Amirkabir University of Technology
(Tehran Polytechnic)

Operating Systems

Virtual Memory-Page Replacement

Seyyed Ahmad Javadi

sajavadi@aut.ac.ir

Fall 2023

What Happens if There is no Free Frame?

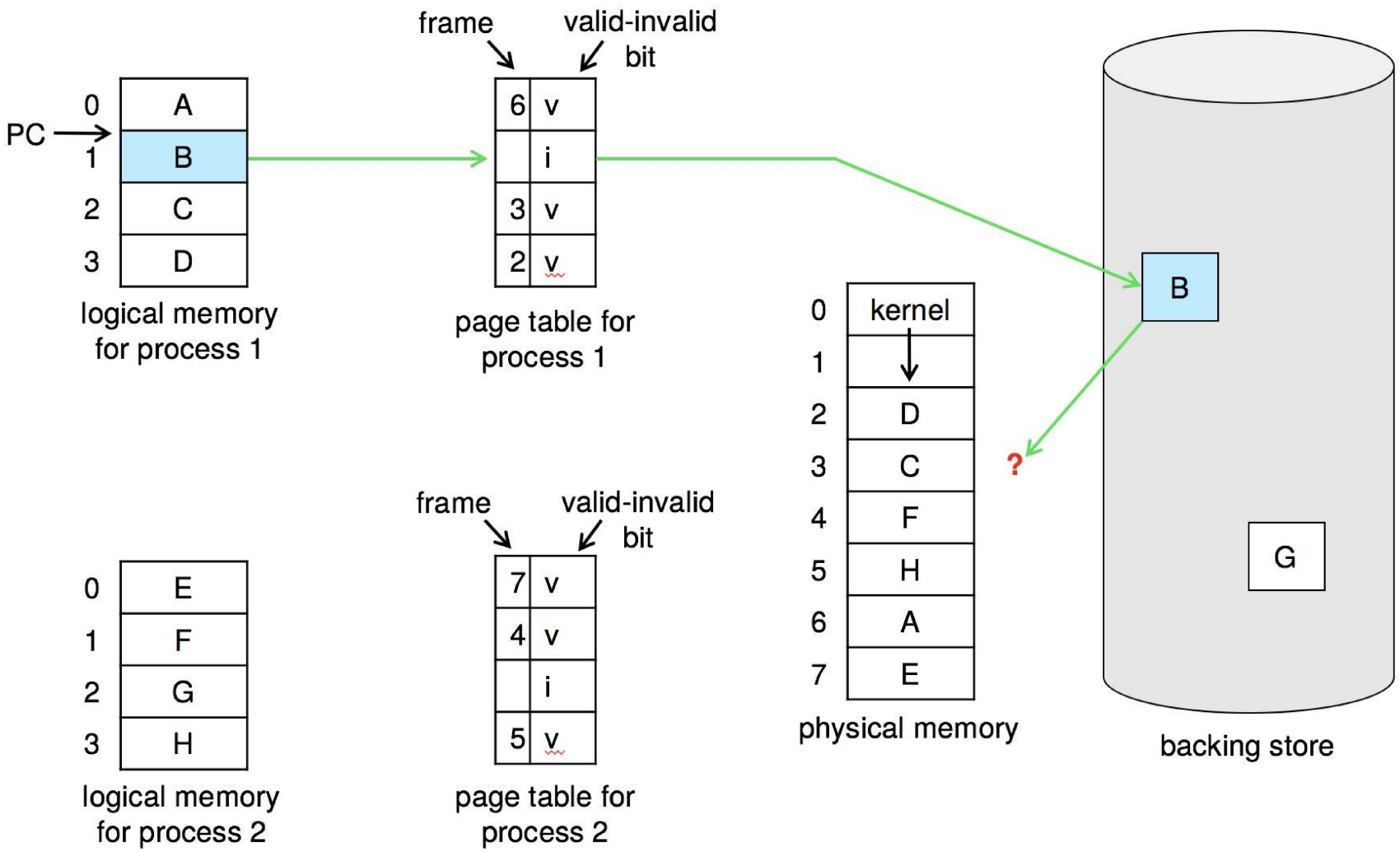
- Used up by process pages
- Also in demand from the kernel, I/O buffers, etc

- How much to allocate to each?

