BITS Pilani
K K Birla Goa Campus

Operating Systems

Dr. Lucy J. Gudino
Dept. of CS and IS

Last Class

- Advantages of virtual memory
- Virtual memory implementation
  - Demand paging
  - Demand segmentation
- Demand paging
  - similar to swapping
  - uses **Lazy swapper**
    - Swapper that deals with pages is a **pager**
- Advantages of demand paging
  - Less I/O needed
  - Less memory needed
  - Faster response
  - More users

Last Class…

- valid – invalid bit
- Procedure to be followed:

Refer page table. If the entry is invalid then issue page fault trap

a) Operating system looks at another table to decide:
   - Invalid reference $\Rightarrow$ abort
   - Just not in memory
b) Find empty frame
c) Swap page into frame
d) Reset tables
e) Set validation bit = **v**
f) Restart the instruction that caused the page fault

Last Class…

- Performance of demand paging
  - Effective Access Time (EAT)

    EAT = (1 – p) x ma + p x page fault time
  - where Page Fault Rate p: $0 \leq p \leq 1.0$
    - if $p = 0$ no page faults
    - if $p = 1$, every reference is a fault

    ma : memory access

Copy-on-Write

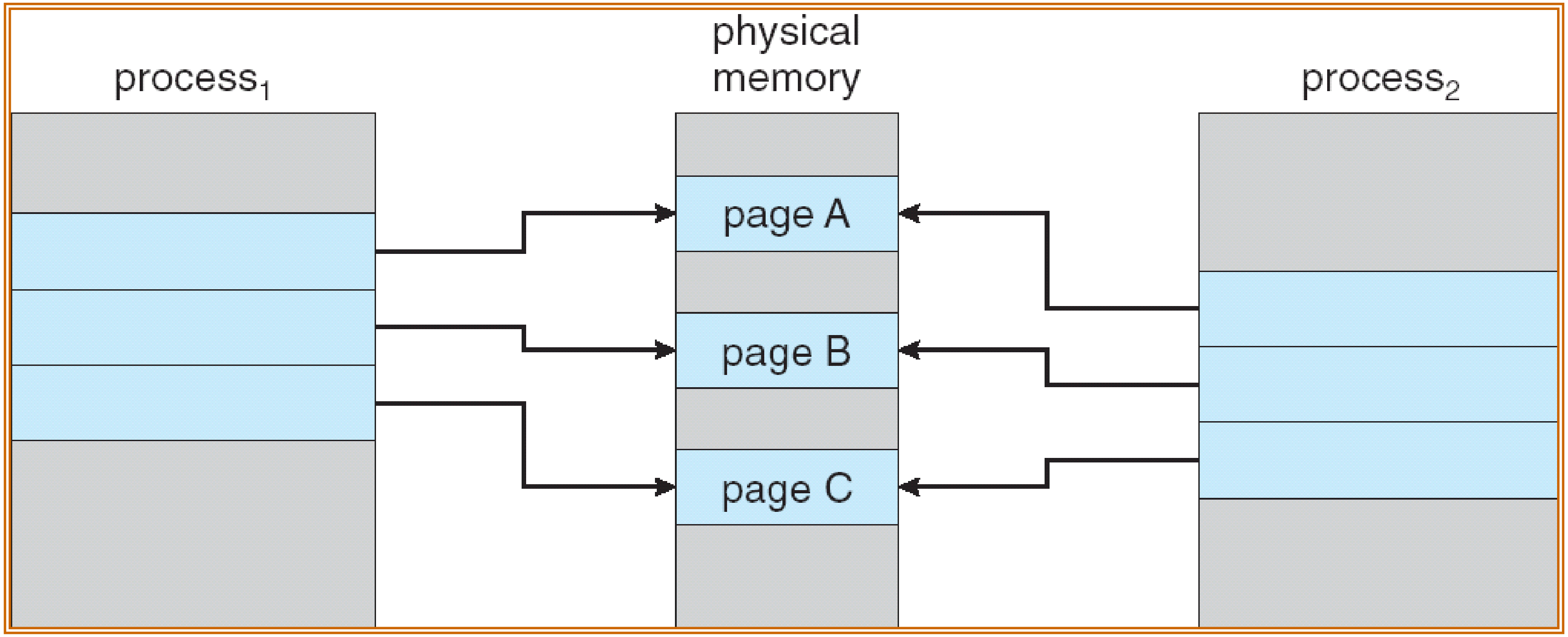
- Copy-on-Write (COW) allows both parent and child processes to initially *share* the same pages in memory

