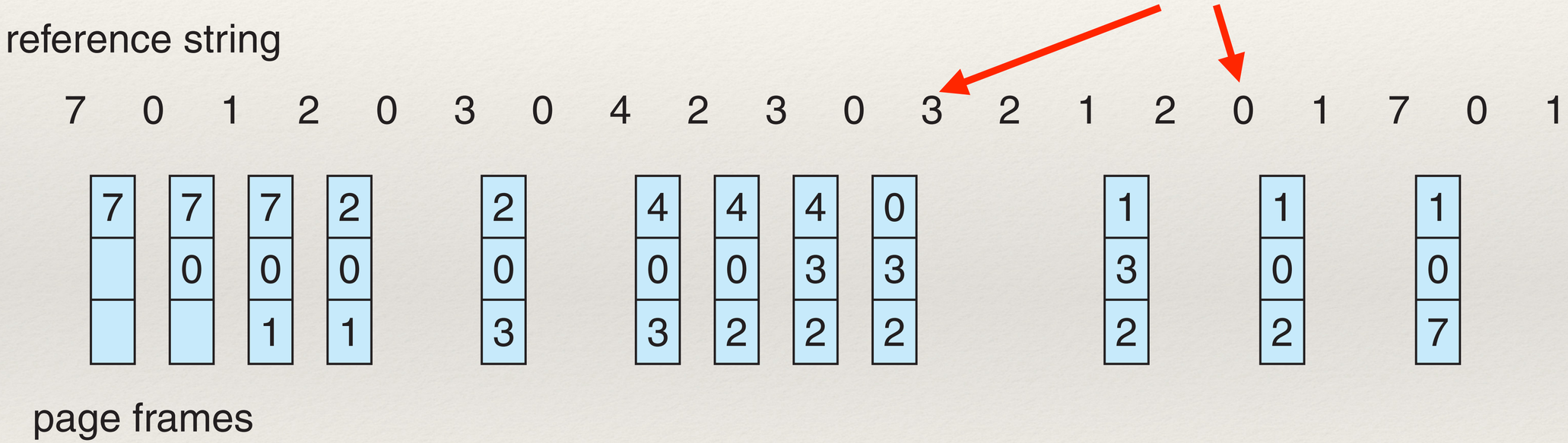
LRU Page Replacement



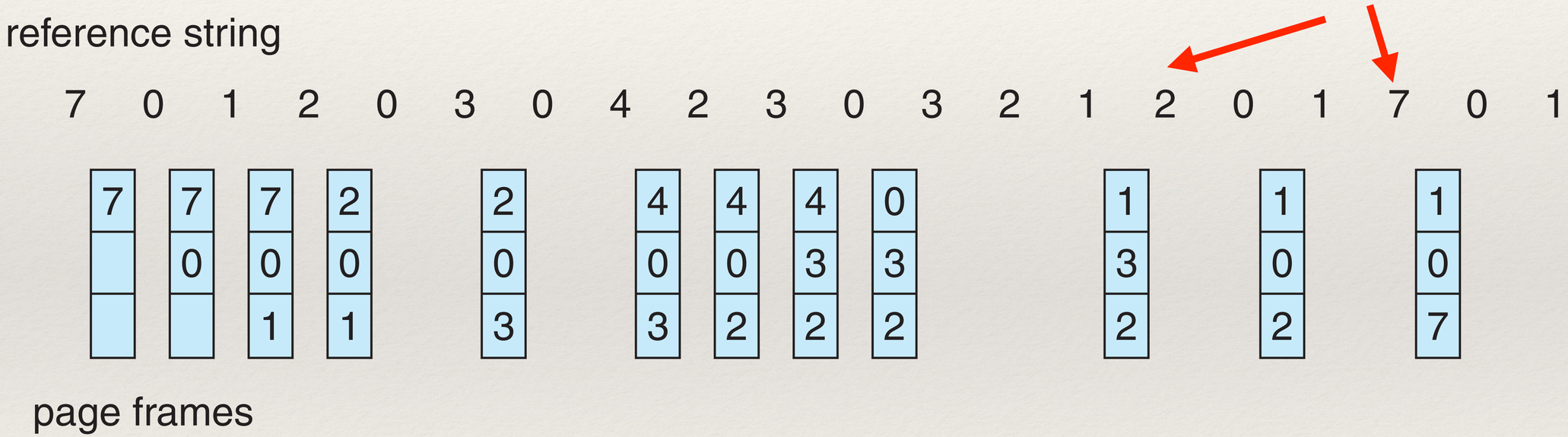
LRU Page Replacement



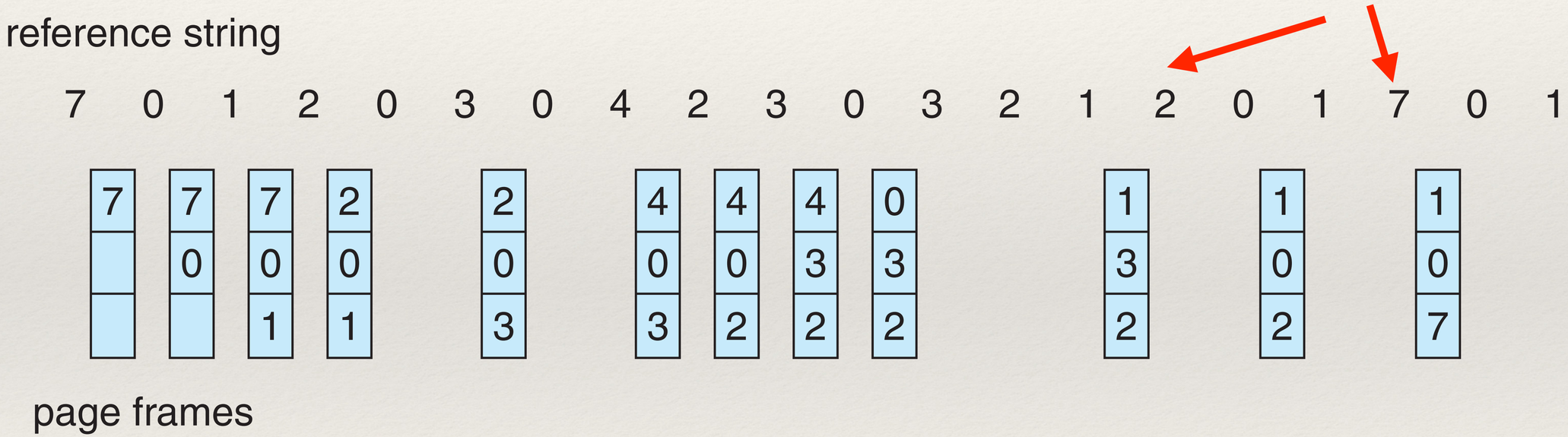
LRU Page Replacement



LRU Page Replacement



LRU Page Replacement



LRU Page Replacement Implementation

- Hardware assistance is needed
  - A table lookup in memory will take too long for every page fault!
  - Can use a counter
    - Clock is incremented with every reference
    - Clock register is copied to a ‘time of use’ field in the page-table entry for that page

LRU Page Replacement Implementation

- Alternative approach
  - Use a stack
    - When a page is referenced it is removed from the stack and put on the top
    - Least used page is on the bottom of the stack
    - Also requires hardware assistance

LRU-Approximation Page Replacement

- True LRU can be expensive (in terms of hardware support and time)
- An approximation can be used by setting a reference bit

LRU-Approximation Page Replacement

- When a page is read from or written to, the reference bit is set
- Can use 1 bit, or an 8 bit byte for the reference bit, set the right most bit.
- After a period of time the OS will shift all bits by 1