What Happens if There is no Free Frame? (cont.)

- Page replacement – find some page in memory, but not really in use, page it out
  - Algorithm – terminate? swap out? replace the page?
  - Performance – want an algorithm which will result in minimum number of page faults
- Same page may be brought into memory several times

Need For Page Replacement

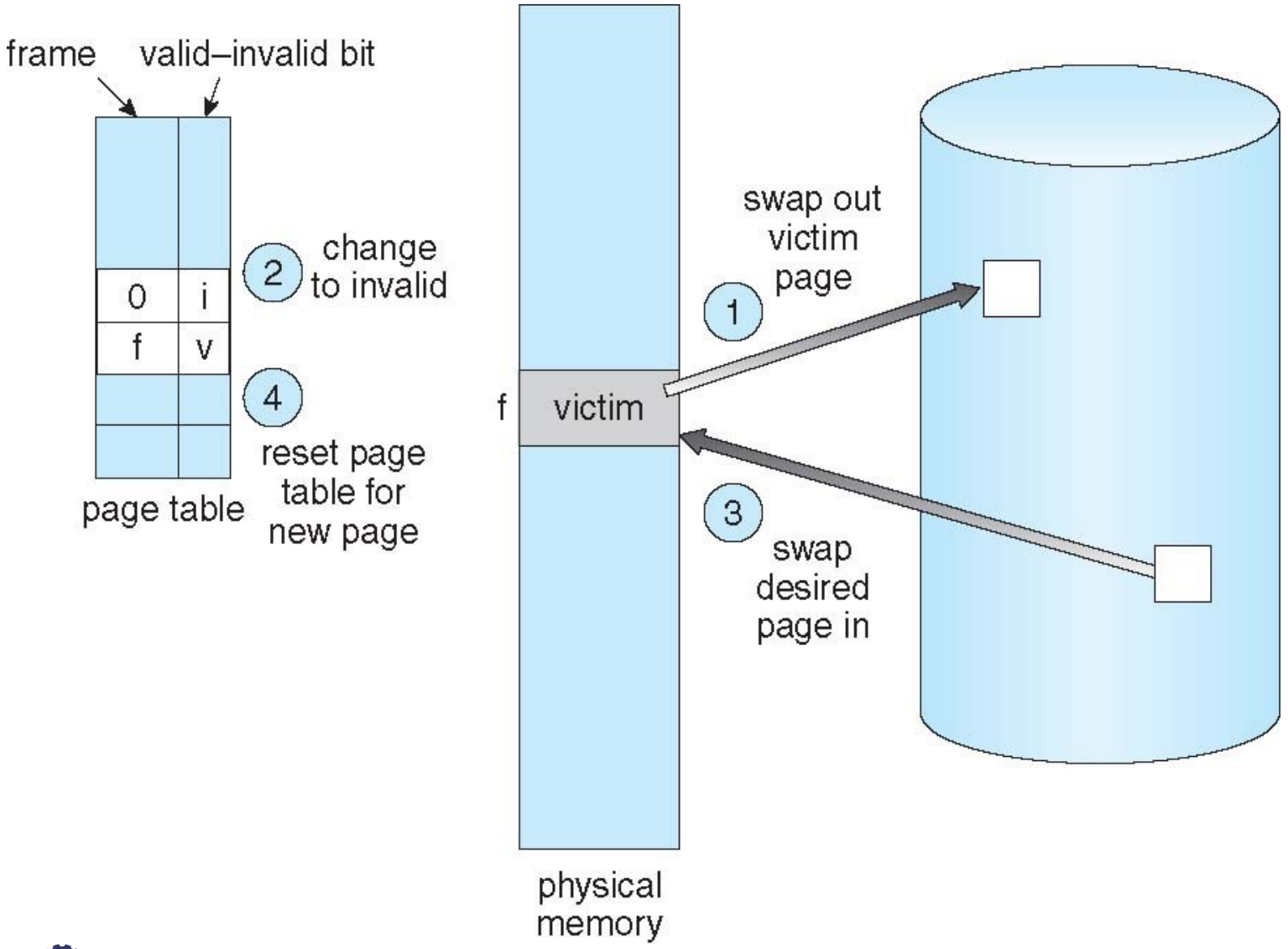


Basic Page Replacement

1. Find the 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***
   - Write victim frame to disk if dirty
3. Bring the desired page into the (newly) free frame; update the page and frame tables
4. Continue the process by restarting the instruction that caused the trap