  If either process modifies a shared page, only then is the page copied

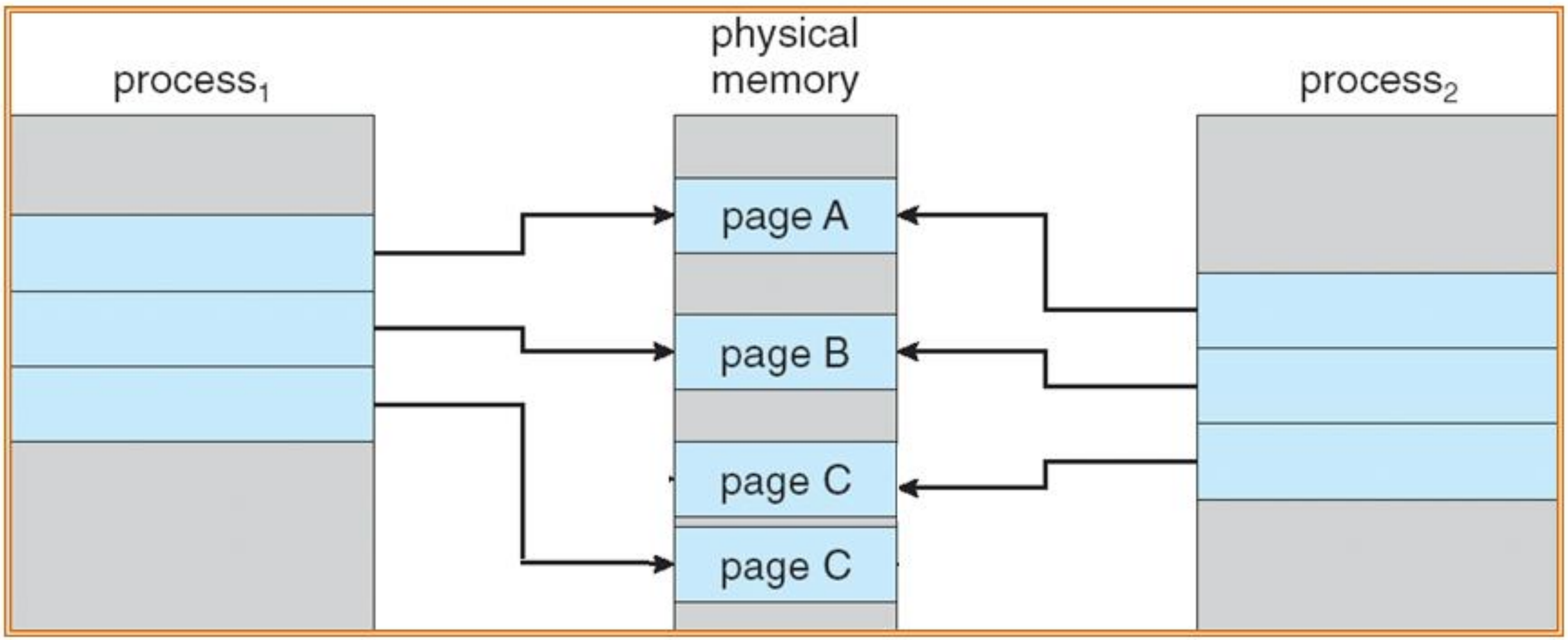
- COW allows more efficient process creation as only modified pages are copied