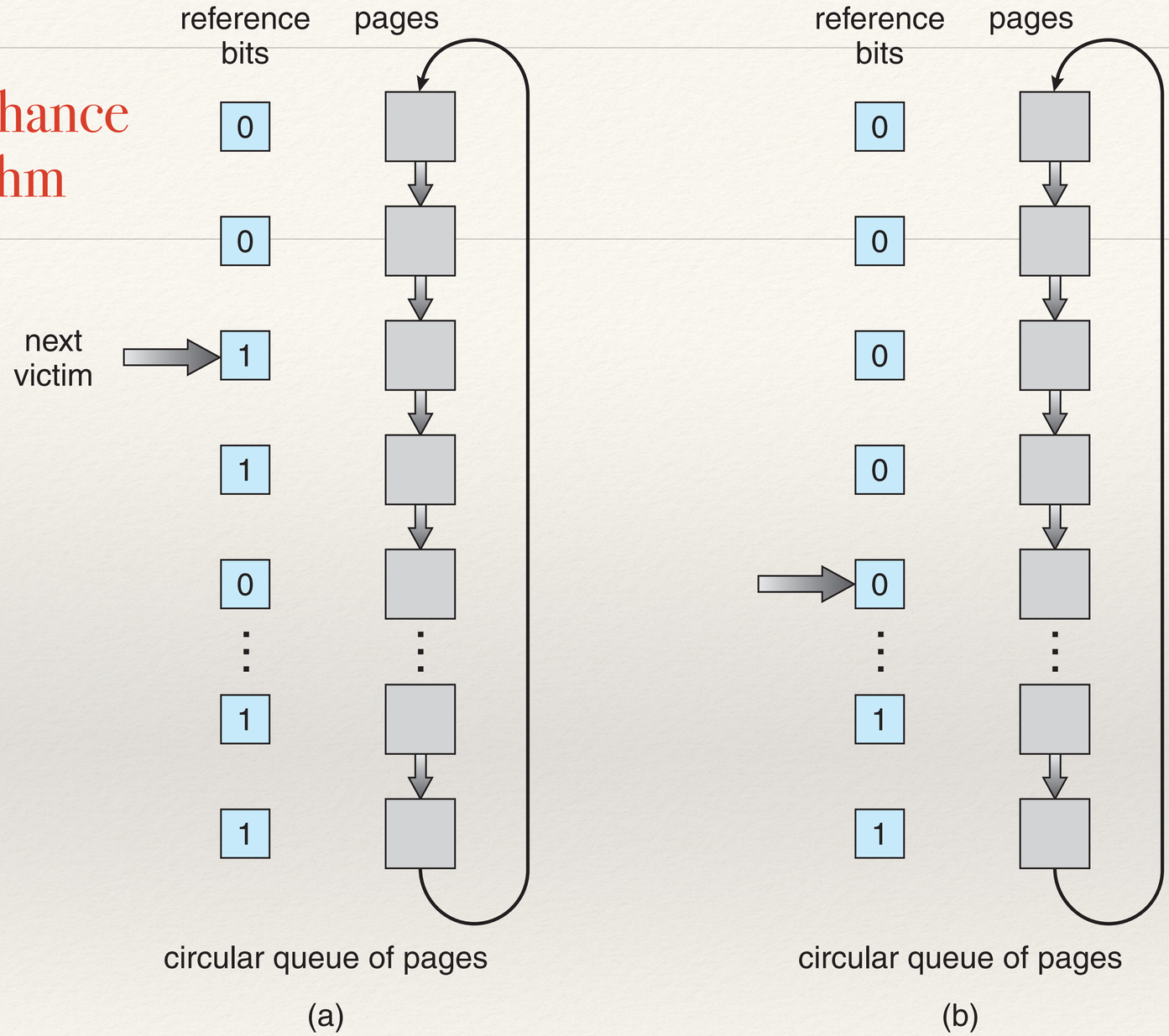
LRU-Approximation Page Replacement

- 00000000 - This page has not been utilized in the last 8 cycles
- 11111111 - This page has been utilized every time in the last 8 cycles
- 00000010 - This page was accessed 2 cycles ago, but no other times
- 10000000 - This page was accessed 8 cycles ago, but no other times
- 11110000 - This page was accessed for 4 cycles, then not accessed for 4 cycles