Note now potentially 2 page transfers for page fault – increasing EAT

Page Replacement



Page and Frame Replacement Algorithms

- **Frame-allocation algorithm** determines
  - How many frames to give each process
  - Which frames to replace

- **Page-replacement algorithm**
  - Want lowest page-fault rate on both first access and re-access

Page and Frame Replacement Algorithms

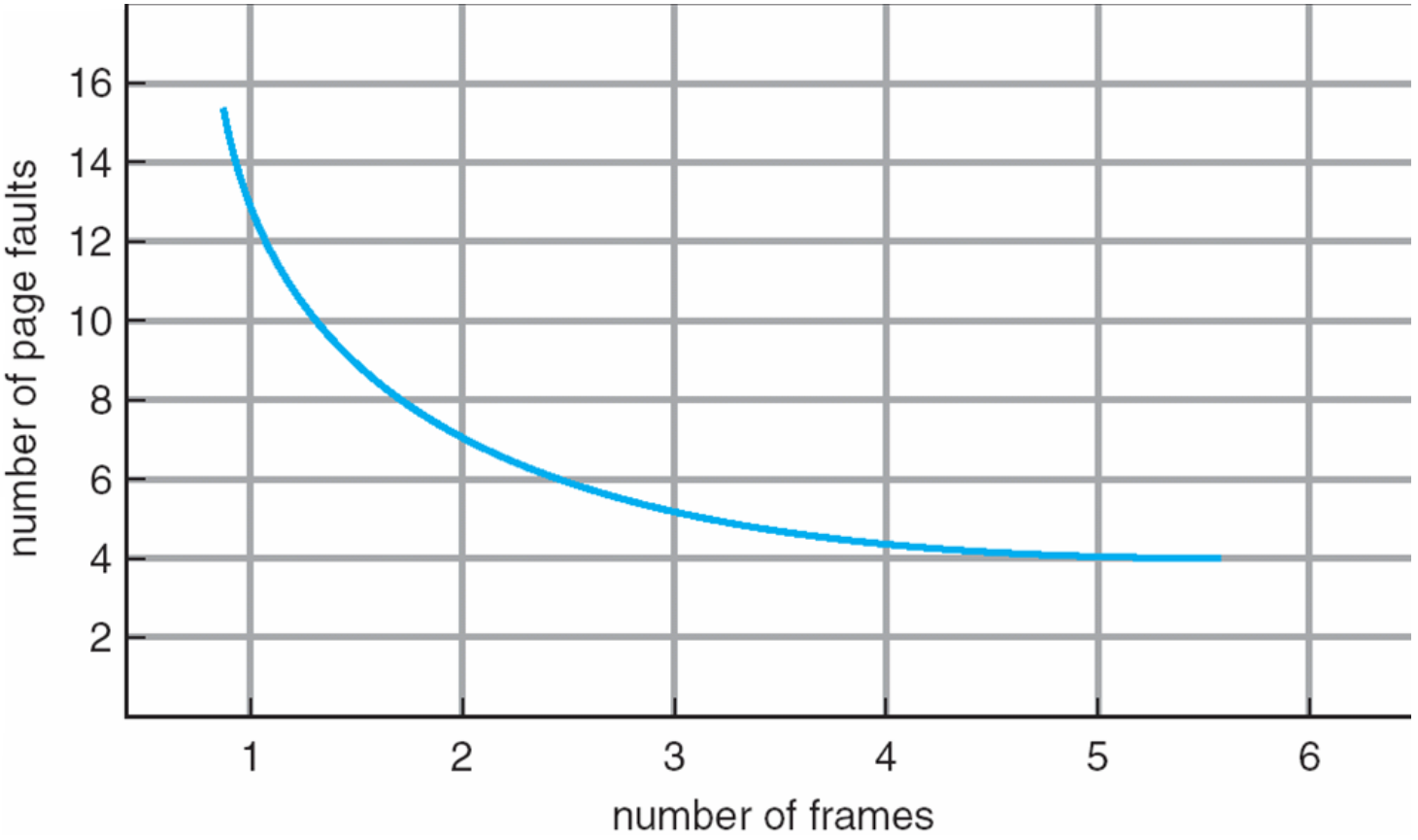
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string
  - String is just page numbers, not full addresses
  - Repeated access to the same page does not cause a page fault
  - Results depend on ***number of frames available***