- Free pages are allocated from a **pool** of zeroed-out pages

Before Process 1 Modifies Page C

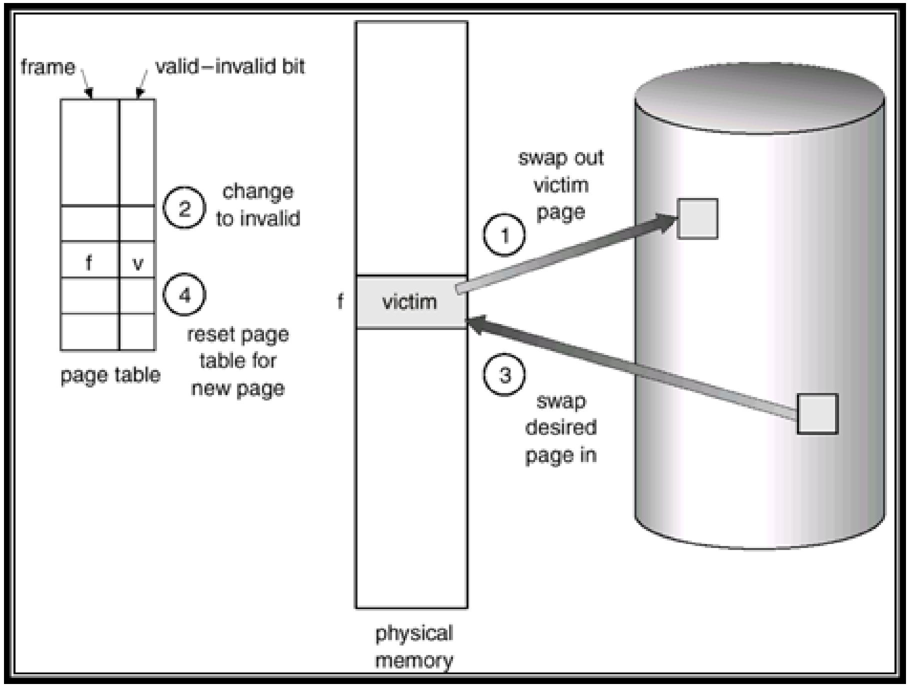


After Process 1 Modifies Page C



What happens if there is no free frame?

- Page replacement – find some page in memory, but not really in use, swap it out
  - use a page replacement algorithm to select a *victim* frame
  - performance – want an algorithm which will result in minimum number of page faults
- Same page may be brought into memory several times



Procedure

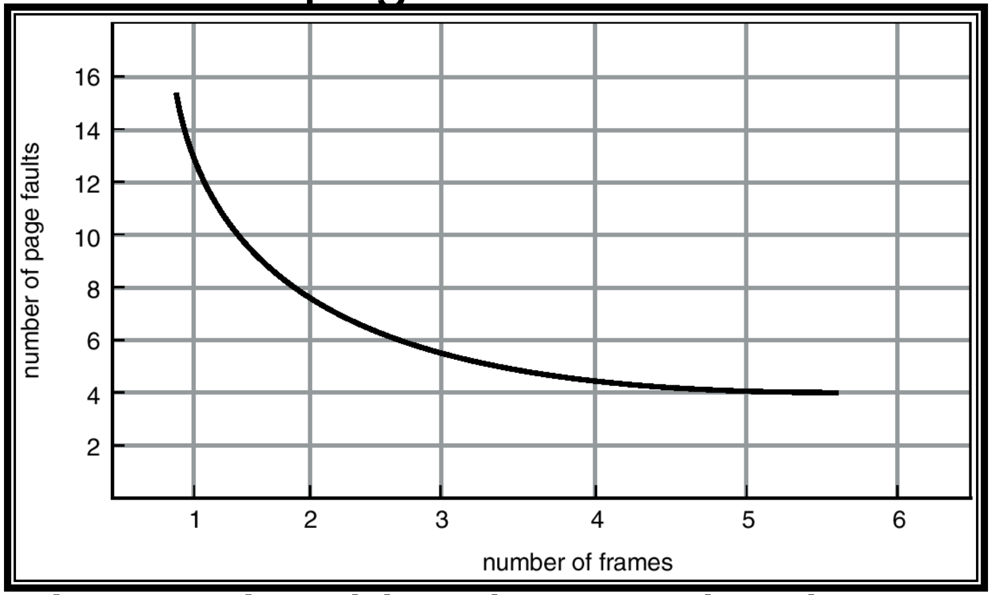
1. Find the location of the desired page on the disk.
2. Find a free frame:
   a) If there is a free frame, use it.
   b) If there is no free frame, use a page-replacement algorithm to select a victim frame.
   c) Write the victim frame to the disk; change the page and frame tables accordingly.
3. Read the desired page into the newly freed frame; change the page and frame tables.
4. Restart the user process.