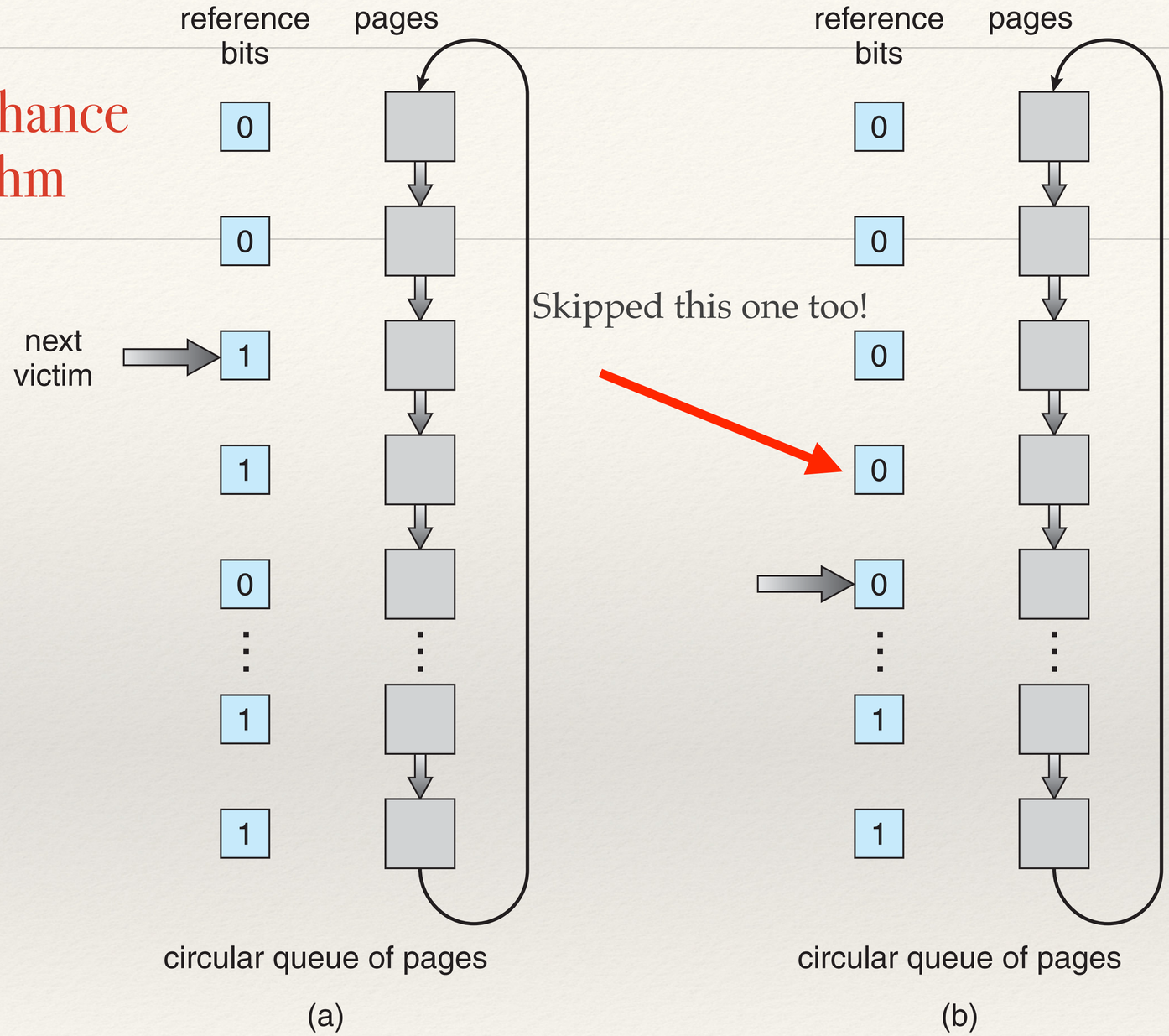
Second-Chance Algorithm

- Utilize a FIFO replacement algorithm
- A single reference bit is used
- Your turn to be replaced? Second chance if the reference bit is set

Second-Chance Algorithm



Second-Chance Algorithm



Enhanced Second Chance Algorithm

- Add a ‘modify’ bit to the reference bit
- (0, 0) not recently used nor modified (Best page to replace)
- (0, 1) not recently used but modified - page will need to be written
- (1, 0) recently used but not modified - may be used again?
- (1, 1) recently used and modified - worse choice

Counting-Based Page Replacement

- Least frequently used (LFU)
  - The page with the smallest count is replaced
  - Problem: a page could be used heavily initially and then never used again
  - Solution: shift bits at regular intervals
- Most frequently used (MFU)
  - The logic with this is that the page with the smallest count was probably just brought in and has yet to be used

Counting-Based Page Replacement

- Least frequently used (LFU)
- Most frequently used (MFU)

Both are expensive and do not approximate the optimum algorithm well

Page Replacement

- Basic Page Replacement
- FIFO Page Replacement
- Optimal Page Replacement
- LRU Page Replacement
- Page Buffering Algorithms

Page-Buffering Algorithms

- Create a pool of free frames
- Victim Frame is selected
- Page brought directly to a page from the free pool
- After the victim frame is written out, added to the free pool

Page-Buffering Algorithms Modifications

- Keep track of what frames are in the “free” pool
- May get lucky and reuse it (since the frame contents are not modified when the frame is written to disk)

Applications and Page Replacement

- Certain applications may understand their need for memory management and I/O buffering better than the operating system
  - Database, Data warehouses
  - OS can give access to a large sequential array of logical blocks without filesystem data structures (raw disk) for the application to manage.