Page and Frame Replacement Algorithms (cont.)

- In all our examples, the **reference string** of referenced page numbers is

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

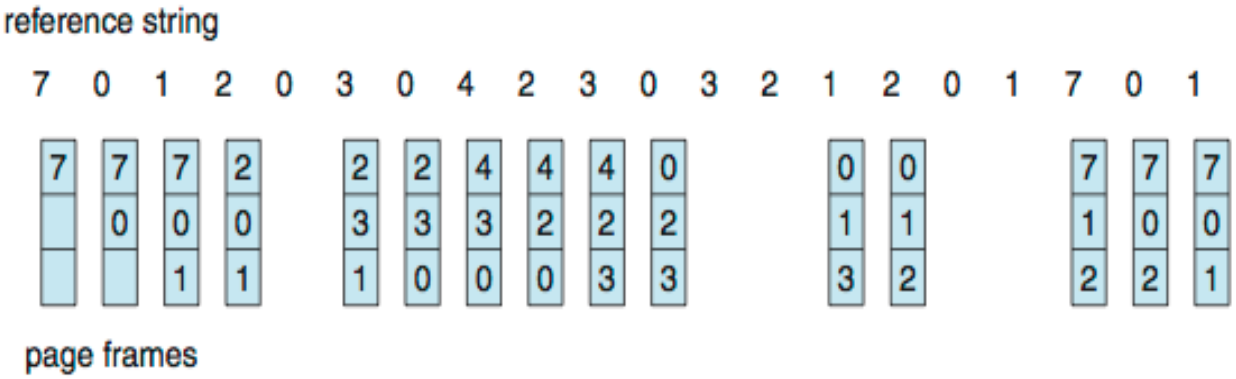
Graph of Page Faults Versus the Number of Frames



First-In-First-Out (FIFO) Algorithm

- Reference string: **7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1**
- 3 frames (3 pages can be in memory at a time per process)
- How many page faults?