Contd…

- if no frames are free, *two page transfers (one out and one in) are* required.
- Main drawback : doubles the page-fault service time and increases the effective access time accordingly.
- Use **modify (dirty) bit** to reduce overhead of page transfers – only modified pages are written to disk
- Demand paging requires two algorithms:
  - frame – allocation algorithm
  - page – replacement algorithm
- Frame allocation algorithm : handles frame allocation to each process.
- page – replacement algorithm : deals with the frames that are to be replaced

Page Replacement Algorithms

- Want lowest page-fault rate.



- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string.
- In all our examples, the reference string is

  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

First-In-First-Out (FIFO) Algorithm

Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

3 frames (3 pages can be in memory at a time per process)

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

9 page faults

4 frames

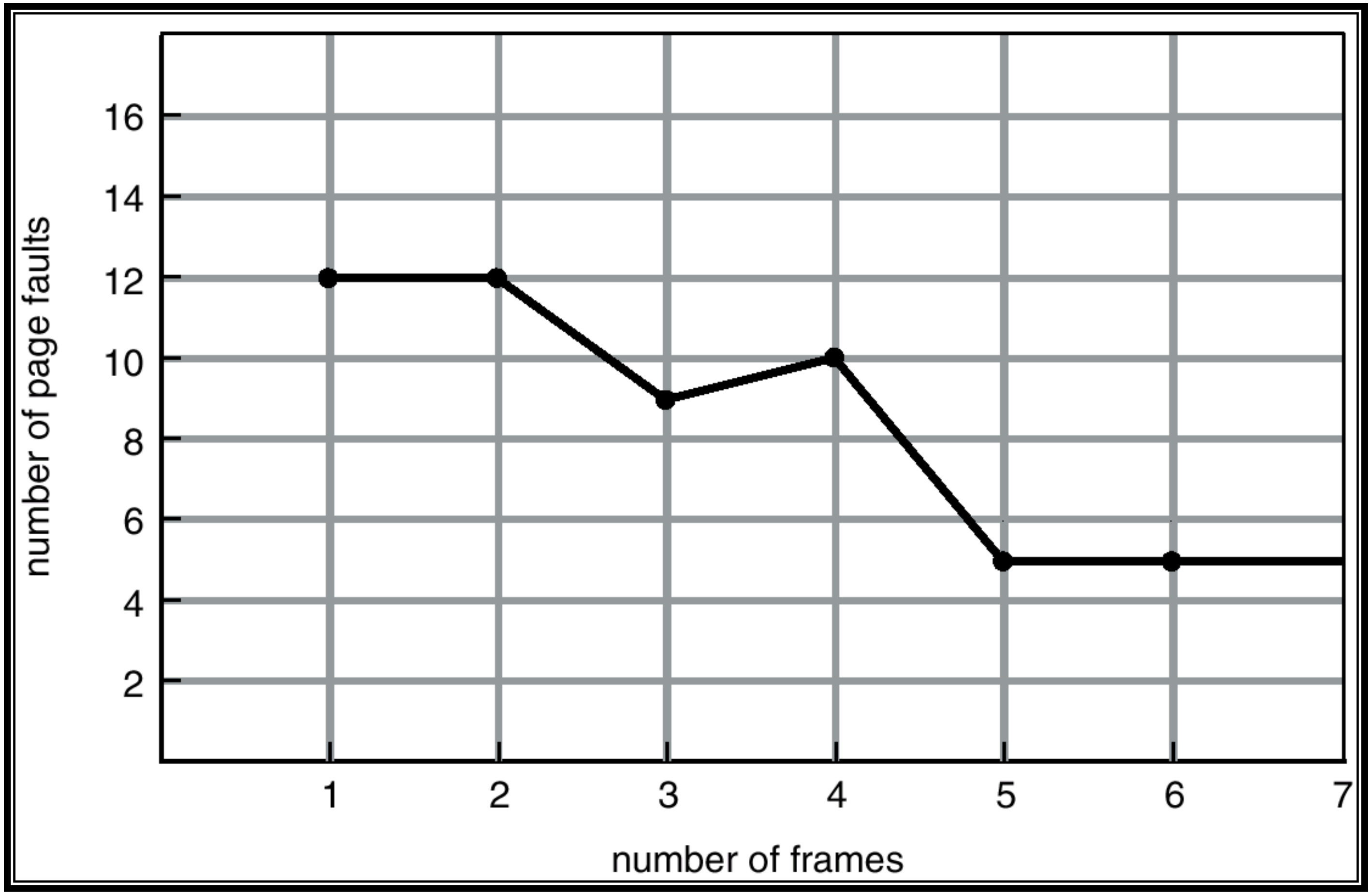
1	1	5	4
2	2	1	5
3	3	2	
4	4	3	

10 page faults

FIFO Replacement – Belady's Anomaly

- more frames $\Rightarrow$ more page faults

FIFO Illustrating Belady's Anamoly



Example 2

Consider the following sequence of address

0100, 0432, 0101,0612, 0102, 0103, 0104, 0101, 0611, 0102, 0103, 0104,0101,0610, 0102, 0103, 0104, 0101, 0609, 0102, 0105

Assume 100 byte page.

Find out the reference string.

Optimal Algorithm

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

4 frames example

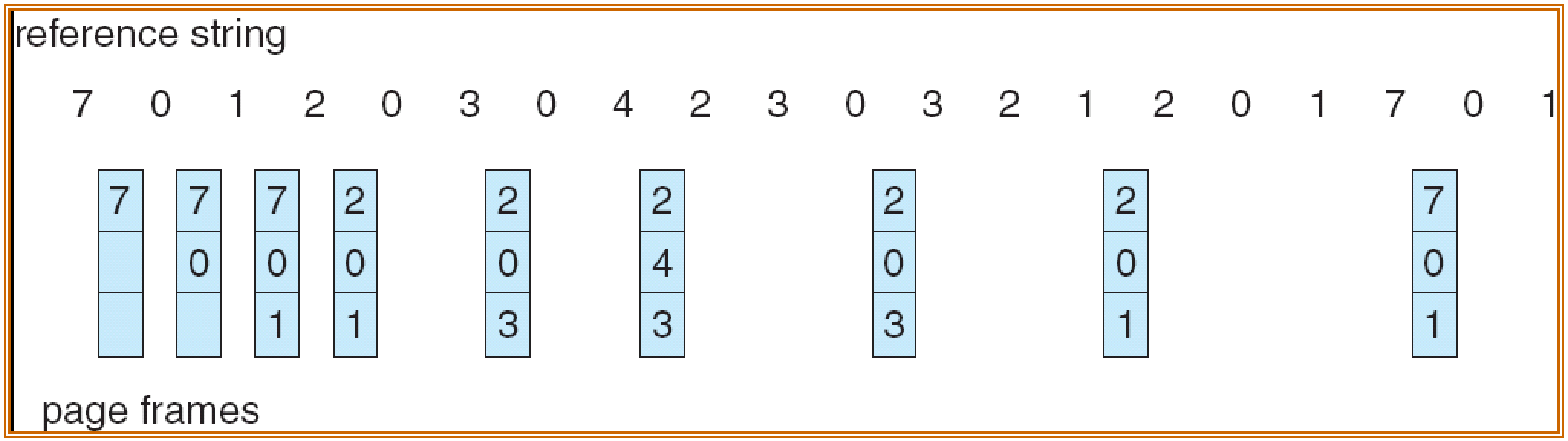
1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	4
2	
3	
4	5