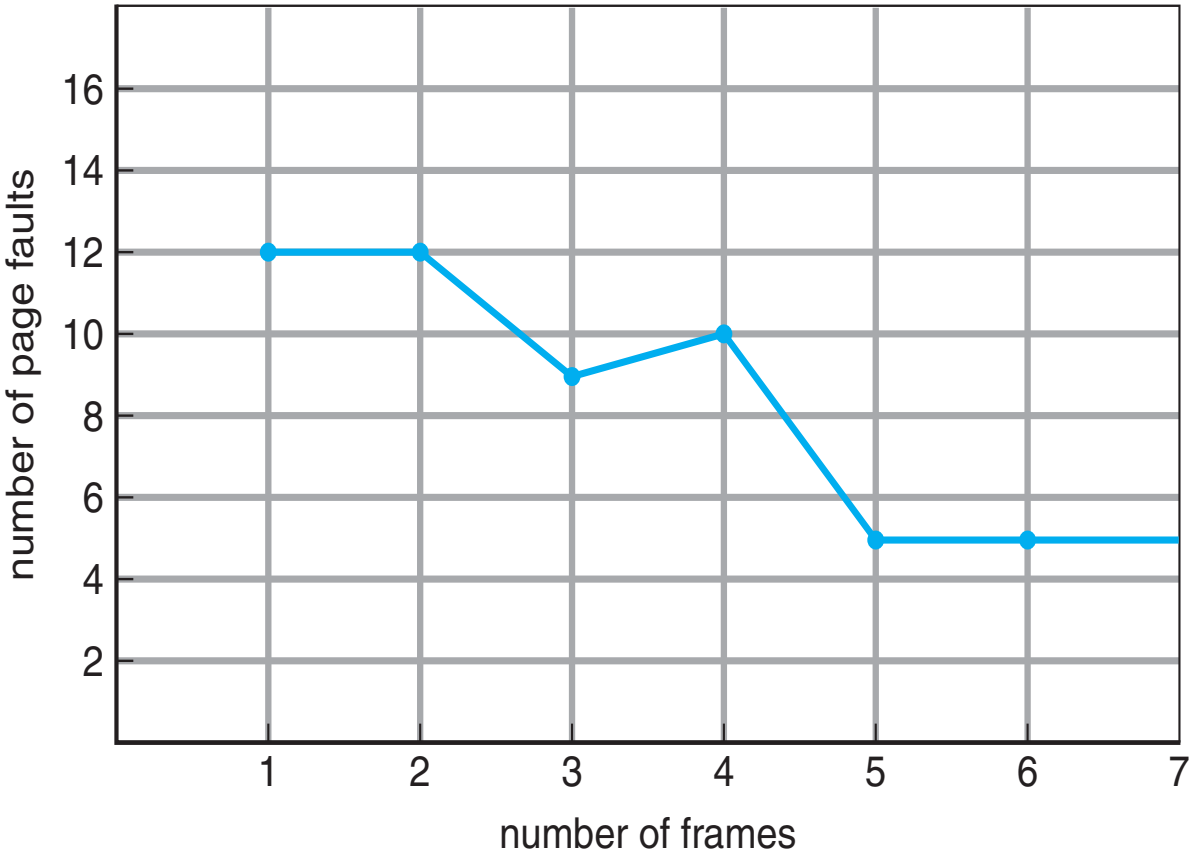
15 page faults



First-In-First-Out (FIFO) Algorithm

- Can vary by reference string: consider 1,2,3,4,1,2,5,1,2,3,4,5
  - Adding more frames can cause more page faults!
    - **Belady's Anomaly**
      - for some page-replacement algorithms, the page-fault rate may increase as the number of allocated frames increases

FIFO Illustrating Belady's Anomaly



First-In-First-Out (FIFO) Algorithm

- Can vary by reference string: consider 1,2,3,4,1,2,5,1,2,3,4,5
  - Adding more frames can cause more page faults!
    - **Belady's Anomaly**
      - for some page-replacement algorithms, the page-fault rate may increase as the number of allocated frames increases
- How to track ages of pages?
  - Just use a FIFO queue

Belady's Anomaly in FIFO – Example

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frame → 9 page faults

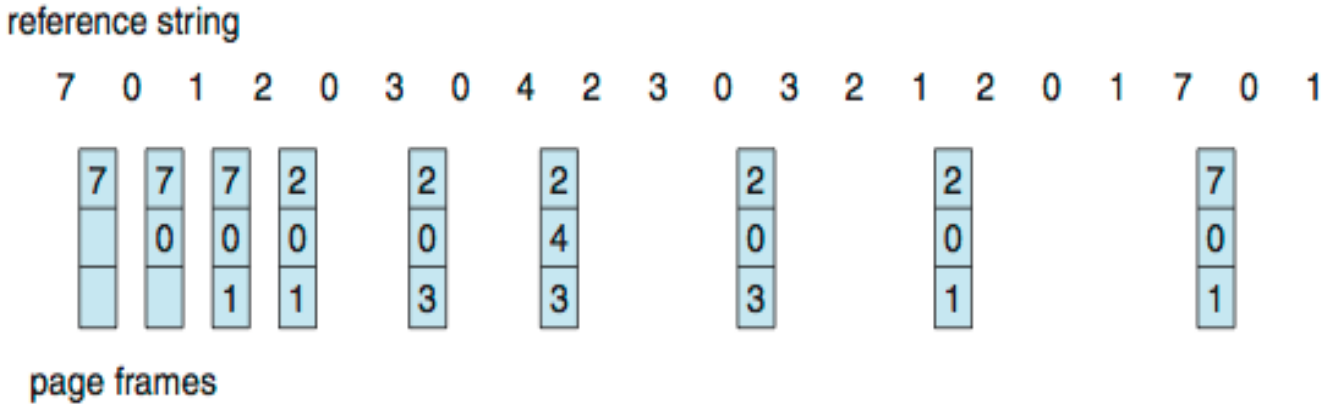
1	1	1	2	3	4	1	1	1	2	5	5
	2	2	3	4	1	2	2	2	5	3	3
		3	4	1	2	5	5	5	3	4	4
PF	PF	PF	PF	PF	PF	PF	X	X	PF	PF	X

- 4 frame → 10 page faults

1	2	3	4	1	2	5	1	2	3	4	5
	1	2	3	4	1	2	5	1	2	3	4
		1	2	3	4	1	2	5	1	2	3
			1	2	3	4	4	4	5	1	2
PF	PF	PF	PF	X	X	PF	X	X	PF	PF	PF

Optimal Algorithm

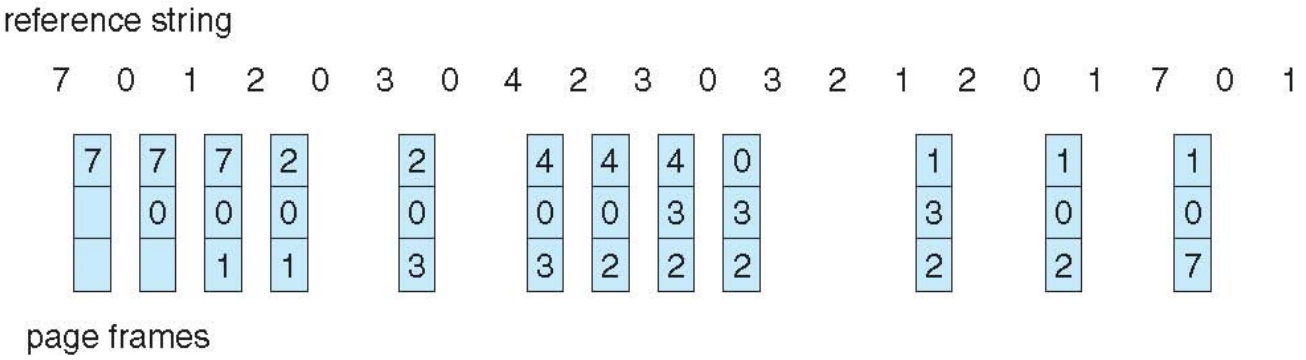
- Replace page that will not be used for longest period of time
  - 9 is optimal for the example



- How do you know this?
  - Can't read the future
- Used for measuring how well your algorithm performs

Least Recently Used (LRU) Algorithm

- Use past knowledge rather than future
- Replace page that has not been used in the most amount of time
- Associate time of last use with each page



- 12 faults – better than FIFO but worse than OPT
- Generally good algorithm and frequently used
- But how to implement?

LRU Algorithm (cont.)