6 page faults

Main drawback: requires future knowledge of the reference string.

Example 2:


reference string
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7 7 7 2 2 2 2 2 7
0 0 0 0 4 0 0 0
1 1 3 3 3 1 1
page frames

Least Recently Used (LRU) Algorithm

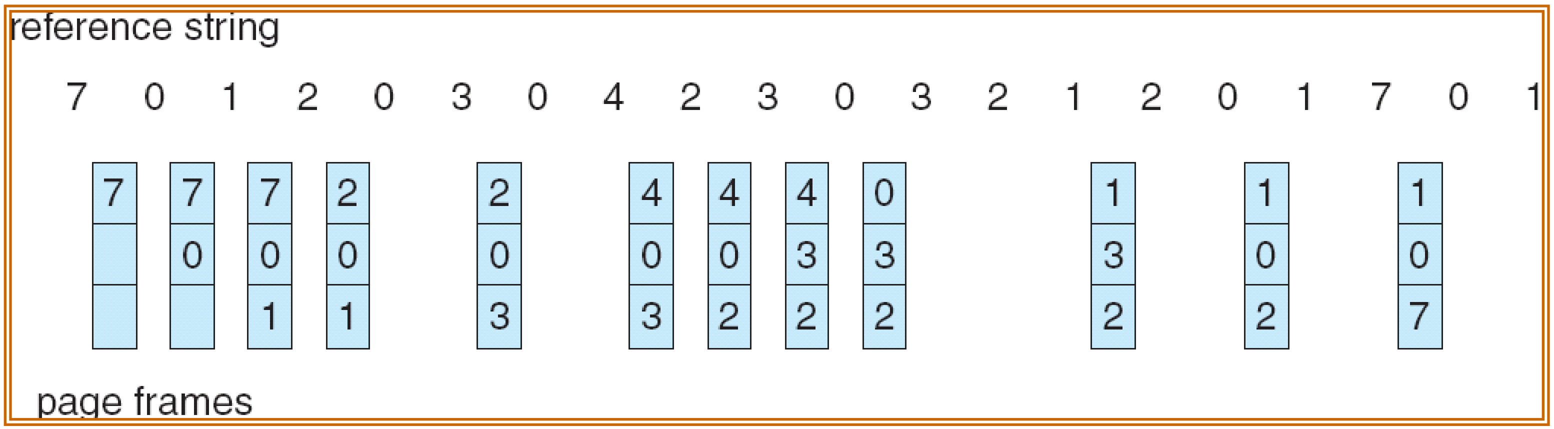
Reference string: 1, 2, 3, 4, 1, 2, **5**, 1, 2, **3**, **4**, **5**

1	1	1	1	**5**
2	2	2	2	2
3	**5**	5	**4**	4
4	4	**3**	3	3

Counter implementation

- Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter
- When a page needs to be changed, look at the counters to determine which are to change

Example 2


reference string
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7
7 0
7 0 1
2 0 1
2 0 3
4 0 3
4 0 2
4 3 2
0 3 2
1 3 2
1 0 2
1 0 7
page frames