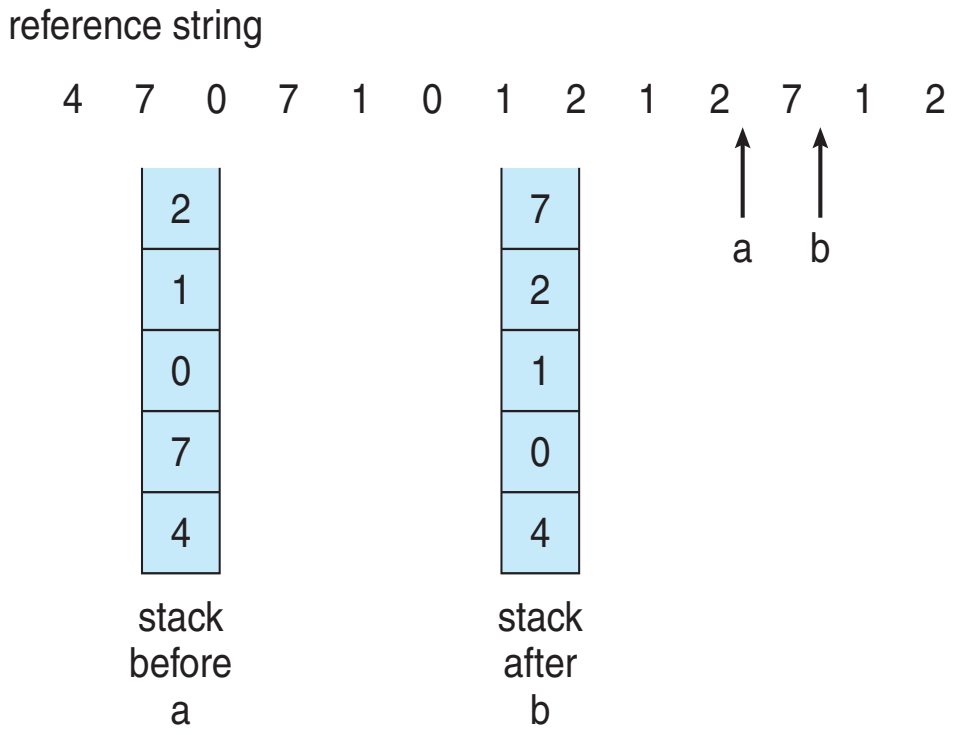
- Counter implementation
  - Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter
  - When a page needs to be changed, look at the counters to find smallest value
    - Search through table needed

LRU Algorithm (cont.)

- Stack implementation
  - Keep a stack of page numbers in a double link form:
  - Page referenced:
    - move it to the top
    - requires 6 pointers to be changed
  - But each update more expensive
  - No search for replacement

LRU Algorithm (Cont.)

- LRU and OPT are cases of **stack algorithms** that don't have Belady's Anomaly
- Use Of A Stack to Record Most Recent Page References



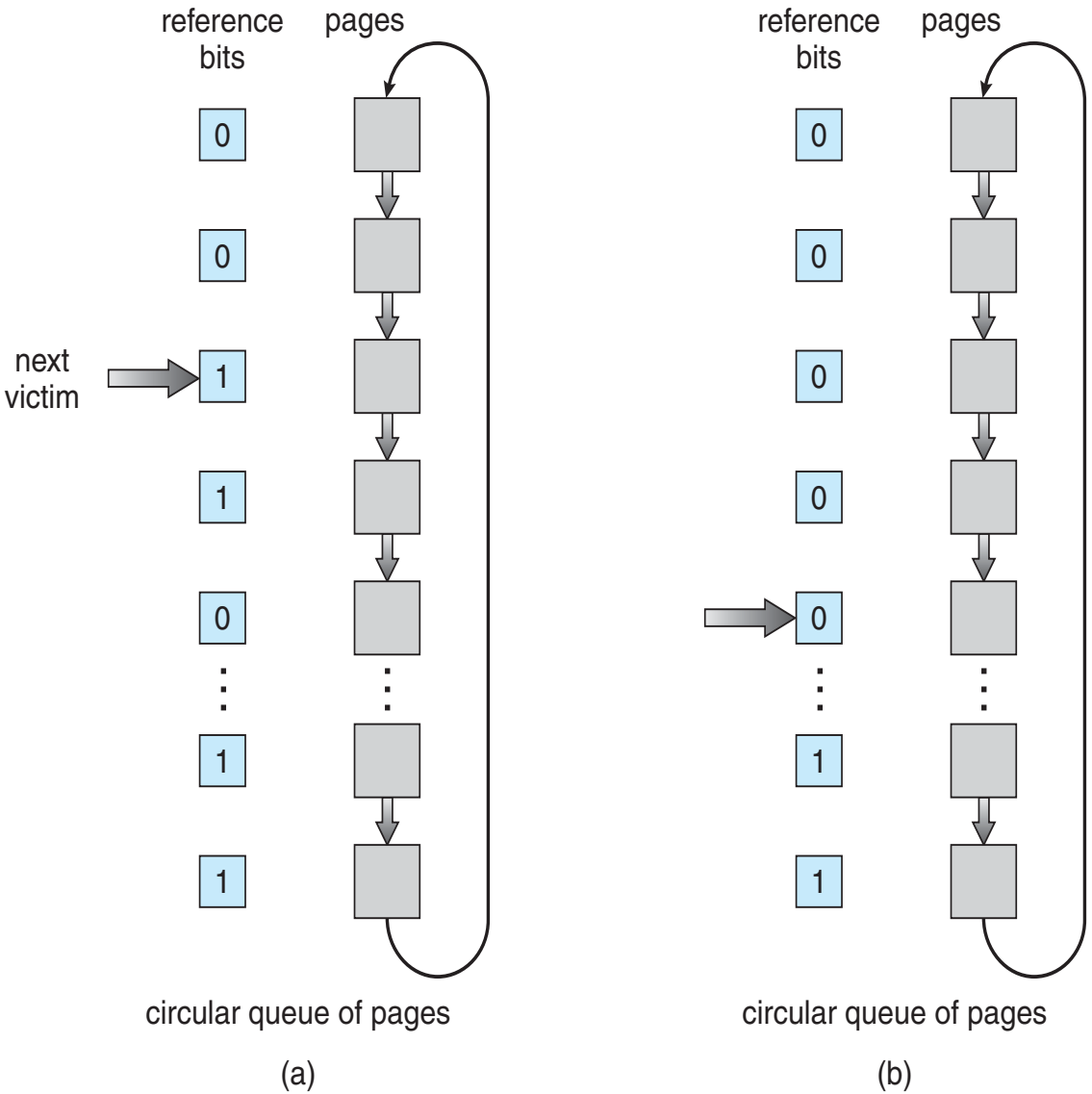
LRU Approximation Algorithms

- LRU needs special hardware and still slow
- **Reference bit**
  - With each page associate a bit, initially = 0
  - When page is referenced bit set to 1
  - Replace any with reference bit = 0 (if one exists)
    - We do not know the order, however

LRU Approximation Algorithms (cont.)

- **Second-chance algorithm**
  - Generally FIFO, plus hardware-provided reference bit
  - **Clock** replacement
  - If page to be replaced has
    - Reference bit = 0 -> replace it
    - reference bit = 1 then:
      - set reference bit 0, leave page in memory
      - replace next page, subject to same rules

Second-chance Algorithm



Enhanced Second-Chance Algorithm

- Improve algorithm by using reference bit and modify bit (if available)
- Take ordered pair (reference, modify):
  1. (0, 0) neither recently used not modified – best page to replace
  2. (0, 1) not recently used but modified – not quite as good, must write out before replacement
  3. (1, 0) recently used but clean – probably will be used again soon
  4. (1, 1) recently used and modified – probably will be used again soon and need to write out before replacement

Enhanced Second-Chance Algorithm (cont.)

- When page replacement called for, use the clock scheme but use the four classes.
- Replace page in lowest non-empty class
  - Might need to search circular queue several times
- The major difference between this algorithm and the simpler clock algorithm is that here we give preference to those pages that have been modified to reduce the number of I/Os required.